Course Code: Doctor rer. nat.
PhD Thesis: Computer Science
March 2025

Abstract Ensembles for Anomaly Detection and Beyond

Simon Klüttermann

Faculty of Computer Science
TU Dortmund University
Dortmund, Germany

A Study on

Abstract Ensembles for Anomaly Detection and Beyond

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor rer. nat.

in

Computer Science

by

Simon Klüttermann

to

Faculty of Computer Science

TU Dortmund University

Dortmund - 2025, Germany

March 2025

Abstract

Detecting anomalies is essential in domains such as cybersecurity, industrial monitoring, and scientific discovery, yet current methods often struggle with computational efficiency, robustness, reliability and interpretability. Similar to most machine learning tasks, ensemble methods have shown great promise here. Traditional ensemble methods combine powerful anomaly detection models, assuming that diverse models reduce individual errors. While this slightly improves performance, it comes at a high computational cost. In contrast, we explore ensembles composed of simple, specialized submodels that, individually, are weak detectors but collectively form an effective system. As we will show, this approach enables the construction of large and effective ensembles while maintaining computational efficiency, as each submodel focuses on solving a simple, abstract subtask. We refer to such ensembles as abstract ensembles.

This thesis advances the study of abstract ensembles in four key ways. First, we introduce a novel deep-learning framework for anomaly detection based on abstract ensembles, providing an effective and easy-to-use training methodology. Next, we enhance this approach by proposing a shallow variant optimized for speed and a test-time training mechanism for improved performance. Furthermore, we analyze the advantages of abstract ensembles in terms of interpretability and adversarial robustness, demonstrating their potential for more transparent and resilient anomaly detection. Finally, we extend these ideas beyond anomaly detection to re-identification tasks, illustrating the broader applicability of abstract ensemble methods.

This thesis establishes abstract ensembles as a principled, efficient, and robust approach to anomaly detection and related tasks with significant theoretical and practical implications.

ACKNOWLEDGMENTS

My thanks first go to the creators of this template: Anthony Dang and Dr. Chandranath Adak, who saved countless hours better spent on actual research.

During my research, I was profoundly fortunate to have collaborated with exceptional co-authors, particularly Jérôme Rutinowski. Their insights were instrumental in shaping this thesis.

I also wish to acknowledge my brilliant students, whose curiosity and dedication allowed me to explore many strange and novel concepts.

Above all, I remain eternally grateful to my academic advisor for their steadfast mentorship and for nurturing my passion for scientific inquiry.

Finally, my deepest appreciation goes to my family and friends for their unwavering support and patience during my countless tirades about esoteric machine-learning phenomena over the years.

List of Publications

Related to the Thesis

1. Böing, B., Klüttermann, S., & Müller, E. (2022). Post-robustifying deep anomaly detection ensembles by model selection. *2022 IEEE International Conference on Data Mining (ICDM)*, 861–866. https://doi.org/10.1109/ICDM54844.2022.00098

2. Klüttermann, S., Rutinowski, J., Reining, C., Roidl, M., & Müller, E. (2022). Towards graph representation based re-identification of chipwood pallet blocks. *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1543–1550. https://doi.org/10.1109/ICMLA55696.2022.00279<

3. Klüttermann, S., & Müller, E. (2023). Evaluating and comparing heterogeneous ensemble methods for unsupervised anomaly detection. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN54540.2023.10191405

4. Klüttermann, S., Balestra, C., & Müller, E. (2024). On the efficient explanation of outlier detection ensembles through shapley values. In D.-N. Yang, X. Xie, V. S. Tseng, J. Pei, J.-W. Huang, & J. C.-W. Lin (Eds.), *, pacific-asia conference on knowledge discovery and data mining (pakdd), advances in knowledge discovery and data mining* (pp. 43–55). Springer Nature Singapore. https://doi.org/https://doi.org/10.1007/978-981-97-2259-4_4

5. Klüttermann, S., Rutinowski, J., & Müller, E. (2024). The phenomenon of correlated representations in contrastive learning. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN60899.2024.10649913

6. Klüttermann, S., & Müller, E. (2024). About test-time training for outlier detection [Under review]. *2025 IEEE International Conference on Big Data (BigData)*. https://arxiv.org/abs/2404.03495

7. Klüttermann, S., Peka, V., Doebler, P., & Müller, E. (2024). Towards highly efficient anomaly detection for predictive maintenance [Accepted and presented, to appear]. *Proceedings of the 23rd IEEE International Conference on Machine Learning and Applications (ICMLA), Special Session 3: Machine Learning for Predictive Models in Engineering Applications (MLPMEA)*

8. Klüttermann, S., Rutinowski, J., Polachowski, F., Nguyen, A., Grimme, B., Roidl, M., & Müller, E. (2024). On the effectiveness of heterogeneous ensemble methods for re-identification [Accepted and presented, to appear]. *Proceedings of the 23rd IEEE International Conference on Machine Learning and Applications (ICMLA), Special Session 3: Machine Learning for Predictive Models in Engineering Applications (MLPMEA)*

9. Klüttermann, S., Katzke, T., & Müller, E. (2025). Unsupervised surrogate anomaly detection [To appear]. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*. https://arxiv.org/abs/2504.20733

10. Klüttermann, S., & Müller, E. (2025). Rare anomalies require large datasets: About proving the existence of anomalies [Under review]. *International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*

11. Klüttermann, S., Gupta, S., & Müller, E. (2025). Evaluating anomaly detection algorithms: The role of hyperparameters and standardized benchmarks [Under review]. *International Joint Conference on Artificial Intelligence (IJCAI)*

OTHERS

1. Rutinowski, J., Schrötler, N., Klüttermann, S., Roidl, M., & Janiesch, C. (2024). A laser-based volumetric measurement approach for industrial settings. *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 3595–3601. https://doi.org/10.1109/CASE59546.2024.10711535

2. Rutinowski, J., Klüttermann, S., Endendyk, J., Reining, C., & Müller, E. (2024). Benchmarking trust: A metric for trustworthy machine learning. In L. Longo, S. Lapuschkin, & C. Seifert (Eds.), *Explainable artificial intelligence* (pp. 287–307). Springer Nature Switzerland. https://doi.org/https://doi.org/10.1007/978-3-031-63787-2_15

3. Kumar, V., Srivastava, V., Mahjabin, S., Pal, A., Klüttermann, S., & Müller, E. (2024). Autoencoder optimization for anomaly detection: A comparative study with shallow algorithms. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN60899.2024.10650057

4. Hossain, M. S., Hossain, M. S., Klüttermann, S., & Müller, E. (2024). Evaluating anomaly detection algorithms: A multi-metric analysis across variable class imbalances. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN60899.2024.10650351

5. Ok, M., Klüttermann, S., & Müller, E. (2024). Exploring the impact of outlier variability on anomaly detection evaluation metrics. https://arxiv.org/abs/2409.15986

6. Klüttermann, S., Rao, N., & Müller, E. (2025). Multi-conceptual autoencoder ensembles for anomaly detection [Under review]. *International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*

FRACTION OF PERSONAL CONTRIBUTIONS TO THIS THESIS

Chapter 1 100%, no collaboration

Chapter 2 100%, no collaboration

Chapter 3 95% my work, supervision from Prof. Müller (5%)

Chapter 4 65% my work, detailed help from Prof. Müller (15%) and Tim Katzke (20%)

Chapter 5 65% my work. While based on the thesis of a student (Vanlal Peka), his work needed to be repeated in full. He contributed only 5% to the final version. Prof. Doebler provided detailed feedback (15%) and Prof. Müller has provided supervision 5%.

Chapter 6 45% my work. This Chapter is based on an equal contribution paper with Benedikt Böing (45%). 10% of supervision was contributed by Prof. Müller.

Chapter 7 75% my work. The initial idea, the initial proof and all experiments where my work. Chiara Balestra has cleaned up my proof and wrote the related work (20%). Prof. Müller supervised 5%.

Chapter 8 95% my work. Prof. Müller has supervised 5%.

Chapter 9 95% my work. Prof. Müller has supervised 5%.

Chapter 10 80% my work. This work is based on a thesis by Jessia Joby. However, all experiments were repeated. In the final version, she contributed only 5%. Shubham Gupta has contributed 10% and Prof. Müller has supervised 5%.

Chapter 11 50% my work. This Chapter is based on two papers. The first one is an equal contribution work with Jérôme Rutinowski (45% for both of us, and 10% for the remaining authors). I am the primary author for the second contribution (55%); however, Jérôme Rutinowski has also contributed significantly (35%). The remaining authors contributed 10% in total. My contribution to both papers averages to 50%.

Chapter 12 65% my work. Jérôme Rutinowski contributed 30%, while Prof. Müller contributed 5%.

Chapter 13 100%, no collaboration

List of Supervised Theses

Related to the Thesis

1. Tang, H. P. (2022). *Interpreting anomaly detection: Optimized preprocessing for ensemble methods* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/ping.pdf

2. Rao, N. (2022). *Improving autoencoder ensemble learning for unsupervised anomaly detection* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/nikitha.pdf

3. Grimme, B. (2023). *Entwicklung eines algorithmus für die wiedererkennung von palettenklötzen basierend auf maschinellem lernen* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/britta.pdf

4. Katzke, T. (2023). *Dean-ts: Deep ensemble anomaly detection for time series* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/tim.pdf

5. Häusler, A. (2023). *Using large language models for anomaly detection on text datasets* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/anton.pdf

6. Nguyen, P. H. (2023). *Evaluating sequential autoencoder ensemble for anomaly detection* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/phuong.pdf

7. Thiyaagu, H. (2024). *Stacking ensemble methods for anomaly detection* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/haritha.pdf

8. Nguyen, T. N. A. (2024). *Ensemble methods for outlier node detection on attributed static graphs* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/anh.pdf

9. Sadari, S. (2024). *Improving re-identification using specialized outlier detection algorithms* [Master's thesis, TU Dortmund University]

10. Peka, V. (2024). *Anomaly detection using an ensemble with simple sub-models* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/peka.pdf

11. Wickes, J. (2024). *Surveying the effects of lipschitz continuity on the robustness of anomaly detection algorithms* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/justin.pdf

12. Joby, J. E. (2024). *Understanding the impact of automated hyperparameter tuning on anomaly detection methods* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/jessia.pdf

13. Ok, M. (2024). *Enhancing anomaly detection through test-time training* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/minjae.pdf

14. Hariharan, G. D. (2024). *Anomaly detection model selection using super-auc scores* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/gautam.pdf

OTHERS

1. Barghidarian, S. (2022). *Extending isolation forest to support non-numerical data* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/sina.pdf

2. Komati, H. (2024). *Enhancing anomaly detection with deep autoencoder network using a custom semi-supervised loss* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/hepsiba.pdf

3. Mekki, L. (2024). *Understanding and mitigating training challenges for generative adversarial networks (gans)* [Master's thesis, TU Dortmund University]

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Anomaly detection is the search for rare, unusual, and exceptional events. It has many applications in diverse and critical fields, like fraud detection, predictive maintenance and scientific research. For these applications, *effective* algorithms are crucial. Next to their ability to make *accurate* predictions (Olteanu et al., 2023), we believe that such algorithms also have to be *reliable* in different situations (Yuan & Wu, 2022) and *robust* towards intentional attacks (Najari et al., 2021; S. K. Singh & Roy, 2020). Many applications need to be run in real-time (Decker et al., 2020; Gigoni et al., 2019), and thus we expect an effective anomaly detection algorithm to make predictions *fast*. Finally, anomalies often represent faults to be fixed or interesting samples to be further understood (Langone et al., 2020; Z. Li et al., 2023; Mikuni et al., 2022), and thus, an effective anomaly detection algorithm also needs to be *interpretable*.

In this thesis, we study a solution for effective anomaly detection, which we call abstract ensembles. This approach produces algorithms that are highly accurate with low runtime while being very reliable, and allows for robustness to adversarial attacks and for explaining the reason behind anomalous samples.

Abstraction is the process of simplifying complex systems by focusing on high-level concepts (Colburn & Shute, 2007; Giunchiglia & Walsh, 1992). This means an abstract description of a process is a simple but general representation of a possibly more complex system. We extend this into abstract ensembles, which combine multiple simple abstract submodels. Each of these submodels might only be able to describe a limited fraction of a complicated behavior. However, by combining many of them into an ensemble, each is

able to understand a different facet, and the ensemble can represent a more holistic understanding.

Because an abstract description is less susceptible to noise, it tends to generalize better to new data (An et al., 2023). Furthermore, an ensemble of many diverse, abstract submodels tends to have both a lower bias and a lower variance compared to non-ensemble methods (Gupta et al., 2022) and is thus able to describe complicated processes more accurately. We exploit these effects with abstract ensembles, proposing multiple highly *accurate* anomaly detection algorithms.

Other approaches towards anomaly detection aim to model the entire density distribution as, for example, a gaussian mixture model (Oluwasegun & Jung, 2023) or a normalizing flow (Kirichenko et al., 2020; Rezende & Mohamed, 2015) do. In contrast, our abstract descriptions only describe a small part of a possibly very complex process each. This means that every of the submodels we use can be very simple models, which can be trained and evaluated extremely fast. We will use this to create one of the *fastest* anomaly detection algorithms ever proposed.

Describing complex behaviors through simpler, more abstract models is a core concept in explainable machine learning research (Ribeiro et al., 2016; Soleymani et al., 2018; Y. Wang & Tuerhong, 2022), as in contrast to large neural networks, the behavior of these models can be understood by humans analyzing their weights. Furthermore, we will show here that by studying relatively large ensembles of submodels considering slightly different aspects of a process, we can exploit the structure of such an ensemble to *interpret* why a sample is considered anomalous.

Adversarial attacks (Madry et al., 2018) are a problem common in machine learning, and as anomaly detection is often applied to safety-critical fields, this is a problem that effective anomaly detection has to solve (Böing et al., 2020). As our abstract descriptions are a collection of simple submodels, which are supposed not to model every process in detail, these are often already less exposed to the risk of adversarial attacks. Furthermore, the largest problem of algorithms verifying safety against adversarial robustness is the exponential runtime of such algorithms as a function of the complexity of models. As our abstract submodels are more simple and thus relatively fast to verify, we extend such verification to ensemble methods and show that we can both provably verify the robustness of an ensemble in realistic time and also exploit the ensemble structure to make it even more *robust*.

An often understudied problem in anomaly detection is the reliability and simplicity of the training process. While not as important in supervised machine learning, anomaly

detection methods are usually trained without knowing how an anomaly might look like (Olteanu et al., 2023). This means that such an anomaly detection method has to function well every time, as there is no feedback available later. This is what we refer to as reliability. Training a simple, abstract submodel is generally easier and, thus, more reliable than training complex models. Additionally, since we combine many diverse submodels, the resulting ensemble is able to cancel out single submodel errors (Dodge, 2008). Thus, we will show that our abstract ensembles are highly *reliable* and, furthermore, that we can trade this reliability for even more accurate anomaly detection ensembles.

Thus, we believe that abstract ensembles provide an important perspective on anomaly detection, offering an effective balance between accuracy, speed, reliability, robustness, and interpretability. In this thesis, we make the following key contributions:

- We develop a novel anomaly detection algorithm based on abstract ensembles and show its competitive performance against state-of-the-art methods.
- We verify the reliability and stability of such an abstract ensemble method and show that they outperform standard methods in terms of consistency and adaptability.
- We improve this method by using even simpler submodels and show that this results in one of the fastest anomaly detection algorithms proposed so far.
- We propose an alternative, more complex training procedure that, under certain conditions, can drastically outperform state-of-the-art competitors even further.
- We show that such an abstract ensemble method is more resistant to adversarial attacks and propose an efficient method for verifying their robustness.
- We demonstrate how abstract ensembles provide interpretable anomaly explanations, leveraging the structure of diverse submodels to highlight anomalous patterns.
- We further analyze how abstract ensemble approaches can be applied to non-anomaly detection tasks and show that approaches based on abstract ensembles also lead to effective re-identification models.

Chapter Overview

To provide a comprehensive understanding of abstract ensembles and their applications, this thesis is structured into the following chapters:

Chapter	3	4	5	6	7	8	9	10	11	12
Accuracy		✓	✓			✓	✓			✓
Speed			✓	(✓)	✓				✓	✓
Robustness				✓						(✓)
Interpretability					✓		(✓)		✓	
Reliability		✓	✓					✓	(✓)	

Table 1.1: Overview of the chapters in which we discuss different benefits provided by abstract ensembles.

- Chapter 2 provides related work on anomaly detection, ensemble learning, and abstraction.
- Chapter 3 represents a study searching for best practices in ensemble methods for anomaly detection, which helps to guide the way we propose new methods in later chapters.
- Chapter 4 introduces the theoretical background of abstract ensemble learning and our own abstract ensemble method. This method, DEAN, is benchmarked against state-of-the-art competitors, and its reliability, including its low dependency on hyperparameters, is demonstrated.
- Chapter 5 introduces SEAN, which uses even simpler submodels compared to DEAN for extremely fast anomaly detection.
- Chapter 6 explores adversarial robustness, including a verification framework for ensuring ensemble reliability and a method to increase the robustness of such an ensemble further.
- Chapter 7 discusses interpretability in anomaly detection and demonstrates how abstract ensembles can provide meaningful explanations.
- Chapter 8 introduces a further paradigm for anomaly detection, test-time training, which trades the reliability of our abstract ensemble method against drastically higher performance under certain circumstances.
- Chapter 9 discusses a limitation of the previous chapter and generalizes it into a more fundamental property of anomaly detection rarely discussed.

- Chapter 10 extends the benchmarking of our algorithms, by also considering the hyperparameters of our competitors.
- Chapter 11 considers heterogeneous abstract ensembles for re-identification to show that abstract ensembles are a valid approach not only in anomaly detection.
- Chapter 12 extends the previous chapter to homogenous DEAN-like ensembles and shows that such abstract ensembles are a more effective solution to re-identification than commonly used complex neural networks.
- Chapter 13 concludes the thesis with a discussion of key findings, limitations, and future research directions.

We further list which chapter considers which aspect of abstract ensembles in Table 1.1.

Part I

State of the art

CHAPTER 2

LITERATURE REVIEW

While topic-specific relevant literature will be discussed in the related chapters, this chapter reviews those topics that are generally relevant to understanding the following chapters.

2.1 Anomaly Detection

Hawkins (1980) defines the concept of an anomaly (or an outlier) as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.". Anomaly detection is the search for such strange, unusual, and exceptional observations. This has many different applications, classical ones like fraud detection (Hilal et al., 2022; Levin, 2019; T. Wu & Wang, 2021), machine fault detection (Brito et al., 2021; Dong et al., 2017; Miljković, 2011; Radaideh et al., 2022) and predictive maintenance (Carvalho et al., 2019; Çınar et al., 2020; Dalzochio et al., 2020; Jovicic et al., 2023; Shukla et al., 2020; W. Zhang et al., 2019; Zhu et al., 2024), but more recently also healthcare (Agosta et al., 2013; Archana et al., 2015; C. M. I. P. M. Castro, 2014; Fernando et al., 2022; J. Zhang et al., 2020), computer safety (Salem, 2018; S. K. Singh & Roy, 2020; W. Xu et al., 2022) and even scientific discovery (Craig et al., 2024; Lochner & Bassett, 2021; Mikuni et al., 2022; Yamada, 2019).

Matching a large number of applications, many methods also have been proposed. These can be classified according to different attributes. The likely most important of these differences is the amount and type of available labeled data. In the unsupervised case, no information about any labels is available, and thus, unsupervised anomaly detection meth-

ods search for rare and unusual events when compared to the remaining samples. This can be a daunting task since more uncommon samples sometimes appear indifferent to anomalies, but it is very easy to apply, as no labels are required. In the opposite case, when given access to labeled data, supervised anomaly detection (Yamanaka et al., 2019) is more similar to the classification of imbalanced data. However, labeling many possibly very rare anomalies can be extremely expensive. To solve this, semi-supervised methods (Ruff et al., 2019) only use a small subset of labeled data to enhance an unsupervised method. This can be quite effective (Ruff et al., 2021), but requires at least one of each type of anomaly to be known before as it will not generalize well to new types of anomalies.

Because of this, we consider weakly supervised anomaly detection in this thesis. Here, some normal samples are labeled, but no anomalies are known. Thus, the task of anomaly detection means understanding the generating process and finding samples deviating from them. Sadly, the nomenclature in anomaly detection research is inconsistent. Confusingly, the weakly supervised case is also sometimes referred to as semi-supervised or unsupervised (Olteanu et al., 2023). A more accurate name that is also sometimes used is one-class classification (S. S. Khan & Madden, 2010), but we will refer to this as weakly supervised anomaly detection to keep consistency with the alternative cases.

The weakly supervised case is interesting since labeling normal samples is generally much cheaper than abnormal ones and can often even be inferred from past behavior ("Since this machine worked for the last weeks, we can consider its measured samples as normal"). Additionally, given a large dataset of normal samples, it is straightforward to define what we are searching for, as we simply have to search for points that could not appear in this dataset. Moreover, weakly supervised methods tend to generalize well to new types of anomalies as they do not use any information about anomalous data. Finally, weakly supervised anomaly detection is also easier to evaluate than the unsupervised case. Training a model on an unsupervised dataset results in a performance that depends on the fraction of anomalies in the dataset used. This forces benchmarks to choose between datasets that realistically have extremely rare anomalies and highly uncertain performances because of the low number of anomalies or, instead, relatively common anomalies and measurably lower performances. Instead, in the weakly supervised case, we consider the limit of arbitrarily rare anomalies in the training set and are still able to use as many anomalies as we want during evaluation. This also implies that most unsupervised anomaly detection algorithms can also be used in the weakly supervised case and vice versa.

Both of these approaches require minimal labeling and, thus, are easy to use. Because of this, many methods have been proposed in this setting. These can be further classified

based on their approach to anomaly detection: A common idea is to model the density distribution of the training set and consider samples in a low-density region as more anomalous. This is directly done by algorithms like kernel density estimation (kde) (Rosenblatt, 1956), gaussian mixture models (gmm) (Oluwasegun & Jung, 2023) and normalizing flows (nf) (Rezende & Mohamed, 2015). Other methods, like knn (Gu et al., 2019) and lof (Breunig et al., 2000), are based on the distance of a new sample to existing ones or try to isolate more rare samples like isolation forests (ifor) (Hariri et al., 2018; F. T. Liu et al., 2008) and deep isolation forests (dif) (H. Xu et al., 2023). Still, it can be shown that both of these approaches are also effectively modeling density distributions with slight restrictions (Buschjäger et al., 2022; Ramaswamy et al., 2000). Other approaches try to solve surrogate tasks on normal data points and identify anomalies as those samples where this surrogate task fails. This includes those models that try to compress and reconstruct samples, like autoencoder (Sakurada & Yairi, 2014) and pca (Callegari et al., 2011). Also, DeepSVDD (Ruff et al., 2018) and NeutralAD (Qiu, Pfrommer, et al., 2022) follow this paradigm, learning a constant representation and data transformations, respectively. Similarly generative models, like anogan (Mattia et al., 2021) and diffusion time estimation (dte) (Livernoche et al., 2024) can be exploited to find anomalies. Finally, methods like dagmm (Zong et al., 2018) and variational autoencoder (Kingma & Welling, 2014) combine multiple of the previous approaches.

However, what most of these quite diverse approaches have in common is that they tend to produce anomaly scores instead of clear predictions. The higher such a score is, the more likely a corresponding sample is considered to be anomalous. To convert this into a label, a threshold needs to be chosen. Samples above this threshold are considered anomalous, and samples below them are considered normal. The benefit of such an anomaly score is that the threshold can be chosen in a way that fits the application best. For example, a lower threshold might be beneficial when anomalies represent possible life-threatening illnesses (since false positive diagnoses are preferable to false negatives and missing a medical problem), while we would choose the opposite for a spam filter (since missing an important email is usually worse than reading an additional spam mail).

Most metrics, like a simple accuracy or the more fitting F1-Score (Goutte & Gaussier, 2005), require first choosing such thresholds before evaluating a model. Instead, we prefer evaluation metrics that can directly evaluate anomaly scores; otherwise, the performance would depend on the threshold choice. We primarily use here the area under the ROC curve (ROC-AUC) (Hanley & Mcneil, 1982). It is defined as the probability of a random abnormal sample having a higher anomaly score than a random normal one. This means that a model

with no predictive capabilities has a ROC-AUC score of 0.5. A perfect model reaches 1.0, and the worst possible model, the one that considers every anomaly more normal than every normal sample, reaches an ROC-AUC of 0.0. Effectively, ROC-AUC scores below 0.5 represent useless models, and in practice, even models with an ROC-AUC score of ≈ 0.7 might not be very useful. One crucial benefit of the ROC-AUC is that its calculation is invariant under the fraction of anomalies in the test dataset (Ok et al., 2024). This makes it more useful than, for example, the AUC-PR (Boyd et al., 2013), as it allows us to have unrealistically many anomalies in the test set to reduce the uncertainty of the evaluation metric.

2.2 Ensemble Methods

Ensembles (Schapire, 1990) combine the outputs of multiple machine-learning models to produce a higher-quality output. Ensembles have originally been studied for supervised methods, where ideas like bagging (Breiman, 1996), boosting (Schapire et al., 1999), and stacking (Mienye & Sun, 2022) allow even weak learners to perform well and generally outperform non-ensemble methods. Most ensemble approaches work by training and evaluating submodels and combining these based on their performance. For unsupervised and weakly supervised tasks, as studied here, this is not possible, as we are not able to evaluate individual submodels. To solve this, some approaches use pseudo-labels to build boosting ensembles (Sarvari et al., 2021; H. Ye et al., 2023). However, this relies on an effective strategy to create inaccurate labels.

Instead, another approach is based on bagging multiple predictions together and combining their model outputs with a combination function. We study and compare such combination functions in Chapter 3. This approach works even without labels since it is less likely that multiple diverse models make the same mistake (Dodge, 2008). In terms of the bias-variance tradeoff (Bishop, 2007), such an ensemble is less restricted in the type of functions it can learn (lower bias) and also smoothes out individual submodel errors (lower variance) (Gupta et al., 2022). For this effect to work well, we require different submodel predictions to not be highly correlated.

In anomaly detection, two approaches exist to reach such a set of submodels with high variance: Heterogeneous ensembles combine different anomaly detection methods. This generally means that they combine high-variance submodels. However, this also limits the size of the ensemble by the number of implemented anomaly detection approaches, and it has a significantly higher runtime than the non-ensemble approaches it uses as submodels. As we will show in Chapter 3, it is also nontrivial to combine such predictions into a better

model.

Instead, we mostly prefer to study homogeneous ensembles. These combine similar models with usually just different random initializations. While there is often less variance between submodels, this is mitigated by the larger number of models that can be combined. Additionally, combining similar models is usually easier.

Some examples of homogeneous ensembles have already been proposed. An isolation forest (F. T. Liu et al., 2008) combines many semi-random decision trees. They also prove that an ensemble does not need to be slower compared to an individual model. We will see that some of the findings of this thesis also generalize to isolation forests (See Chapter 7). Still, the fact that isolation forests use simple decision trees might limit the complexity of functions they can learn, and the fact that they model density distribution (Buschjäger et al., 2022) limits how well they handle more high dimensional tasks (Bellman, 1966).

Thus, a deep learning alternative might be preferable. We want to highlight here RandNet (J. Chen et al., 2017), which combines multiple autoencoder models into a better one. However, Randnet also highlights the difficulties of such a heterogenous ensemble: Because they use a complicated choice of submodels, the performance improvement through the ensemble is limited, even after using a dropout (Srivastava et al., 2014) like modification of their neural networks (a random cut to the layer connections) to increase the variance of their submodels. Additionally, this results in a relatively slow ensemble, as even when using a low number of just 10 submodels, they still need to train 10 large neural networks.

Instead, we suggest using abstract ensembles. The key difference is that similar to isolation forests, we don't expect each submodel to be an effective anomaly detection model. This allows us to drastically simplify these submodels, which in turn also increases their prediction variance, resulting in a more effective and robust ensemble. And since simple submodels are also much easier to train, these ensembles are also often faster even when training many more models.

2.3 Abstraction

Abstraction is the process of simplifying complex systems by focusing on high-level concepts (Colburn & Shute, 2007; Giunchiglia & Walsh, 1992). It is widely studied both outside of machine learning and applied to different machine learning tasks.

For example, procedural abstraction is used in software engineering to hide complicated behavior in functions and classes, which allows programmers to focus only on the interaction between such objects (Ward & Rd, 1995). More fundamentally, the ability to build

abstract theories to understand complicated behaviors is a crucial skill with many applications (Yee, 2019): In mathematics (Mitchelmore & White, 2012), we define abstract concepts that generalize the behavior of complex objects into their most important parts (e.g., both the Euclidean and the Levenshtein distance are both mathematical objects called metrics, and we can prove properties of both at the same time). Similarly, physicists (Vaz, 2024) postulate that the entire universe can be described by a collection of various abstract fields. But also outside of fundamental science, an abstract understanding is used when fables and allegories are used to convey complex ethical lessons or when we group together complex political ideologies into a simple left-right separation.

As this seems to be a fundamental part of how we think, it is perhaps not surprising that abstraction is also studied in many facets of machine learning (Kozma et al., 2018). Some approaches directly search for simplified models to approximate complex ones. Such a model is commonly referred to as a surrogate model (Koziel et al., 2011). For example, LIME (Ribeiro et al., 2016) uses simple interpretable surrogates to explain the behavior of a more complex model locally. Similarly, pruning (Blalock et al., 2020) and knowledge distillation (C. Yang et al., 2024) create a simpler secondary model that should mimic a more complicated one. Here, abstraction results in either more interpretability or a reduced model size and a reduction in overfitting. Additionally, the resulting increase in generalizability is used by transfer learning methods (Zhuang et al., 2020) to further specify the task a model is trained on.

In another approach, some methods replace general fully connected neural networks with smaller, more abstract layer types by assuming certain properties of data. For example, convolutions (LeCun et al., 2015) assume translation invariance and that pixels close to each other matter more towards image classification than pixels far away from each other. Similarly, recurrent neural networks (Rumelhart et al., 1986) process sequences in an ordered manner, where past information influences future predictions, though they struggle with long-range dependencies. More recently, transformers (Vaswani et al., 2017) use self-attention to dynamically learn dependencies across the entire input, overcoming these limitations.

Finally, some methods create a simplified data representation (Soleymani et al., 2018). This kind of abstraction assumes that a complicated behavior with many features can actually be described by lower dimensional representation. Examples here include regularization (G. Li et al., 2021) and lasso regression (Tibshirani, 1996), searching for the most important subset of features. When this is not effective, dimensionality reduction methods like PCA (Callegari et al., 2011) or Autoencoder (Fournier & Aloise, 2019; Maggipinto et al.,

2018) can create a dependent set of features.

Generally, we see four benefits of abstraction:

1. Abstract methods have fewer parameters and thus are less susceptible to overfitting effects and **generalize** better to new situations (An et al., 2023). In the most extreme case, a physical law can describe both the motion of cells and planets.

2. Fewer parameters also make a model more **explainable** (Y. Zhang, 2024).

3. Similarly, smaller abstract models also make **faster predictions** (Ashok et al., 2020; Flusser & Somol, 2022).

4. Finally, abstraction tends to allow a more **modular** view of a problem, as every part is only built to solve a small part of a more complicated problem (Hudson & Manning, 2019; Modi et al., 2023; M. Müller et al., 2018; H. Sun & Guyon, 2023)

Anomaly detection, which we primarily study in this thesis, is also often done in an abstract way, where we train models on simplified surrogate tasks. Here, a failure of this abstract model in its surrogate task usually indicates an anomaly. For example, autoencoder, when used for anomaly detection, are tasked to reconstruct input samples and recognize anomalies when failing to reconstruct. Similarly, DeepSVDD is tasked to output a constant value, and deviations from this task represent anomalies.

We extend this idea to abstract ensembles. The core difference is that instead of one simpler but still quite complicated model, we can better represent a complicated system through many much simpler, more abstract submodels. By combining these submodels, a superior anomaly detection algorithm emerges, surpassing each submodel alone. We will show that such an approach is not only preferable in its performance but can also result in a more explainable and faster model and that we can exploit its modular structure.

CHAPTER 3

BENCHMARKING ENSEMBLE METHODS

This chapter follows: Klüttermann, S., & Müller, E. (2023). Evaluating and comparing heterogeneous ensemble methods for unsupervised anomaly detection. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN54540.2023.10191405

3.1 Introduction

Ensembles are a powerful tool for weakly supervised anomaly detection, that can combine possibly quite different model predictions into a better one. Consider for example the decision boundaries in Figure 3.1: While both submodels have their own weaknesses, the ensemble improves upon both.

However, when using ensembles instead of individual models, this introduces additional hyperparameters like the combination function merging individual submodel predictions or the choice of models to include in the ensemble. As we will discuss in more detail later (See Chapter 10), hyperparameters are a significant problem for unsupervised or weakly supervised methods, as it is not possible to optimize them without supervised feedback.

So, before proposing our own ensembles, we will first study the impact of these ensemble hyperparameters to guide our algorithms. While we consider homogenous ensembles (ensembles that are constructed from very similar submodels) for most of the rest of this thesis, we will study here heterogeneous ensembles (ensembles that are constructed from

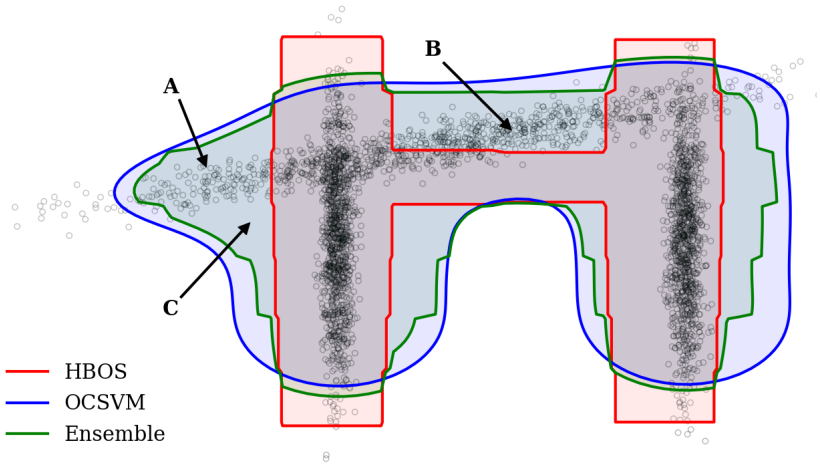


Figure 3.1: Regions shown which are considered normal by two algorithms, OCSVM and HBOS, as well as of an ensemble of both. This ensemble considers the average score of both models and can create a more accurate representation of the underlying distribution: While HBOS fails to capture the distribution at point A and clearly makes a mistake at point B, it also captures the vertical parts of the distribution much better (see point C). The ensemble handles points A and B right and captures the distribution at C better than the OCSVM. Still, the combination is not perfect; finding the right function to build an ensemble is a complicated task.

different anomaly detection methods), as most combination functions are proposed for the heterogenous case and the less simple combination task shows more significant differences.

Many combination functions for such ensembles have been proposed in the literature, ranging from simple approaches like the average of submodel predictions to complicated ones using pseudo labels (Schubert et al., 2012) or one based on item response theory (Z. Chen & Ahn, 2019).

Still, no unbiased comparison of these methods exists. This might be because there are exponentially many model combinations to be considered. These include the choice of the dataset, the choice of combination function, and the set of anomaly detection algorithms. To solve this, we create the biggest known comparison to date, evaluating more than 6 billion ensembles, which, to the best of our knowledge, is multiple orders of magnitude more than every other study before.

We are also the first to study the effect different normalization functions have on the resulting combination, and we use the high number of ensembles we study to also extract information about the best submodels to use.

By doing so, we provide a complete guide for creating effective weakly supervised anomaly

detection ensembles. This knowledge will then be used in later chapters to propose our own ensemble methods.

3.2 Related Work

Other papers compare different combination functions for anomaly detection ensembles.

Each of the papers that introduce the combination functions of the next Section also compares them to other methods (C. Aggarwal, 2012; Z. Chen & Ahn, 2019; Schubert et al., 2012). Still, their results tend to contradict each other, limiting the trust we can have in their conclusions. Also, their evaluation is often limited to a low number of ensembles, further limiting how well we can rely on these comparisons. One exception to this is the paper introducing the IRT combination function (Z. Chen & Ahn, 2019), which evaluates combination functions on 10000 different ensembles and inspired our evaluation. Still, even compared to this high number of ensembles, we are creating 10000 times more ensembles while using more datasets and individual submodels.

Another notable evaluation is given by (Chiang et al., 2016), which compares different methods in an unbiased way, but their limited evaluation makes this paper also less useful in our opinion.

3.3 Combination functions

In this Section, we will briefly review different ensemble combination methods.

3.3.1 Simple functions

C. Aggarwal (2012) suggests using the average of multiple model predictions or the maximum model prediction. Similarly, J. Chen et al. (2017) uses a median of the model predictions.

In addition to these, we also test the minimum model prediction

$$ens(x) = min_i(model_i(x))$$

and both a quadratic, cubic and quartic mean.

$$ens(x)^a = mean_i(model_i(x)^a),\ a = 2,3,4$$

C. Aggarwal (2012) also suggests combining average and maximum model prediction into an average of maxima (AoM) or maximum of average (MoA). But using those requires

choosing many additional options: Which combination to use and which submodel to assign to which of how many groups. Because of this, we leave these functions for a dedicated study.

3.3.2 Stacking

The three most common ensembling methods for supervised tasks are bagging, boosting and stacking (Odegua, 2019). In the unsupervised and weakly supervised settings, we mostly look at bagging, as it is much easier to do compared to boosting (C. Aggarwal, 2012), which requires labels. But we do not see a reason prohibiting stacking: We can understand the individual models as learning a one-dimensional feature representation each in which anomalies are more easily visible. And on this representation, we can train another anomaly detection algorithm.

Still, stacking has not been studied well in the literature, except for supervised anomaly detection (Sandim, 2017).

To understand why there is still not much research into stacking, we test a few common anomaly detection methods as combination functions. For this, we train our stacking model on 20% of our normal training data. We refer to this dataset as our normalization samples, as we will also use the same samples to calculate normalization constants in the next section. We train 5 different KNN (Gu et al., 2019) models with various values of k (1,3,5,10,100), and an isolation forest (F. T. Liu et al., 2008). We also use the (the inverse) probability density function obtained by modeling the distribution as a single multi-variate Gaussian distribution. Further tests were later done by a thesis student (Thiyaagu, 2024).

3.3.3 Combination through clustering

Similar to the stacking approach, we try to model the resulting anomaly detection representation space here. But instead of training an anomaly detection algorithm, we cluster these points. This means when we have multiple groups of anomalies (one algorithm only finds one kind of anomaly, while another one only finds a different kind of anomaly), we assume that points in the center of each of these clusters are normal, while points with a high distance to them are not. To evaluate this, we use affinity propagation (Frey & Dueck, 2007) to calculate the clusters. For computational reasons, we restrict ourselves here to using z-score normalization.

3.3.4 Threshold combinations

A threshold combination of submodels (C. Aggarwal, 2012) is defined as

$$ens_{thresh}(x) = mean_i(relu(model_i(x) - t) + t)$$

where $relu(x) = \frac{1}{2} \cdot (x + \|x\|)$. So, high anomaly scores are not changed, while low anomaly scores are cut off and brought to a constant level. Anomaly detection usually only cares about high ranges of scores, which represent anomalies. This means that the combination can be perturbed by small fluctuations in how normal a sample is considered, which can be fixed by threshold ensembles. We implement this here only with the original z-score normalization, as this allows us to understand the Threshold value t as Z scores by assuming the distribution of anomaly scores to be Gaussian. We test here 5 different values for $t = -2, -1, 0, 1, 2$.

3.3.5 Greedy ensembles

A greedy ensemble (Schubert et al., 2012) is very similar to an approach like boosting for unsupervised anomaly detection. This would usually not be possible without true labels, so the approach here is to construct an approximative label vector from the submodels themself. This works, as we can be reasonably sure that the most abnormal samples (we assume here 5) found by each algorithm are true anomalies. Using this vector, we can try to average only a subset of models. We choose those that maximize a weighted correlation to our constructed vector in a greedy manner. We restrict ourselves to a 01 normalization for computational reasons.

3.3.6 Item response theory

Finally, Z. Chen and Ahn (2019) suggests using IRT, a method commonly used in psychology to extract hidden variables, by treating the optimal anomaly score as a hidden truth.

3.4 Normalization functions

We compare 5 different common normalization methods. For each of these methods, we use a subset of our training samples containing 20% of the points to calculate the individual normalization constants. This means, that when we demand all our samples to be in the range [0,1], all of our (normal) normalization samples are in this range, but for our training

and test samples, this is not guaranteed. Still, also most (normal) test samples are in this range.

We think this approach best matches the task of weakly supervised anomaly detection. While most normal data is normalized, no such restriction is made for abnormal samples.

The only exception is the IRT ensemble because the calculation requires every score to be between 0 and 1. Here, we normalize our samples using the full dataset.

3.4.1 01 normalization

By applying 01 normalization, we demand each model output to be between 0 and 1.

$$x_i^{01} = \frac{x_i - min_j(x_j)}{max_j(x_j) - min_j(x_j)}$$

3.4.2 Z-score normalization

When applying z-score normalization, we want to fix the mean and the standard deviation of each model's output to be 0 and 1.

$$x_i^{Z} = \frac{x_i - mean_j(x_j)}{std_j(x_j)}$$

3.4.3 Simple normalization

Using no normalization would make little sense for most models, as their scale is often quite different. Instead, we simply divide each value by the mean of all predictions. We use this normalization mostly to test how important careful normalization is.

$$x_i^{simple} = \frac{x_i}{mean_j(x_j)}$$

3.4.4 Clipped normalization

Finally, we suggest an extension to z-score normalization, in which values are limited to a given absolute value c.

$$x_i^{clipped} = min(max(x^Z, -c), c)$$

This means that the influence of a single model is limited, as it can no longer have an arbitrarily high contribution. We test this normalization once with $c = 1$ and with $c = 2$. Assuming a Gaussian distribution, this should alter $\approx 32\%$ and $\approx 4.6\%$ of samples, respectively.

3.5 Experiments

3.5.1 Experimental setup

We design our experiments so that we can consider the highest number of ensembles available, by covering as many datasets and submodels as possible.

On every of the 182 datasets currently contained in the yano library (Klüttermann, 2023), we apply 21 models. For this we use knn (Gu et al., 2019), lof (Breunig et al., 2000), kde (Rosenblatt, 1956), isolation forest (F. T. Liu et al., 2008), abod (Kriegel et al., 2008), anogan (Mattia et al., 2021), deepsvdd (Ruff et al., 2018), ocsvm(Bounsiar & Madden, 2014), autoencoder (Sakurada & Yairi, 2014), a variational autoencoder (Kingma & Welling, 2014), pca (Callegari et al., 2011), Inne (Bandaragoda et al., 2018), Sod (Kriegel et al., 2009), HBos (Goldstein & Dengel, 2012), Cblof (Z. He et al., 2003), Loda (Pevný, 2016), COF (J. Tang et al., 2002), Copod (Z. Li et al., 2020), Ecod (Z. Li et al., 2022) and Lmdd (Arning et al., 1996).

All models are trained using the pyod library (Y. Zhao et al., 2019) with their originally suggested hyperparameters. Using these results, we take a look at every possible combination of models to build our ensembles. We only consider ensembles that contain at least five submodels. Some models are also incompatible with some datasets, producing faulty or erroneous results; we simply ignore those combinations and build ensembles from the remaining submodels.

In total, we look at 124.053.401 different ensemble-building tasks. We apply every of our 50 different pairs of combination (Section 3.3) and normalization functions (Section 3.4) to each of them. This means, in total, we look at more than 6.2 billion different ensembles to produce our conclusions.

While we use additional datasets and further methods in later chapters of this thesis, we do not include them in this chapter, as rerunning them would be expensive and likely still result in similar results.

3.5.2 Evaluating ensembles

A single anomaly detector is usually evaluated using a ROC-AUC score (Bradley, 1997). It is defined as the probability of random anomaly samples being assigned a higher anomaly score than a random normal one.

If we want to extend this to comparison ensembles, we can look at its rank-1 accuracy (Z. Chen & Ahn, 2019): In how many cases does a given ensemble produce the highest possible AUC score?

However, we think this is not the best choice, as this ignores the magnitude of change and introduces an unrealistic scenario. Most of the time, we only care that an ensemble reliably performs well and not that no other ensemble performs slightly better. Furthermore, as the stated rank-1 accuracy depends on the number of combination functions we consider, which complicates parallelizing our calculations, we evaluate our combination functions differently.

For this, we propose 10 simple metrics, which not only allow us to find the best combination method but also to learn more about how ensembles work for unsupervised and weakly supervised tasks.

The first metric that we want to consider, is the reliability of the ensemble. How often is using an ensemble a good choice, compared to a random model.

$$AUC(ens(x)) \geq mean_i(AUC(model_i(x)))$$

Similarly, we also state the probability that the ensemble improves on the best and the worst individual model it is built from (*metric 2-4*).

But reliability alone is not enough. We also care about the size of the improvement. For this, we can directly look at the average ROC-AUC score (*metric 1*). But this value stays fairly constant around 0.75, as we have some datasets on which models commonly reach a good AUC score of ≈ 1.0 and some datasets where they only reach a low score ≈ 0.5. Because of this, it is hard to interpret any differences. So, we additionally also use the same comparisons as before and give the average difference in ROC-AUC between the ensemble and the worst/average/best submodel (*metric 5-7*).

Still, this weights badly-performing models differently from well-performing models. This is because improving an AUC score from 0.8 to 0.9 compared to 0.98 to 0.99, has a much bigger effect. Even though, in both cases, the probability that a random normal sample is considered more anomalous is halved. Because of this, we define a new metric to capture these probability changes:

$$\Delta Err = log(\frac{1 - AUC_1}{1 - AUC_2}) = log(1 - AUC_1) - log(1 - AUC_2)$$

Here, a model halving the error rate always has a fixed $\Delta Err = log(2) \approx 0.3$. We also calculate this error rate compared to the performance of the worst/average/best submodel (*metric 8-10*).

3.5.3 Comparison of combination functions

We state these 10 metrics for each of our 50 ensemble procedures in Tables 3.1, 3.2, 3.3, 3.4, 3.5 and 3.6. Because of formatting limitations, we had to split our results here into six tables. Please see the paper this chapter is based on for a singular table.

We achieve the best ensembles with a simple maximum combination function normalized with a 01 normalization. Overall simple functions perform quite well, many of them outperforming each of the more complicated methods.

It is the case that most ensembles contain a submodel that performs better than the combined ensemble, showing that heterogenous ensembles for unsupervised and weakly supervised anomaly detection work substantially differently compared to supervised tasks.

At least ensembles outperforming an average model is fairly likely when using a good combination function. Still, even this is not guaranteed. There also is even a small chance, that the ensemble performs worse than each individual model. However, using the right methods only happens for less than 1 out of every 500 different ensemble, and thus can likely be safely ignored.

From the simple methods we evaluate, the maximum seems to be clearly the best. But interestingly this depends on the normalization applied; when using a clipped normalization it performs worse than every other model (by most metrics). We think this happens because removing extreme values also removes the reason for making the maximum efficient: Only a single model has to consider a sample anomalous, but when $\approx 16\%$ of samples are equally as anomalous, this loses meaning.

Because of this dependency on normalization effects, we see the maximum combination function as high risk and high reward. While it can produce the best ensembles according to 8 out of 10 different metrics, it might also simply not perform well at all.

In the case of zero-labeled anomalies and thus no feedback, a simple mean might be more reliable. As long as a common normalization is used, it does not matter much which one is chosen. And removing very extreme values, we can create the model with the highest probability of outperforming the average performance. Still, the average ROC-AUC performance is about 1% worse than for the best ensemble.

Higher powers of the mean don't seem to serve many purposes compared to a simple mean. Also using the minimum of all models might at most be useful for very specialized tasks.

On the other hand, the threshold ensemble can outperform a simple mean very slightly. However, since the improvement is minor and this ensemble also interacts with a clipped normalization, it is probably not possible to increase it much further. The best possible

Table 3.1: Anomaly detection ensemble comparison (Metric 1-4, part 1)

Methods	Mean AUC	Bigger Than		
		Minimum	***Mean***	***Maximum***
maximum (01)	**0.766**	**99.837**%	75.39%	4.445%
maximum (z)	0.758	99.248%	72.99%	3.258%
maximum (simple)	0.723	96.634%	64.153%	4.05%
maximum (clipped 1)	0.66	91.431%	23.552%	0.282%
maximum (clipped 2)	0.712	95.002%	53.596%	0.739%
mean (01)	0.754	96.794%	84.015%	2.64%
mean (z)	0.756	96.796%	84.857%	2.633%
mean (simple)	0.734	96.801%	69.156%	3.767%
mean (clipped 1)	0.747	96.8%	82.687%	1.308%
mean (clipped 2)	0.754	96.794%	**85.903**%	1.576%
median (01)	0.752	96.708%	82.115%	1.742%
median (z)	0.755	96.764%	84.038%	1.803%
median (simple)	0.744	96.659%	74.613%	2.805%
median (clipped 1)	0.743	96.761%	78.658%	0.475%
median (clipped 2)	0.753	96.759%	83.664%	1.065%
minimum (01)	0.673	95.942%	36.93%	0.902%
minimum (z)	0.671	96.439%	35.013%	0.682%
minimum (simple)	0.742	97.023%	71.487%	4.143%
minimum (clipped 1)	0.667	96.466%	29.391%	0.643%
minimum (clipped 2)	0.672	96.431%	34.76%	0.671%
l2mean (01)	0.753	97.075%	77.929%	3.079%
l2mean (z)	0.729	95.947%	52.75%	4.131%
l2mean (simple)	0.735	97.753%	65.479%	2.892%
l2mean (clipped 1)	0.703	93.547%	37.37%	2.09%
l2mean (clipped 2)	0.731	94.967%	53.955%	2.933%
l3mean (01)	0.751	97.333%	76.353%	3.242%
l3mean (z)	0.728	96.185%	52.526%	4.219%
l3mean (simple)	0.734	97.618%	65.393%	3.095%
l3mean (clipped 1)	0.706	93.834%	39.472%	2.095%
l3mean (clipped 2)	0.734	95.336%	56.223%	3.013%
l4mean (01)	0.75	97.537%	75.127%	3.361%
l4mean (z)	0.728	96.396%	52.426%	4.203%
l4mean (simple)	0.733	97.599%	65.068%	3.15%
l4mean (clipped 1)	0.708	94.011%	40.809%	2.117%
l4mean (clipped 2)	0.735	95.912%	57.487%	3.016%

Table 3.2: Anomaly detection ensemble comparison (Metric 1-4, part 2)

Methods	Mean AUC	Bigger Than		
		Minimum	***Mean***	***Maximum***
Threshold ($t = -1$)	0.757	96.8%	85.036%	2.655%
Threshold ($t = -2$)	0.756	96.796%	84.934%	2.652%
Threshold ($t = 0$)	0.756	96.734%	81.793%	3.009%
Threshold ($t = 1$)	0.746	96.797%	70.396%	2.296%
Threshold ($t = 2$)	0.719	97.651%	47.855%	3.156%
knn ($k = 1$)	0.757	97.326%	68.474%	4.434%
knn ($k = 3$)	0.757	97.532%	68.761%	4.434%
knn ($k = 5$)	0.756	97.664%	69.132%	4.074%
knn ($k = 10$)	0.752	97.068%	68.905%	3.22%
knn ($k = 100$)	0.664	78.755%	44.347%	3.768%
IFor	0.754	97.283%	69.757%	2.502%
Gaussian	0.754	97.807%	66.289%	**6.148**%
Clustered	0.752	96.74%	76.265%	3.991%
Greedy	0.751	96.73%	83.008%	2.167%
IRT	0.731	96.511%	70.052%	2.624%

threshold seems to be $t = -1$, where the $\approx 16\%$ most normal samples are modified (assuming a Gaussian distribution), contrary to the original paper (C. Aggarwal, 2012), which suggests $t = 0$ (affecting 50% of samples).

Stacking could be quite promising. For the knn, a low value of k is the best, performing better than each of the other approaches, including the isolation forest, a Gaussian fit, or even the clustering approach (in most metrics). It is able to outperform each simple mean, except in the likelihood of outperforming the mean case, nor in the Δerr metrics. This implies that it performs best in cases where the anomaly detection task is complicated and the more complicated combination function can be utilized well. We see these as specialized high-risk high-reward methods. These models require the right situation (for example many normalization samples for the knn), but then can perform very well. And considering that we only looked at four simple approaches for stacking, we think there might be even better-dedicated algorithms for our task.

The Gaussian fit deserves special attention, as it has by far the highest probability of outperforming every single submodel and thus might be able to benefit from the ensemble similar to how supervised algorithms work. Combining multiple definitions of anomalies and recognizing which parts of which definition is helpful. But before this is the case, we still have far to go, as the Gaussian fit is not very reliable, assumes a specific distribution of

Table 3.3: Anomaly detection ensemble comparison (Metric 1,5-7, part 1)

Methods	Mean AUC	ΔAUC to		
		Minimum	***Mean***	***Maximum***
maximum (01)	**0.766**	**0.28**	**0.036**	**-0.081**
maximum (z)	0.758	0.271	0.027	−0.09
maximum (simple)	0.723	0.237	−0.007	−0.124
maximum (clipped 1)	0.66	0.174	−0.07	−0.188
maximum (clipped 2)	0.712	0.226	−0.018	−0.136
mean (01)	0.754	0.267	0.023	−0.094
mean (z)	0.756	0.27	0.026	−0.092
mean (simple)	0.734	0.248	0.004	−0.114
mean (clipped 1)	0.747	0.26	0.016	−0.101
mean (clipped 2)	0.754	0.267	0.023	−0.094
median (01)	0.752	0.265	0.021	−0.096
median (z)	0.755	0.268	0.024	−0.093
median (simple)	0.744	0.258	0.014	−0.104
median (clipped 1)	0.743	0.257	0.013	−0.104
median (clipped 2)	0.753	0.267	0.023	−0.095
minimum (01)	0.673	0.187	−0.057	−0.174
minimum (z)	0.671	0.185	−0.059	−0.177
minimum (simple)	0.742	0.255	0.011	−0.106
minimum (clipped 1)	0.667	0.181	−0.063	−0.181
minimum (clipped 2)	0.672	0.186	−0.058	−0.175
l2mean (01)	0.753	0.267	0.023	−0.095
l2mean (z)	0.729	0.243	−0.001	−0.119
l2mean (simple)	0.735	0.249	0.005	−0.113
l2mean (clipped 1)	0.703	0.216	−0.028	−0.145
l2mean (clipped 2)	0.731	0.244	0.0	−0.117
l3mean (01)	0.751	0.265	0.021	−0.097
l3mean (z)	0.728	0.242	−0.002	−0.12
l3mean (simple)	0.734	0.247	0.003	−0.114
l3mean (clipped 1)	0.706	0.22	−0.024	−0.142
l3mean (clipped 2)	0.734	0.247	0.003	−0.114
l4mean (01)	0.75	0.263	0.019	−0.098
l4mean (z)	0.728	0.241	−0.003	−0.12
l4mean (simple)	0.733	0.247	0.003	−0.114
l4mean (clipped 1)	0.708	0.222	−0.022	−0.14
l4mean (clipped 2)	0.735	0.249	0.005	−0.112

Table 3.4: Anomaly detection ensemble comparison (Metric 1,5-7, part 2)

Methods	Mean AUC	ΔAUC to		
		Minimum	***Mean***	***Maximum***
Threshold ($t = -1$)	0.757	0.271	0.027	−0.091
Threshold ($t = -2$)	0.756	0.27	0.026	−0.091
Threshold ($t = 0$)	0.756	0.27	0.026	−0.091
Threshold ($t = 1$)	0.746	0.26	0.016	−0.101
Threshold ($t = 2$)	0.719	0.233	−0.011	−0.129
knn ($k = 1$)	0.757	0.271	0.027	−0.091
knn ($k = 3$)	0.757	0.27	0.026	−0.091
knn ($k = 5$)	0.756	0.27	0.026	−0.092
knn ($k = 10$)	0.752	0.266	0.022	−0.096
knn ($k = 100$)	0.664	0.178	−0.066	−0.184
IFor	0.754	0.268	0.024	−0.093
Gaussian	0.754	0.268	0.024	−0.094
Clustered	0.752	0.265	0.021	−0.096
Greedy	0.751	0.264	0.02	−0.097
IRT	0.731	0.245	0.001	−0.117

our samples, and does not work well with few data points.

Our last methods, the greedy ensemble, and the IRT approach perform well, but also not exceptionally so. And since in most cases we can achieve better results with less work, they don't seem worth the extra effort in most situations.

Finally, when we look at the last columns, we can reach an ΔErr of almost 1. This means that when a random model misclassifies a sample, on average, there is only a $e^{-0.955} \approx 38\%$ chance that the maximum (01) ensemble also misclassifies it. So unless in very special situations, using an ensemble is generally a good idea.

3.5.4 Model contributions

Next to choosing the right combination procedure, we also study which models create the best ensembles. For this, we calculate the average AUC score for each ensemble considering a given model. We are no longer able to use all our ten metrics, as most metrics would compare the combined AUC score to the performance of individual models, which also changes when we remove/add a given model.

In Figure 3.2, we show the difference in the average performance when using all models for the maximum combination function and 01 normalization. We only show this depen-

Table 3.5: Anomaly detection ensemble comparison (Metric 1,8-10, part 1)

Methods	Mean AUC	ΔErr compared to		
		Minimum	***Mean***	***Maximum***
maximum (01)	**0.766**	**1.741**	**0.955**	**-1.386**
maximum (z)	0.758	1.601	0.814	−1.527
maximum (simple)	0.723	1.37	0.584	−1.757
maximum (clipped 1)	0.66	0.487	−0.299	−2.64
maximum (clipped 2)	0.712	0.814	0.027	−2.314
mean (01)	0.754	1.599	0.813	−1.528
mean (z)	0.756	1.604	0.817	−1.524
mean (simple)	0.734	1.505	0.718	−1.623
mean (clipped 1)	0.747	1.199	0.412	−1.928
mean (clipped 2)	0.754	1.321	0.534	−1.807
median (01)	0.752	1.401	0.614	−1.727
median (z)	0.755	1.447	0.661	−1.68
median (simple)	0.744	1.381	0.595	−1.746
median (clipped 1)	0.743	1.056	0.269	−2.072
median (clipped 2)	0.753	1.295	0.509	−1.832
minimum (01)	0.673	0.687	−0.1	−2.441
minimum (z)	0.671	0.66	−0.126	−2.467
minimum (simple)	0.742	1.203	0.417	−1.924
minimum (clipped 1)	0.667	0.629	−0.158	−2.499
minimum (clipped 2)	0.672	0.66	−0.126	−2.467
l2mean (01)	0.753	1.649	0.862	−1.479
l2mean (z)	0.729	1.311	0.524	−1.817
l2mean (simple)	0.735	1.458	0.671	−1.67
l2mean (clipped 1)	0.703	0.803	0.016	−2.325
l2mean (clipped 2)	0.731	1.087	0.3	−2.041
l3mean (01)	0.751	1.637	0.851	−1.49
l3mean (z)	0.728	1.314	0.528	−1.813
l3mean (simple)	0.734	1.444	0.657	−1.684
l3mean (clipped 1)	0.706	0.827	0.04	−2.301
l3mean (clipped 2)	0.734	1.115	0.329	−2.012
l4mean (01)	0.75	1.604	0.817	−1.523
l4mean (z)	0.728	1.316	0.53	−1.811
l4mean (simple)	0.733	1.436	0.649	−1.692
l4mean (clipped 1)	0.708	0.844	0.058	−2.283
l4mean (clipped 2)	0.735	1.132	0.346	−1.995

Table 3.6: Anomaly detection ensemble comparison (Metric 1,8-10, part 2)

Methods	Mean AUC	ΔErr compared to		
		Minimum	***Mean***	***Maximum***
Threshold ($t = -1$)	0.757	1.608	0.821	−1.52
Threshold ($t = -2$)	0.756	1.604	0.817	−1.524
Threshold ($t = 0$)	0.756	1.599	0.813	−1.528
Threshold ($t = 1$)	0.746	1.5	0.713	−1.628
Threshold ($t = 2$)	0.719	1.295	0.508	−1.833
knn ($k = 1$)	0.757	1.543	0.757	−1.584
knn ($k = 3$)	0.757	1.537	0.751	−1.59
knn ($k = 5$)	0.756	1.523	0.736	−1.605
knn ($k = 10$)	0.752	1.48	0.693	−1.648
knn ($k = 100$)	0.664	1.044	0.258	−2.083
IFor	0.754	1.352	0.566	−1.775
Gaussian	0.754	1.397	0.611	−1.73
Clustered	0.752	1.694	0.907	−1.434
Greedy	0.751	1.339	0.552	−1.789
IRT	0.731	1.125	0.338	−2.003

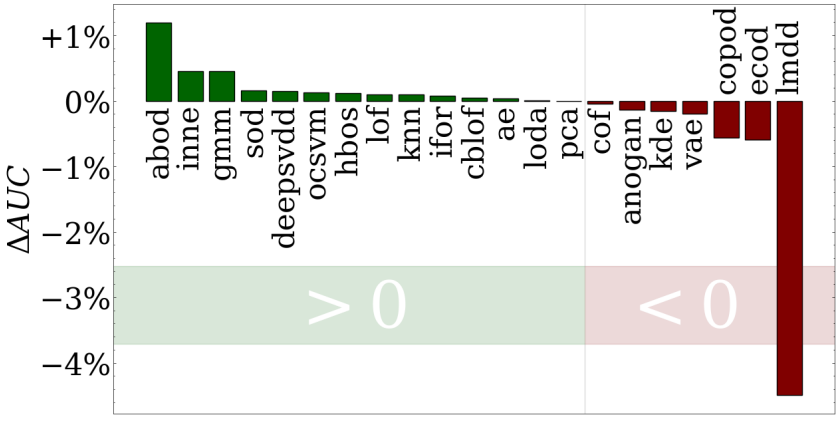


Figure 3.2: Average ROC AUC score of ensembles using a specific submodel, compared to the average of all models considered. Here only shown for the maximum (01) ensembles, as this produces the best ensembles, and the rest look very similar.

dency, as it has the best overall performance and the other ensemble methods show similar relations.

ABOD seems to be by far the most helpful method to include in an ensemble. We think this is happening because its approach of considering angles is fairly unique, and so many models can profit from the new concept it provides. Overall, ABOD should not be neglected in any ensemble, as its inclusion alone affects the resulting ensemble as much as choosing a slightly better ensemble procedure. Interestingly, we have learned since writing the paper this chapter is based on that the implementation of ABOD used here was likely faulty[1]. So it is surprising that this algorithm is still more useful than other algorithms. Our explanation for this is that combining multiple submodels requires mostly diversity from each submodel, and thus, a slightly hindered performance of a creative algorithm does not necessarily need to be a drawback.

Other algorithms seem to actively hurt the ensemble performance. We do not suggest adding the Lmdd algorithm to any ensemble and would also suggest not employing Copod or Ecod except in special situations.

Interestingly, algorithms with similar concepts also perform relatively similarly. Both the Autoencoder and the Pca are based on reconstruction errors, the Anogan, KDE and variational autoencoder all model densities and DeepSVDD is partially based on the concepts of the OCSVM. All of these groups perform quite similarly, reinforcing our belief that the diversity of concepts is an important goal for each ensemble.

To study this further, we take a look at higher-order combinations. Namely, we look at the average AUC score for each ensemble that also considers two specific models simultaneously. This reduces our statistics, but we still consider about 280 thousand ensembles for each combination of models.

In Figure 3.3 we show the average ROC AUC score of every ensemble that includes two different submodels, again only for maximum (01) ensembles. We compare this AUC score to the expected performance considering each submodel.

$$AUC(m_1, m_2, \ldots) - \frac{(AUC(m_1, \ldots) + AUC(m_2, \ldots))}{2}$$

A positive score (green color) means ensembles using both models perform better than the expectation, while a negative score (red color) means they perform worse than expected. Overall, we see a group of models (from ABOD, Inne till autoencoder) that only have positive interactions with each other. Because of this, we suggest using them to build an effective ensemble.

[1] See https://github.com/yzhao062/pyod/issues/522

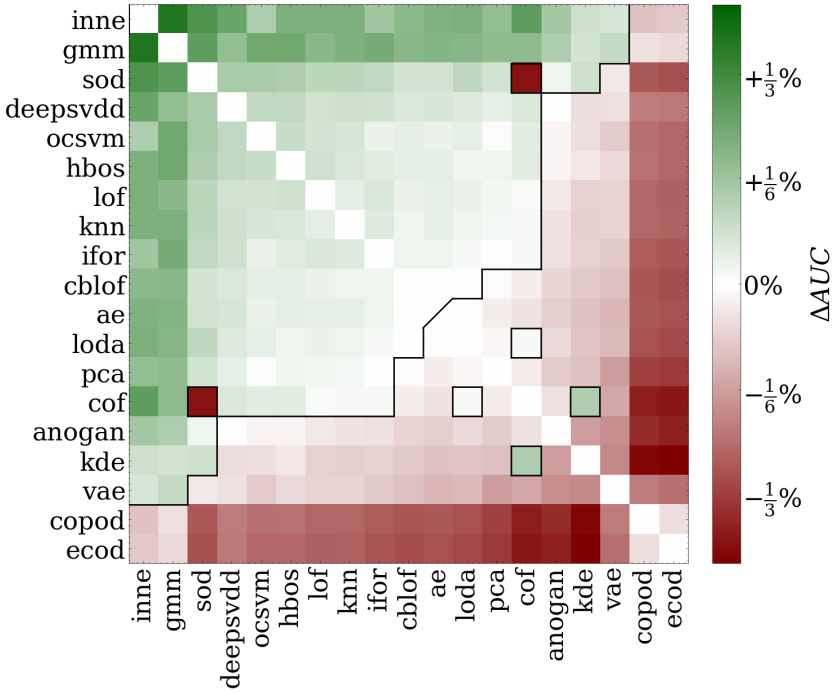


Figure 3.3: Average AUC score of ensembles also considering two different models. We compare this to the expected score considering the average performance of both models individually. This means that each diagonal value always has a difference in AUC score of 0.0. We removed the two most extreme models ABOD and Lmdd, to increase the readability of this plot.

But overall, this relation looks quite similar to the contribution of a single model. The same six models are either much more or less useful than the rest. The only exception to this are a few interactions that behave strangely, all including the COF model. This also implies, that the strengths of the SOD model might be undervalued in Figure 3.2.

3.6 Conclusion

In this chapter, we created the most extensive comparison of combination procedures of heterogeneous ensembles for weakly supervised anomaly detection by far. To the best of our knowledge, we are also the first to study not only the effect of different combination functions but also of various normalizations and the contributions of individual models.

Using insights from each of these, we extract the following best practices for the creation of such ensembles:

1. From the set of algorithms (Abod, Inne, gmm, SOD, DeepSVDD, OCSVM, HBos, Lof, knn, ifor, cblof, autoencoder) train as many models on the training data as possible.

2. When computation time is a limiting factor, ignore some of the later models.
3. Choose a combination procedure
    - When it is possible to approximate the performance of the resulting ensemble, choose a maximum combination function with a 01 normalization.
    - If this is not the case, or the resulting ensemble performs poorly, rely instead on a simple mean of the model outputs, as well as a z-score normalization.
4. Normalize each model output using the chosen normalization and combine the resulting values using the chosen combination function.
5. Use the resulting ensemble to score new samples.

Learning from this for our own ensembles, we find that the benefit of complicated combination functions is limited. Very simple combination functions can outperform all of them. The maximum combination can be the best, but it is also very sensitive to the effects of normalization functions. The opposite is achieved by the mean-based combination functions, which perform slightly worse but do so more reliably and less dependent on normalization effects.

We were also able to show that stacking for weakly supervised anomaly detection is a rather promising research direction. Stacking methods can outperform each other combination function in at least one of the metrics we considered. We think they are only limited, because our approaches here are still quite simple. Thus, this was further investigated by a thesis student (Thiyaagu, 2024), who verified that stacking approaches are beneficial but also was not able to prove a significant improvement.

Part II

Abstract Anomaly Detection

CHAPTER 4

DEAN: DEEP ENSEMBLE ANOMALY DETECTION

This chapter follows: Klüttermann, S., Katzke, T., & Müller, E. (2025). Unsupervised surrogate anomaly detection [To appear]. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*. https://arxiv.org/abs/2504.20733

4.1 Introduction

To solve the task of weakly supervised anomaly detection, various anomaly detection algorithms have been proposed. Methods such as AnoGan (Mattia et al., 2021) or Kernel Density Estimation (Rosenblatt, 1956) aim to model the probability density distribution of normal samples and consider samples in low-density regions as abnormal. Other algorithms, like KNN (Ramaswamy et al., 2000), consider samples that are further away in the variable space from other samples as more anomalous. Approaches like Isolation Forests (F. T. Liu et al., 2008) measure how easily samples can be separated. Still, it can be shown that both of these alternative approaches effectively model densities too (Buschjäger et al., 2022; Gu et al., 2019).

While density estimation is an intuitive solution to anomaly detection, any method based on this approach has two fundamental drawbacks. First, they do not scale well to high-dimensional data, which is known as the curse of dimensionality (Bellman, 1966). Second, there is usually no explicit modeling of the data-generating process involved, which limits how well the method generalizes to new data.

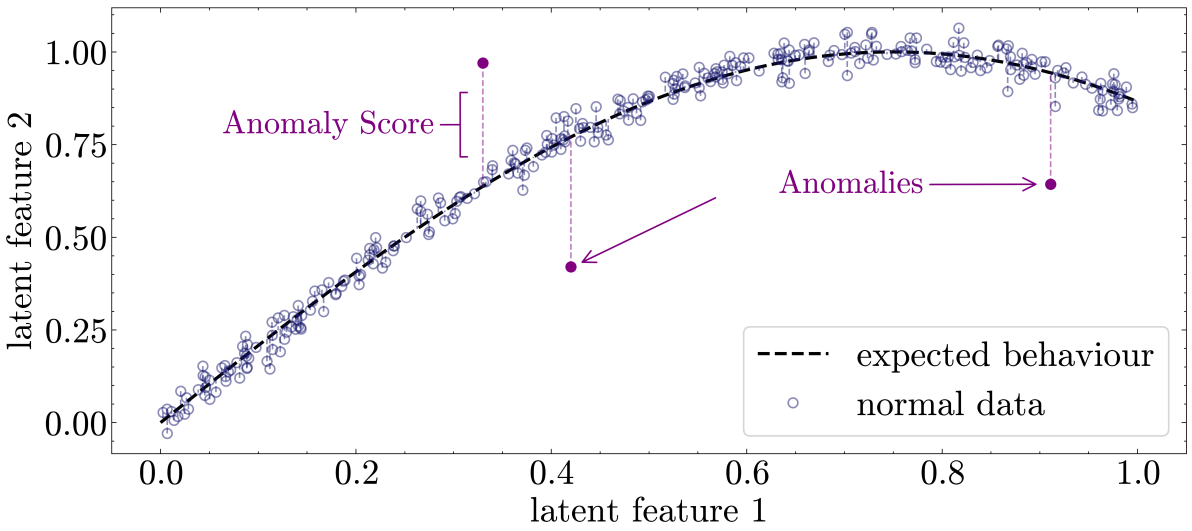


Figure 4.1: Example of surrogate anomaly detection: Anomalies are detected by learning a representation that encodes the regular patterns of normal data and measuring deviations from the expected behavior.

In contrast to modeling complex density distributions for high dimensional input data, we consider in this chapter algorithms that learn low dimensional representations as approximations of the underlying regular patterns in the data. More specifically, we are inspired by surrogate models in engineering applications (Koziel et al., 2011), where simple surrogate models are used to abstract more complex or expensive processes. Similarly, we search for models that capture a pattern of the underlying data-generating process instead of modeling the whole distribution, which we will subsequently refer to as surrogate anomaly detection models. An illustrative toy example of such a model is shown in Figure 4.1.

Some existing algorithms can be considered instantiations of such surrogate anomaly detection models. Autoencoder (Sakurada & Yairi, 2014) and PCA-based anomaly detection (Callegari et al., 2011) learn an identity function to compress regular data and measure the deviation of anomalies as the reconstruction error. DeepSVDD (Ruff et al., 2018) learns a representation in which regular data can be modeled by mapping it to a lower dimensional constant, where anomalies deviate highly from this constant value. Because these algorithms don't need to model the entire density distribution in its original input space, they scale more effectively to high dimensional data (W. Chen et al., 2020).

However, these methods are not without limitations either. Unlike density estimation methods, the objective of training a surrogate is much less well-defined, leading to an unlimited amount of options on how to create such a surrogate, each with its own drawbacks.

We exploit this variability to formalize the idea of surrogate models into a blueprint for creating arbitrary surrogates. Within this framework, we propose five axioms that an ideal surrogate model should satisfy. Based on these, we suggest a new surrogate AD algorithm called DEAN, which, to the best of our knowledge, is the first algorithm adhering to all of them.

We evaluate our algorithm by following the procedure outlined in a recent benchmark survey paper (Han et al., 2022), comparing it against 19 competitors across 121 datasets. Our algorithm performs highly competitively, showing only minor performance differences with the best non-surrogate competitors and outperforming all other surrogate-based methods.

4.2 Surrogate Models

Surrogate models are simplified approximations of more complex or computationally expensive models (Koziel et al., 2011) and thus a subfield of abstraction. In engineering, they are commonly used when, for example, physical simulations are too costly (Mora-Mariano & Flores-Tlacuahuac, 2022). In machine learning, surrogate models have been used to approximate, accelerate, or explain other machine learning models. This has been applied to uncertainty estimation (Sudret et al., 2017), explainability (either globally (Monteiro & Reynoso-Meza, 2023), or locally (Ribeiro et al., 2016)), surrogate task-based models (F. Ye et al., 2021), and to accelerate anomaly detection (Flusser & Somol, 2022).

In contrast, we propose surrogate anomaly detection to learn an approximation of the regular patterns in the input data such that anomaly detection can use these surrogate models to measure deviation in a more reliable way then with traditional density estimation, especially for high dimensional input data.

4.3 Theory of Surrogate Anomaly Detection

In engineering applications, surrogate models are frequently employed to abstract processes that are too complicated to model directly. Similarly, when it comes to anomaly detection, it may be impractical to model a complex distribution directly. We define a surrogate as a model that approximately learns abstract patterns characteristic of normal samples and identifies anomalies through deviations from these patterns.

In general, we can write any such an approximative pattern as

$$f(x) \approx g(x), \ \forall x \in X \tag{4.1}$$

where, $f : \mathbb{R}^d \to \mathbb{R}^k$ is a learnable function that aims to capture the underlying structure of normal data by approximating a pattern $g : \mathbb{R}^d \to \mathbb{R}^k$, with $X \subset \mathbb{R}^d$ representing the set of normal data samples. For this purpose, f may be realized by training a neural network to map each data sample to a latent representation, for complex data typically with lower dimensionality $k \ll d$.

The pattern g could, for example, be chosen as the identify function ($g_{AE}(x) = x$, as with an autoencoder), but since sufficiently powerful neural networks are universal function approximators, there are no inherent restrictions for the choice of g.

To assign an anomaly score that quantifies to which extent Equation 4.1 does not hold for a sample x, we can measure the deviation between $f(x)$ and $g(x)$:

$$score(x) = \| f(x) - g(x) \| \tag{4.2}$$

Since we search for a pattern that is satisfied by normal training samples X_{train}, we can minimize Equation 4.2 to create a loss $\mathscr{L}$ to train this model:

$$\mathscr{L} = \sum_{x \in X_{train}} score(x) \tag{4.3}$$

Equations 4.2 and 4.3 provide a general framework for developing new surrogate anomaly detection algorithms from a pattern g. While any g consistent with our function definition can be used, their effectiveness in yielding a well-performing anomaly detector may differ significantly.

4.3.1 Surrogate Axioms

To guide the selection of g, we propose five axioms to which we believe an optimal surrogate algorithm should adhere to. We assume here that a performance measure $m(f)$ exists (e.g. ROC-AUC), that evaluates how well a model seperates anomalies from normal samples.

First, we note that the comparison in Equation 4.2 depends not only on the relative deviation of $f(x)$ from $g(x)$, but also on the magnitude of $\|g(x)\|$. This means that when the size of $g(x)$ varies significantly, some samples are considered more or less anomalous regardless of what data is considered normal.

Axiom 1 (Scale Consistency)**.** *The pattern function g should produce outputs of similar scale for all inputs:* $\forall x_1, x_2 \in \mathbb{R}^d$ *it holds that* $\|g(x_1)\| \approx |g(x_2)\|$.

An optimal surrogate must be capable of being trained with reliable results.

Axiom 2 (Reliable Training Procedure)**.** *When learning to approximate g multiple times under identical training conditions, the performance variance should be small. For learned instances* $f_1, ..., f_n$ *it holds that* $Var(m(f_i)) \leq \delta^2$*, where the constant* $\delta > 0$ *is chosen as small as possible.*

It is also crucial to avoid trivial solutions, namely learning a function f that minimizes the loss objective, but has no ability to discern between normal and anomalous samples.

Axiom 3 (No Trivial Solutions)**.** *There should exist no trivial solution* $f_{trivial}$ *as a (local) minima of the loss* $\mathscr{L}$*, i.e.* $\nabla\mathscr{L}(f_{trivial}) = 0$*, which irrespective of the input* $x \in \mathbb{R}^d$*, outputs a near-constant value* $f_{trivial}(x) \approx c$ *for a constant* $c \in \mathbb{R}^k$.

Hyperparameter selection is a significant challenge in anomaly detection (See Chapter 10). Thus, we expect our surrogate to be stable under hyperparameter changes.

Axiom 4 (Hyperparameter Invariance)**.** *The model performance should be stable across reasonable hyperparameter variations. For any two such hyperparameter sets* H_A *and* H_B *and the instances* f_{H_A} *and* f_{H_B} *learned based on them, it holds that:* $|m(f_{H_A}) - m(f_{H_B})| \leq \eta$*, where* $\eta > 0$ *is chosen as small as possible.*

Finally, anomalies might be complicated, and thus, we also require expressive power for the model to approximate even highly complex patterns.

Axiom 5 (Complex Pattern Learning)**.** *The learnable function f needs to be represented by a universal function approximator, capable of approximating any continuous function* $g : \mathbb{R}^d \to \mathbb{R}^k$ *to arbitrary precision on subsets of* $\mathbb{R}^d$.

4.3.2 Axiom Compliance

Both of the most well-established deep anomaly detection approaches that fall under our surrogate definition do not fulfill the previously introduced axioms.

An **Autoencoder** (Sakurada & Yairi, 2014) is created from the function $g_{AE}(x) = x$, and thus trains a neural network to approximate the identity function. To keep this from learning the trivial solution of each network layer representing $x_{i+1} = 1 \cdot x_i + 0$, a compression

step is introduced. In this step, the input samples are compressed to a latent-size dimensional representation before being decompressed again. This violates Axiom 4, since the latent-size must be chosen correctly, significantly affecting how the autoencoder works. There is also a common local minimum of the loss, when $f(x) = mean(X)$ for some features, which can be hard to notice (Axiom 3). Additionally, the scale of g is not constant (Axiom 1), resulting in a biased result.

DeepSVDD (Ruff et al., 2018) is a surrogate model constructed from the pattern function $g_{SVDD}(x) = c$ for a constant c. To make the model less likely to learn the trivial solution of a constant prediction independent of x, they suggest removing the bias terms added to each network layer. This limits the capacity to which complex functions can be learned, thereby violating either Axiom 3 or Axiom 5.

4.4 A Minimal Surrogate: The DEAN Model

Given that no surrogate known to us adheres to all aforementioned axioms, we propose a novel approach. We observe that the complexity of the pattern function g is directly correlated with the arbitrary weighting of different samples (Axiom 1). Moreover, increased complexity in the function that needs to be learned intensifies the challenges in the training procedure (Axioms 2 and 3). Thus, we advocate for the selection of the simplest possible function g that adequately identifies the data patterns.

Depending on how to measure the complexity of a mathematical function, the simplest might be $g_0(x) = 0$. However, this surrogate breaks Axiom 3, as there is a local minima in which every weight in the last layer of a neural network is set to zero (Ruff et al., 2018). And while this minimum might not always be found (Karr et al., 2022), or regularization terms could be used, either would break our axioms. Instead, we consider $g_{DEAN}(x) = 1$ and the surrogate generated by it. While there is still a local minimum through the bias terms of the last neural network layer, it is more manageable, as we show later (Section 4.4.1).

Thus, we will train a neural network to output a constant value of 1. Following our framework, we can do this with the following loss and score functions.

$$\mathscr{L} = \sum_{x \in X_{train}} \|f(x) - 1\|, \qquad score(x) = \|f(x) - q\| \tag{4.4}$$

We use here $q = \frac{1}{\|X_{train}\|} \sum_{x_T \in X_{train}} f(x_T) \approx 1$ instead of $q = 1$ to make sure the distribution of normal samples is centered around q.

Since we only learn a one-dimensional pattern here, in the worst case, we only locally fix one feature as a function of the remaining ones, resulting in many false negatives. To solve this, we suggest using an ensemble (C. C. Aggarwal & Sathe, 2015) to combine many surrogates trained with g_{DEAN} into a more effective model F (F. T. Liu et al., 2008). Here we use a integer constant $power$ to guide how different models are combined.

$$score_F(x) = \frac{1}{\|F\|} \sum_{f_i \in F} \|f_i(x) - q\|^{power} \tag{4.5}$$

Due to the simplicity of our surrogate, training one neural network usually only takes seconds and produces a weak anomaly detection algorithm; thus allowing and forcing us to combine a large number of submodels. The use of separate fully independent neural networks makes our method very easy to parallelize, facilitates that the learned patterns are less correlated, and allows us to use ensemble methods like feature bagging (Lazarevic & Kumar, 2005) to decrease it further. Feature bagging additionally can ensure a constant number of features for each submodel, resulting in a close-to-constant runtime for high-dimensional data.

We call this setup *DEAN* (Deep Ensemble ANomaly detection).

4.4.1 Axiom Compliance of DEAN

While existing surrogate algorithms only adhere to some of our surrogate axioms, DEAN is constructed to fulfill all of them. This we want to show here.

Since $g(x_1) = g(x_2)$ holds for all samples x_1, x_2, Axiom 1 is fulfilled. Axioms 2 and Axioms 4 can only be validated experimentally, see for this Section 4.5.1.

More complicated are Axioms 5 and 3, as the DEAN objective can be flawlessly fulfilled by $f_{trivial}(x) = 0 \cdot x + 1$. Since such a neural network is independent of its input values, its performance is equivalent to random, and thus, we have to make sure another network parameter configuration except $f_{trivial}$ is learned. DeepSVDD has a similar problem and suggests removing the bias terms from the neural network (Ruff et al., 2018). However, this can not be done, without limiting the complexity of the learned patterns and breaking Axiom 5. Instead, we utilize that DeepSVDD often still works when bias terms are used (Klüttermann, 2022), since neural network optimizers are not perfect and often find other solutions before the trivial one. To make this effect more reliable, we suggest removing bias terms only from the last neural network layer. This makes the trivial solution significantly more complicated and thus less likely to be found, while still making sure that the neural networks is able to fit any function and thus that Axiom 5 is fulfilled.

We show this effect in Figure 4.2, where we task neural networks to approximate a simple sinus curve. Here, we use neural networks with three layers of 100 nodes and relu activation in each hidden layer. The three networks differ only by the bias terms they use. While the network with bias terms (green) is clearly able to approximate the sinus curve, the version without bias terms (blue) is not able to do so. And since real anomaly representations can be much more complicated than such a simple sinus curve, we do not think that limiting the neural network complexity is a reasonable choice.

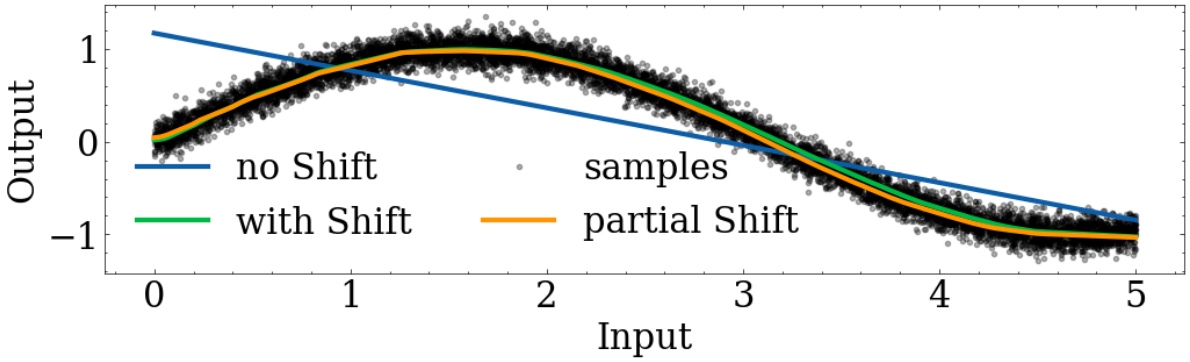


Figure 4.2: Given complicated alinear data, the functions learned by three neural networks with relu activations are shown. The network without bias terms cannot capture the structure of the underlying data, while both a network with bias terms in each layer and a network with bias terms in all layers except the last can describe the alinearity.

Additionally, we can use the ensemble structure of DEAN to guarantee Axiom 3: If a trivial solution is found, the ensemble structure makes sure it does not affect the result. Since $f_{trivial}(x) = 1 \; \forall x$, the contribution to Equation 4.5 is always zero and thus DEAN's performance stays constant when enough submodels are used.

4.4.2 DEAN Hyperparameters

Generally, choosing hyperparameters for an unsupervised or weakly supervised anomaly detection algorithm is a hard problem (Y. Zhao, Rossi, & Akoglu, 2021). Without access to a labeled ground truth, it is impossible to determine which set of hyperparameters works best for a certain dataset. Although our hyperparameters could be fine-tuned for maximum performance on specific datasets, this approach risks poor generalization to other datasets. Instead, we suggest a set of reasonable hyperparameters that, while not necessary optimal for our datasets, are likely work to generalize well across different datasets. We consider other hyperparameters in Section 4.5.1.

For each of our submodels, we train feedforward neural networks with three hidden layers of 255 neurons each. These layers have a ReLU activation, while the output layer (with one neuron) has a SELU activation to prevent dead neurons (L. Lu et al., 2020). We choose a lower than standard learning rate of 0.0001 since we explicitly benefit from the neural network being stuck at diverse local minima. We also choose a relatively high batch size of 512 to make training our ensemble faster. We train for 50 epochs using early stopping with a patience of 10 epochs. When a dataset has at least 200 features, we use feature bagging with 200 random features per model. If a dataset has less than 200 features, we use all features to make sure any important correlations can be mapped. We choose $power = 9$ to emphasize high deviations in one model over multiple small ones as a compromise between the mean and maximum-based combination functions suggested in Chapter 3.

4.5 Experimental Evaluation

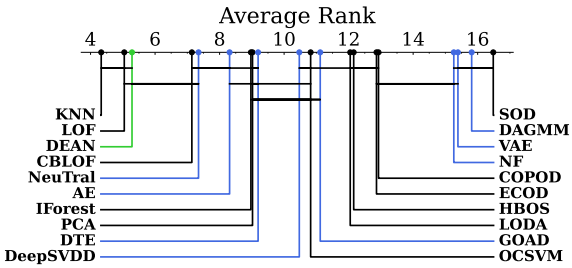
To test our method, we follow the benchmark paper, ADBench (Han et al., 2022), which suggests 121 datasets (57 of which are entirely uncorrelated) to benchmark unsupervised and weakly supervised anomaly detection algorithms on, as well as algorithms to compare to.

Also following this benchmark, we compare against KNN (Ramaswamy et al., 2000), LOF (Breunig et al., 2000), CBLOF (Z. He et al., 2003), Isolation Forest (F. T. Liu et al., 2008), PCA (Callegari et al., 2011), DeepSVDD (Ruff et al., 2018), OCSVM (Bounsiar & Madden, 2014), LODA (Pevný, 2016), HBOS (Goldstein & Dengel, 2012), COPOD (Z. Li et al., 2020), ECOD (Z. Li et al., 2022), SOD (Kriegel et al., 2009) and DAGMM (Zong et al., 2018). Additionally, we use a regular Autoencoder (Sakurada & Yairi, 2014), as it is a surrogate algorithm, as well as a variational autoencoder (Kingma & Welling, 2014) and a normalizing flow (Rezende & Mohamed, 2015) as deep learning density competitors. Similarly, we compare against a few more modern algorithms that were not yet considered in ADBench: We use NeuTral (Qiu, Pfrommer, et al., 2022), based on contrastive learning, DTE (Livernoche et al., 2024), based on diffusion models, and GOAD (Bergman & Hoshen, 2020), based on geometric transformations. In total we compare DEAN against 19 different algorithms.

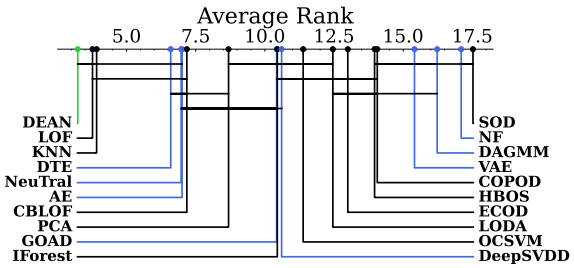
In comparison to ADBench, we train every model on uncontaminated data (weakly supervised setting). We use the ROC-AUC score to evaluate each method.

We give critical difference plots in Figure 4.3. There, we use a Friedman test (Friedman, 1940) to see if any significant differences between algorithm performances (measured in ROC-AUC) exist and connect algorithms that have no significant difference by a Wilcoxon

test (Wilcoxon, 1945). We consider p-values below $p \leq 5\%$ after Bonferroni-Holm correction (Holm, 1979) as significant.



((a)) Critical difference plot using all ADBench datasets.



((b)) Critical difference plot only using the larger half of datasets.

Figure 4.3: Critical difference plots comparing DEAN's ROC-AUC performance to other algorithms. A lower rank is better, and algorithms that don't differ significantly are connected by a horizontal line. While our algorithm is not always the best, there is no algorithm that performs significantly better than DEAN. Also, when only considering large datasets, DEAN achieves the best performance.

Our critical difference plots are slightly different than the one shown in ADBench, partially because we evaluated the algorithms in the weakly supervised setting. However, similar to ADBench, we see that simple, well-known, and shallow algorithms like KNN, LOF, and CBLOF are the strongest competitors. Our algorithm outperforms CBLOF but does not quite reach the same average rank as KNN and LOF. We think that this is because benchmark datasets often are relatively low dimensional and have many samples, a scenario that is not realistic in practice but optimal for distance-based algorithms like KNN and LOF. Still, there is no significant difference between these algorithms and DEAN. And if we only consider larger datasets, DEAN even has a slightly higher rank than all other algorithms. Also, DEAN performs significantly stronger than each other surrogate algorithm or deep learning algorithm (with the exception of NeuTral).

4.5.1 Evaluation of Axiom Compliance

Compliance with Axioms 2 and 4 is hard to assess theoretically, which is why we evaluate it experimentally on the same datasets. For Axiom 2, we show that the repetition uncertainty of DEAN is smaller than for other surrogate deep learning algorithms under identical training conditions in Figure 4.4. Additionally, for Axiom 4 we also show DEANs performance when evaluated with modified hyperparameter sets. Since both uncertainties are almost equal, it can argued that the effect of these hyperparameter sets is close to neglectable,

which is in stark contrast to DeepSVDD (Han et al., 2022) and Autoencoder (Kumar et al., 2024) behaviour.

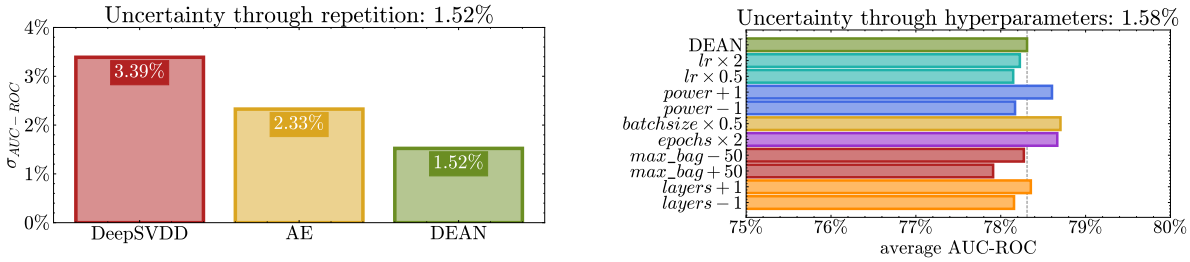


Figure 4.4: Left: Repetition uncertainty for various surrogate algorithms, Right: DEAN performance with varied hyperparameters and the resulting uncertainty.

4.5.2 Runtime

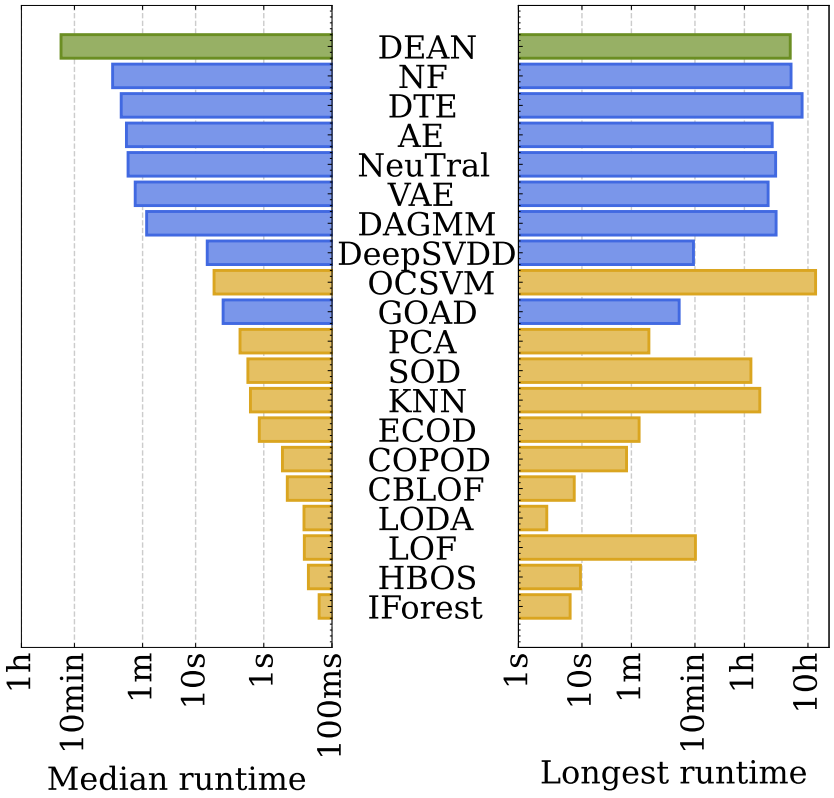


Figure 4.5: Left: median runtime per dataset of our algorithm compared to our competitors, Right: Maximum runtime of each algorithm on the datasets. We show DEAN in green, the remaining deep learning algorithms in blue, and shallow algorithms in yellow.

We compare the median runtime of the algorithms considered in Figure 4.5. Since some datasets are significantly larger than others, the runtimes vary significantly between datasets. Because of this, we show the median runtime over each dataset as well as the maximum over all datasets. To keep the comparison fair, we only use a single CPU core per algorithm and dataset.

Generally, we see deep learning algorithms perform slowest, with DEAN being the slowest algorithm with a median runtime of about $15min$ per dataset. Still, these algorithms would also be the ones most affected when using GPUs. Additionally, DEAN is an ensemble method and thus can be parallelized almost perfectly. Also, because of DEAN's use of feature bagging, the comparison is significantly more similar when considering the worst longest runtime instead. Here, most deep learning approaches require comparable amounts of time, and three even require more. The difference results from DEAN's runtime scaling faster to higher dimensional datasets, as feature bagging results in constant neural network sizes. Still, the runtime of DEAN is not optimal, and thus, we will consider a variation of DEAN optimized for runtime in Chapter 5.

4.5.3 Ensemble Structure

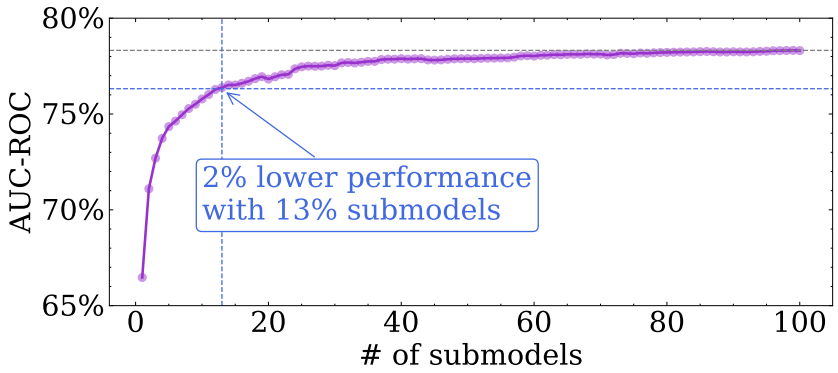


Figure 4.6: Average DEAN ROC-AUC performance over all datasets as a function of the number of submodels used. We reach 2% lower performance with the first 13% of submodels.

The hyperparameter affecting the runtime the most is the number of submodels used. While the performance also depends on how many submodels are used, the relationship is not linear as Figure 4.6 shows. Even with the 100 submodels we employed for the benchmark experiments, we do still see an improvement from using more submodels. However, the change in average performance slows down significantly compared to the first submodels used.

This means we could train a version of DEAN in 87% less time, which would only have a 2% lower average performance.

4.5.4 Existing Surrogates as Ensembles

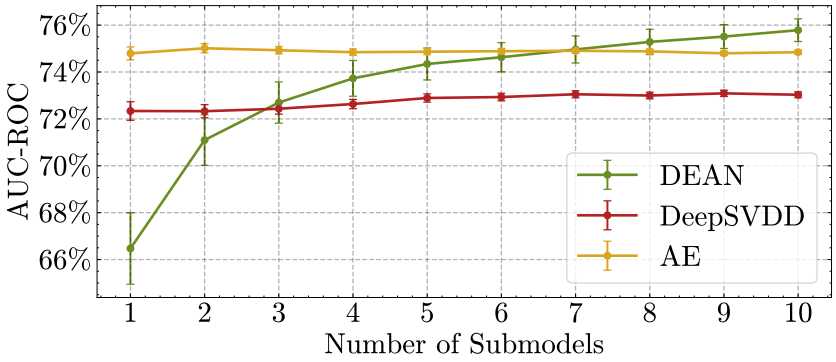


Figure 4.7: Comparing other deep learning surrogate models as ensembles. The average performance over all datasets is shown here. While DEAN improves significantly when more submodels are used, both DeepSVDD and Autoencoder stay almost constant. Please note that the uncertainty of DEAN is higher here compared to Figure **??**, as we are not comparing the entire ensemble, but only a limited number of submodels.

Since we compare our ensemble-based method mostly against non-ensemble methods, we also show in Figure 4.7 the performance of the other deep learning-based surrogate methods when used in an ensemble. While DEAN's performance improves with more submodels, the performance of the other methods stays almost constant. The continuous growth of DEAN with more submodels is a direct consequence of the simple submodels we use and the high variance between submodels they allow. This shows the benefits of abstract ensemble approaches, where individual submodels are able to consider slightly different aspects.

4.6 Anomaly Detection Beyond Accuracy

While our comparison shows the competitive performance of DEAN, it also highlights the problem of creating a strictly better-performing anomaly detection algorithm. Even when using 121 datasets, we are not able to find many significant differences between the performances of various algorithms.

Thus, we argue that while accuracy is vital up to a certain level, usability might be more beneficial than marginally improving an algorithm's performance beyond competitive performance.

Beyond achieving competitive performance, it is important to note that not every task has the same exact requirements. Other surrogate algorithms like DeepSVDD and Autoencoder are arguably prominent because they are easy to modify and extend. For instance, they allow the incorporation of fairness criteria (H. Zhang & Davidson, 2021) and sparse ground truth through a few labeled instances (Ruff et al., 2019), or the integration of preprocessing (Kieu et al., 2022) to boost robustness and explainability. We argue that DEAN is even more versatile than these algorithms, thanks to its simple submodel design and inherent ensemble structure.

The ensemble structure, for example, can be used to explain why specific samples are considered normal or abnormal by leveraging shapley values (See Chapter 7). Here, feature bagging provides a natural solution to the high computational complexity of shapley values. Similarly, the ensemble structure should make it possible to create a version of DEAN with federated learning (X. Wang et al., 2023). Additionally, techniques such as subsampling (Zimek et al., 2013) could be used to create ensembles that change what data they are trained on to be GDPR compliant (Menges et al., 2021). A similar idea could be used to create a version of DEAN that uses active learning to either improve or personalize an anomaly detection algorithm (Lochner & Bassett, 2021). Feature bagging (Lazarevic & Kumar, 2005) instead could likely be used to allow the model to handle missing values (Dietterich & Zemicheal, 2018). And since retraining singular submodels can be done in parallel to predictions, a continuously learning (Stocco & Tonella, 2020) version could be created. Ensembles also allow for ensuring specific properties of our anomaly detection algorithm, such as robustness against adversarial samples, by employing pruning or weighting techniques to remove the least robust ensemble submodels (See Chapter 6).

Similarly, the simplicity of our submodels allows for the easy modification of the training procedure. This allows the input of additional information (See Chapter 8) like, for example, in semi-supervised anomaly detection (Ruff et al., 2019) or with outlier exposure (Hendrycks et al., 2019) and could likely even be used for privacy-preserving anomaly detection (Alabdulatif et al., 2017).

In addition to this, we also think that using different machine learning models could lead to interesting versions of DEAN. Neural symbolic computing (Paisner et al., 2013) could be used to extract human understandable patterns and a more lightweight model could be directly applied on IoT devices (Sedjelmaci et al., 2016). Also, graph neural net-

works (J. Zhou et al., 2018) and recurrent neural networks (Rumelhart et al., 1986) could be used to create a version of DEAN finding anomalies on graphs or time series. Both of these have been tried by thesis students (Katzke, 2023; T. N. A. Nguyen, 2024) with mixed success, and another one (Häusler, 2023) has experimented with using the DEAN model on NLP data.

4.7 Conclusion

In this chapter, we presented the first systematic study of surrogate models for anomaly detection, establishing a framework for the construction of new surrogate models from mathematical functions. To guide the selection of an optimal surrogate, we propose five axioms that such a surrogate anomaly detection algorithm should satisfy. We use these axioms to construct the DEAN ensemble, our optimal surrogate algorithm fulfilling each of these axioms.

We evaluate DEAN thoroughly and show that it performs competitively, particularly excelling over other deep learning-based methods and other surrogates. We also show that its performance is more reproducible and reliable than other surrogates and that its performance is only minorly affected by changes in hyperparameters. Additionally, our axiomatic design makes DEAN an easy-to-use algorithm that can likely be adapted to many different use cases and we will study some of them in Part III of this thesis.

CHAPTER 5

SEAN: SHALLOW ENSEMBLE ANOMALY DETECTION

This Chapter follows: Klüttermann, S., Peka, V., Doebler, P., & Müller, E. (2024). Towards highly efficient anomaly detection for predictive maintenance [Accepted and presented, to appear]. *Proceedings of the 23rd IEEE International Conference on Machine Learning and Applications (ICMLA), Special Session 3: Machine Learning for Predictive Models in Engineering Applications (MLPMEA)*, and is partially based on the work of a thesis student: Peka, V. (2024). *Anomaly detection using an ensemble with simple sub-models* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/peka.pdf

5.1 Introduction

This chapter proposes a modification of DEAN: While the performance we discussed in Chapter 4 is competitive, its runtime is not. And while it is possible to accelerate DEAN by reducing the number of submodels or by using parallelization, the benefits this can provide are limited. Instead, we remember that abstraction is often used to accelerate models (See Chapter 2.3) and search for a more abstract, even simpler submodel to use.

Considering the bias terms used in methods like DEAN and DeepSVDD, the algorithm's performance does not actually decrease much without them (Klüttermann, 2022). Still following the universal approximation theorem (Z. Lu et al., 2017) and as we show in Figure 4.2 this does limit how complicated functions can be learned by each individual submodel. So, for an ensemble of those having similar performance with and without bias terms, the bias reduction of the ensemble (Gupta et al., 2022) has to be strong enough to replace the lower

individual submodel expressiveness.

Most of the runtime of our DEAN ensemble is taken by the neural network training. So, this chapter studies the following hypothesis: What if we increase the bias of each submodel further by replacing the neural networks of DEAN with simple linear models? While this initially also increases the bias of the submodels (and thus decreases its predictive power by breaking Axiom 5), it will also drastically accelerate our ensemble efficiency. Furthermore, the expressiveness of the whole ensemble should still be lower than that of each individual model because of the property of ensembles to reduce bias. And we will show that with larger ensemble sizes made possible by the drastic speed increases, we are even able to slightly outperform DEAN in both predictive power and accuracy.

5.2 Extremely-fast Anomaly Detection

For some anomaly detection applications, speed is more important than only reaching high accuracy (C. M. I. P. M. Castro, 2014; Mihigo et al., 2022). The methods that are used in such cases are typically simple, often relying on deviations from basic statistical measures like the mean to indicate anomalies. While such approaches are quick, they fall short in performance compared to modern methods that detect deviations from complex normal data. For tasks where safety is critical, more sophisticated yet still fast algorithms are necessary. Shallow methods, such as the Isolation Forest (F. T. Liu et al., 2008), which uses simple random decision trees to isolate anomalies, or CBLOF (Z. He et al., 2003), which clusters normal data and measures deviations from cluster centers, are examples of effective shallow approaches. Due to the low time complexity of the Isolation Forest, it scales well to larger data. CBLOF is relatively efficient but not as fast. Because of the ease of implementation, speed, and quality, both are exceptional benchmark methods for ultra-fast AD. However, partially due to limited recent research interest in shallow anomaly detection, these methods are often ignored in favor of more advanced learning techniques. In this chapter, we introduce a new shallow method inspired by the DEAN method introduced in the previous chapter, which not only surpasses other shallow methods but also does so in less time.

5.3 Methodology

The primary distinction between DEAN and our in this chapter proposed algorithm, SEAN (Shallow Ensemble ANomaly detection), lies in the replacement of neural networks with linear regression models. In SEAN, each function f_i is a linear regression where the target

value (dependent variable, label) is constantly equal to 1. We observe that the loss function in Equation 4.3 is equivalent to separately fitting mean squared error linear regressions. Let $X_{normal,i}$ denote the feature vectors used for the training of the ith function. The parameters α_i (=coefficients) that minimize the SEAN loss function

$$\min_{\alpha_i} \sum_{x \in X_{normal,i}} (\alpha_i^T x - 1)^2 \tag{5.1}$$

are found by standard techniques for least squares regression. For relatively few features the textbook solution to this optimization problem requires only matrix inversion instead of neural network training:

$$\alpha_i = (X_{normal,i}^T \cdot X_{normal,i})^{-1} \cdot X_{normal,i}^T \cdot \vec{1}, \tag{5.2}$$

where $\vec{1}$ denotes a vector of ones. Equation 5.2 is numerically unstable, so singular value decomposition (SVD) is often preferred.

We added model indices i to the training data, as when the features $X_{normal,i}$ would be identical for all i, the process would be deterministic. So instead, we apply feature bagging (Lazarevic & Kumar, 2005), similar to DEAN, to ensure diversity among the α_i values. Additionally, we incorporate subsampling (Zimek et al., 2013), training each model on a random subset of samples, further increasing variance by training each submodel on its own slightly different dataset. Both feature bagging and subsampling not only enhance model diversity but also reduce computation time. Feature bagging decreases the number of features used in Equation 5.2, while subsampling reduces the number of samples. Given the complexity of $O(samples \cdot features^2)$ in Equation 5.2, these techniques significantly lower computational costs. Additionally, by effectively limiting both the number of training samples used and the number of features used by each model, the runtime of SEAN scales well to large datasets.

To further improve performance with only a few submodels, we introduce a variation of feature bagging where instead of selecting a random subset of bag features for a submodel, we multiply the training data matrix (size $samples \times features$) by a matrix F (size $features \times bag$), where each element of F is randomly drawn from a normal distribution with mean zero and variance one, as shown in Equation 5.3. For this, we are inspired by reservoir computing (H. Zhang & Vargas, 2023), where a similar approach is used to allow simple models to learn more complicated functions. Because we have seen that variations in the average value of F can affect the model, especially for low features, we normalize F, by subtracting the mean of each row from F and thus setting the mean to be exactly 0 for

each generated feature.

$$X_{train} \rightarrow X_{train} \cdot F \tag{5.3}$$

We will explore the impact of different choices for F in our ablation studies in Section 5.4.2.

Like DEAN, SEAN identifies anomalies using an anomaly score, as defined in Equation 4.2. To further enhance speed and reduce memory consumption, we replace the summation with a maximum operator. This is also supported by Chapter 3, showing that this can outperform averages.

$$score(x) = \max_i |\alpha_i \cdot x - 1| \tag{5.4}$$

5.3.1 Axiom compliance

Comparing this approach to the axioms we propose in Section 4.3.1, SEAN works with an equivalent loss function to DEAN and thus also fulfills Axiom 5. Axiom 2 requesting a reliable training procedure is fulfilled since the submodel training process is deterministic. Similarly, since we do not use bias terms, there is no trivial solution, fulfilling Axiom 3. Hyperparameter invariance, as demanded by Axiom 4, will still be tested experimentally in Section 5.4.2, but generally, shallow methods contain fewer hyperparameters than deep learning algorithms. Thus, the only drawback of SEAN is that it does not fulfill Axiom 5 demanding expressive submodels, and thus will likely have a lower submodel performance, especially for complicated datasets. Our hypothesis in this chapter is that a larger submodel count might still increase the expressiveness of the ensemble to a point where it cancels out the lower submodel expressiveness. This hypothesis is analysed in Section 5.4.4.

5.3.2 Hyperparameters

In the following sections, we evaluate SEAN using a specific set of hyperparameters, which represent a balanced trade-off between performance and runtime. Additional hyperparameters are discussed in Section 5.4.2.

First, similar to DEAN, each feature is normalized so that $\max(X_{normal}^{feature}) = 1$ and $\min(X_{normal}^{feature}) = 0$. The SEAN ensemble consists of 50 submodels, each trained on 1000 randomly drawn samples and using $bag = 10\%$ of the available features. If the number of features is too high or too low, we clip bag to ensure it remains within the range $2 \leq bag \leq 100$. Similarly when fewer than 1000 normal samples are available, we use all available samples.

5.4 Experiments

In this section, we comprehensively evaluate and benchmark SEAN, adhering to the methodology of a recent anomaly detection benchmark study, ADBench (Han et al., 2022) equivalent to the evaluation done in Section 4.5. This means we conduct comparisons across the 121 datasets recommended by ADBench, including 57 uncorrelated datasets. We compare SEAN against various shallow anomaly detection algorithms benchmarked in ADBench, such as distance-based algorithms (e.g., KNN (Ramaswamy et al., 2000) and LOF (Breunig et al., 2000)), density-based algorithms (e.g., COPOD (Z. Li et al., 2020), HBOS (Goldstein & Dengel, 2012), and ECOD (Z. Li et al., 2022)), and other approaches like OCSVM (Bounsiar & Madden, 2014), LODA (Pevný, 2016), PCA (Callegari et al., 2011), IForest (F. T. Liu et al., 2008), and CBLOF (Z. He et al., 2003)). Given the inspiration and similarities in methodology, we also compare SEAN with DEAN.

In contrast to ADBench, we assume a scenario where a set of uncontaminated samples is available (weakly supervised setting).For performance evaluation, we only use the ROC-AUC metric (Bradley, 1997; Hanley & Mcneil, 1982), which is the most widely adopted metric in anomaly detection research. Although other metrics like AUC-PR (Boyd et al., 2013) or F1-Score (Goutte & Gaussier, 2005) were studied, results were vastly consistent with ROC-AUC, and we omit them here for brevity.

Moreover, we measure the runtime of each algorithm, restricting evaluations to CPU cores to avoid unfair advantages from GPU-supported implementations. Each evaluation is repeated 10 times, and the results are averaged to ensure robustness.

5.4.1 Performance Comparison

We begin by analyzing the runtime-performance trade-off for each algorithm, as illustrated in Figure 5.1. SEAN demonstrates slightly lower performance compared to DEAN but operates over 20,000 times faster, making it the fastest among all algorithms studied. SEAN outperforms all shallow algorithms, except for distance-based methods. However, distance-based algorithms are known to suffer from the curse of dimensionality (Bellman, 1966), limiting their applicability to high-dimensional real-world data, despite their success in low-dimensional benchmark datasets. Additionally, these algorithms typically require calculating the distance between each test and each training sample, and thus, their runtime does not scale well to large datasets.

Further performance analysis is provided in the critical difference plot in Figure 5.2. The plot reveals no significant performance differences between SEAN and other effective

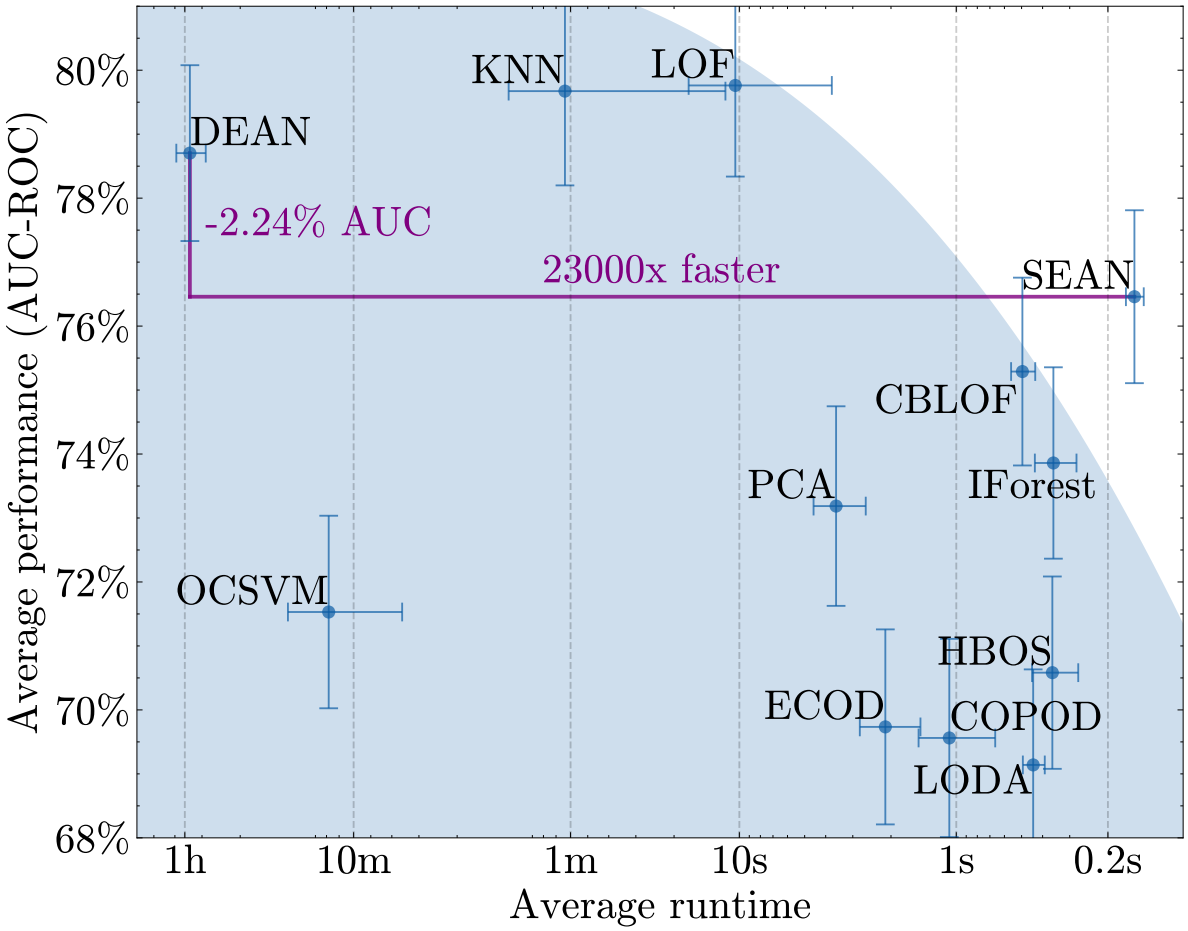


Figure 5.1: Runtime (x-axis, logarithmic) and anomaly detection performance (y-axis, in ROC-AUC) averaged together with their uncertainty across all datasets. An optimal algorithm would be positioned in the top right corner. The blue area indicates the trade-off between speed and performance, suggesting a limit to how fast an algorithm can be while maintaining high performance.

competitors, such as DEAN, CBLOF, IForest, and PCA, highlighting that anomaly detection performance alone may not need to be the most decisive factor in algorithm selection.

We also compare the average runtime required for training and inference by each algorithm, as shown in Figure 5.3. SEAN emerges as the most efficient, with the ability to train nearly 10 ensembles per second on an average dataset. Its inference cost is similarly low, confirming SEAN as the overall fastest algorithm. Notably, even when excluding deep learning algorithms and the relatively slow OCSVM, the runtime differences between algorithms span multiple orders of magnitude, emphasizing the critical importance of an appropriate algorithm choice for real-time applications.

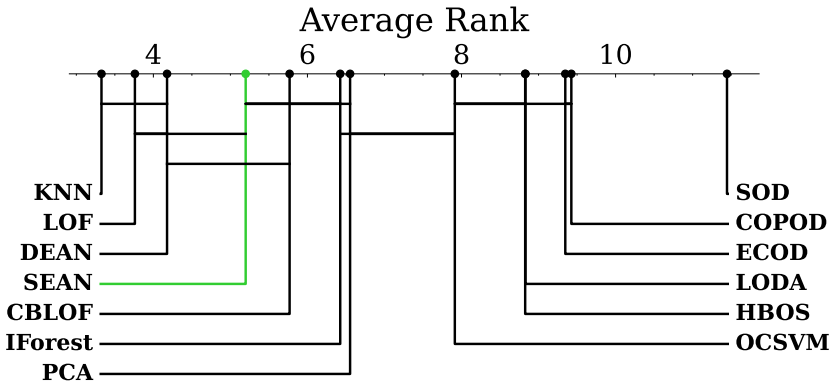


Figure 5.2: Critical difference plot comparing the anomaly detection performance of the algorithms in Figure 5.1. A lower average rank is preferred. A Friedman test (Friedman, 1940) is employed to determine significant differences in ROC-AUC performance, with a Wilcoxon test (Wilcoxon, 1945) connecting algorithms with no significant differences. Significance is determined using p-values below $p \leq 5\%$, after Bonferroni-Holm correction (Holm, 1979).

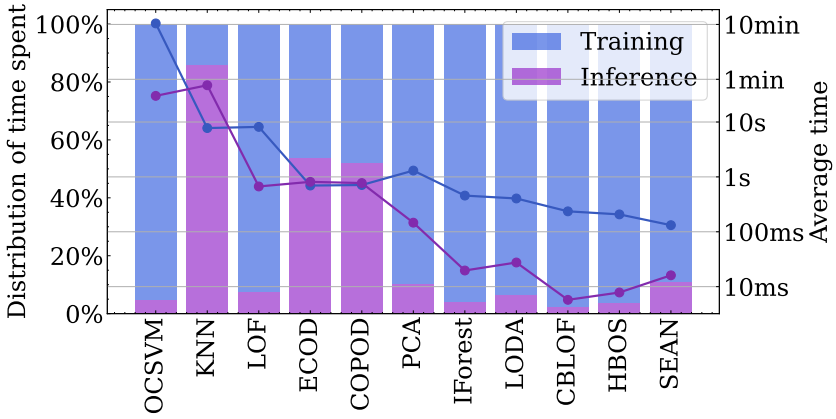


Figure 5.3: Comparison of the runtime of algorithms in Figure 5.1. Training time and inference time are separated, with their relative magnitudes shown in the background.

5.4.2 Ablation Study

We investigate the impact of different hyperparameter choices on SEAN's performance to understand how well SEAN fulfills Axiom 4 in Figure 5.4. Our study reveals that hyperparameters allow for a trade-off between runtime and performance. The most influential hyperparameter is the number of submodels in the ensemble. Increasing the number of submodels by a factor of five approximately multiplies the runtime by five but also reduces the performance gap with DEAN to only about 0.6%, while still maintaining a significant runtime advantage.

We also assess different feature bagging techniques in Table 5.1. Our method of multi-

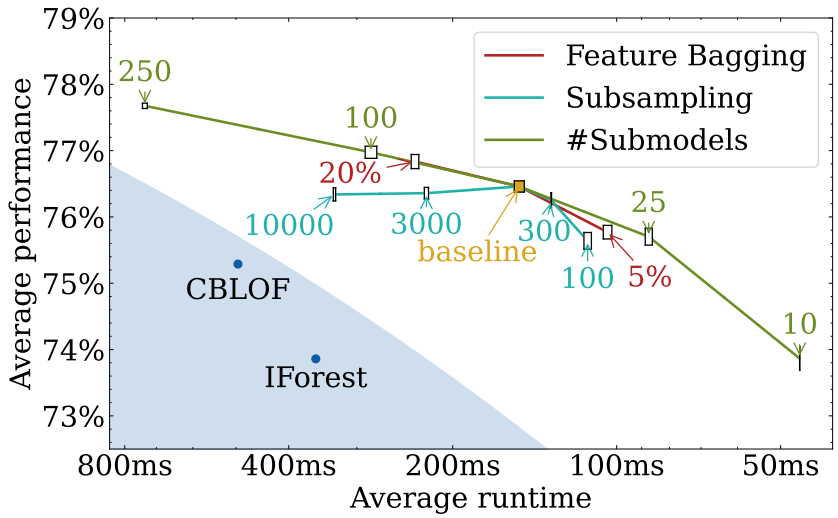


Figure 5.4: Runtime and anomaly detection performance for various hyperparameter combinations. Higher values generally increase runtime. Feature bagging is evaluated with 20% and 5% of features (compared to 10% for the baseline case). Models are trained with 100, 300, 3000, and 10000 samples each (compared to 1000), and SEAN ensembles are formed with 10, 25, 100, and 250 submodels (compared to 50).

plying by a random Gaussian matrix with guaranteed zero mean ("baseline") consistently yields the best performance. If we remove our normalization, guaranteeing that the mean of each matrix row is zero ("baseline no normalization"), the performance is slightly lower but still higher than that of every other bagging method. The "mixture" method no longer draws samples from a normal distribution for the values in the matrix F but draws either a 0 or a 1 from a binomial distribution ($p = 0.5$) for them. It achieves slightly lower performance but offers a marginally faster runtime. The "classical" feature bagging approach (Lazarevic & Kumar, 2005), where each sample is randomly assigned a set of features, underperforms. We believe this to be because of the limited number of fitted submodels, which increases the probability of features being entirely omitted by the ensemble. Alternatively, we suggest the "reverse classical" method, which guarantees that each input sample is assigned and added to exactly one output sample for each submodel. This improves the performance compared to usual feature bagging, but does so at the cost of higher runtime and also without surpassing the baseline approach.

5.4.3 C++ Export

Many real-time anomaly detection tasks are executed on-device (Mihigo et al., 2022), necessitating not only fast models but also minimal memory usage. These environments of-

Table 5.1: Comparison of different feature bagging methods.

Algorithm	Performance (AUC)	Time (s)
baseline	76.46% ± 0.08%	0.1510 ± 0.0032
baseline no normalization	75.47% ± 0.08%	0.1437 ± 0.0026
mixture	74.42% ± 0.05%	0.1172 ± 0.0014
classical	72.40% ± 0.14%	0.1319 ± 0.0031
reverse classical	74.16% ± 0.05%	0.2522 ± 0.0092

ten cannot support high-level languages like Python. To address this, we developed a C++ conversion tool that transforms a trained SEAN model into a compact C++ implementation for on-device anomaly detection. We tested this tool on the largest dataset in ADBench, the Census dataset, which contains 37136 test samples with 500 features each. The resulting executable, including all weights, is less than 100 KB in size and requires under 4 MB of RAM to execute, proving the versatility of our approach.

5.4.4 Individual model performance or submodel speed

We have shown in Figure 5.4 that the number of submodels has a large influence on the performance. This is somewhat expected since a larger ensemble decreases the bias of a model further and thus somewhat cancels out the larger bias of simpler submodels. But this creates another question: Our SEAN model is only slightly (2.24%, see Figure 5.1) worse than the original DEAN model, but more than 20000 times faster when using 50 submodels. Since the runtime of SEAN grows almost perfectly proportionally to the number of submodels, this means we could train an ensemble of over one million SEAN submodels and still be faster than the DEAN model. In Figure 5.5, we investigate what the performance increase of such an extremely large ensemble would be.

However, training $1M$ submodels still requires $1M$ sets of learned parameters and random rotation matrices, and thus, we hit the RAM limitation of the machine we were experimenting on ($512GB$). While we could extend this with a batch-wise implementation, we also already see that with $5-250k$ models, we outperform DEAN while being faster than it. Additionally, a quadratic function seems to describe the relationship quite well, with a $\frac{\chi\check{s}}{ddof} \approx 1.73$, allowing for easy estimation of large ensemble effects.

Overall, it shows that starting from about $5k$ submodels (100× more than in Figure 5.1), SEAN is able to outperform DEAN while still being about 230× faster than the same.

This shows that the bias reduction of the ensemble is more valuable than having initially low-bias submodels, validating our approach to simple abstract ensembles and implying

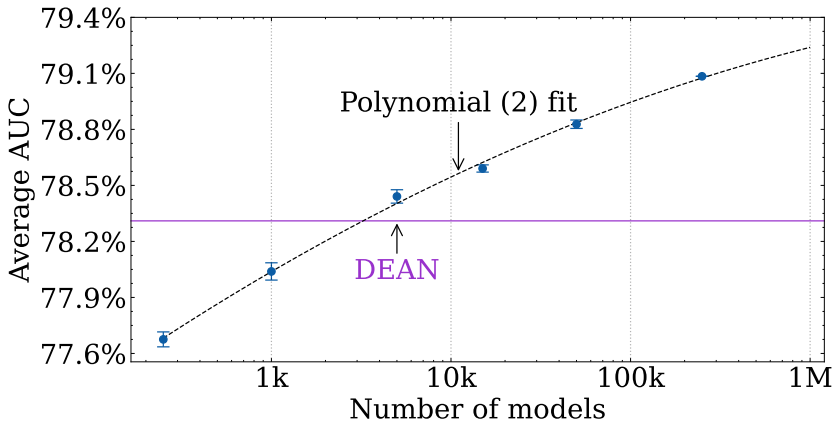


Figure 5.5: Average anomaly detection performance of the SEAN ensemble with extreme numbers of submodels. We investigate up to $250k$ submodels here because of RAM limitations for certain large datasets. This largest ensemble shown here requires more than four times less computation time than the DEAN ensemble introduced in Chapter 4. The uncertainties are calculated by repeating each experiment 10 times.

that Axiom 5 might not be necessary for large ensembles. It also implies that maybe we can create even larger ensembles of even simpler, more abstract ensemble submodels and still reach competitive predictive power.

5.5 Conclusion

This chapter presents SEAN as a highly efficient and effective anomaly detection algorithm. Through extensive benchmarking against 121 datasets, SEAN has demonstrated superior speed, outperforming both shallow and deep learning-based algorithms, particularly in high-dimensional contexts. Our results confirm that SEAN achieves a remarkable balance between performance and computational efficiency, making it ideally suited for time-critical applications. The ablation study further underscores the flexibility of SEAN, showing how its performance and efficiency can be fine-tuned through hyperparameter adjustments. Finally, the ability to export SEAN models to C++ for on-device execution ensures that it meets the stringent requirements of memory and processing constraints typical of industrial environments. SEAN stands out as a robust, scalable, and practical solution for anomaly detection, with significant implications for advancing applications like predictive maintenance and can even outperform DEAN, its original inspiration.

While SEAN has shown strong performance, several avenues for further research are apparent. First, studying the feature bagging process further has shown potential to improve detection accuracy even further.

Expanding SEAN's capabilities to handle streaming data is another promising direction, allowing for more effective real-time anomaly detection in continuously changing environments by also considering previous time steps. But this also implies further challenges in how to intelligently choose features to balance higher resource consumption and performance, as well as how to handle effects like feature drift.

Additionally, optimizing the C++ export tool for deployment on low-resource devices remains an important goal. Enhancing this tool could involve leveraging more efficient libraries or hardware-specific optimizations to better support industrial applications.

Finally, our analysis of SEAN has shown that possible submodel complexity (Axiom 5) is not a requirement for a powerful ensemble, and thus, even larger, even simpler submodels could create interesting ensemble methods.

Part III

Using Abstraction

CHAPTER 6

POST-ROBUSTIFYING DEAN BY MODEL SELECTION

This chapter follows: Böing, B., Klüttermann, S., & Müller, E. (2022). Post-robustifying deep anomaly detection ensembles by model selection. *2022 IEEE International Conference on Data Mining (ICDM)*, 861–866. https://doi.org/10.1109/ICDM54844.2022.00098

6.1 Introduction

While researchers have made a lot of progress designing algorithms for increased anomaly detection performance, including our work in the previous chapters, formal verification of these deep anomaly detectors has often been neglected. Yet, especially in safety-critical areas, it is of utmost importance to formally state and prove guarantees about a model's behavior. This need is exacerbated by the discovery of so-called adversarial samples: inputs designed to fool the model into a wrong prediction (Szegedy et al., 2014). For the anomaly detection task, this translates to false-positive samples indistinguishable from normal samples in the train dataset.

From these adversarial samples, a research branch defending against such attacks has emerged. Methods such as adversarial training (I. J. Goodfellow et al., 2015; Madry et al., 2018) attempt to make neural networks more robust against adversarial samples by adjusting the model's training. If however these methods yield a non-robust model, there is no method to fix it, i.e. robustify it as a post-processing step.

We address the aforementioned challenges by using the inherent properties of our abstract ensemble method. Given an already trained ensemble model, we first assess its robustness

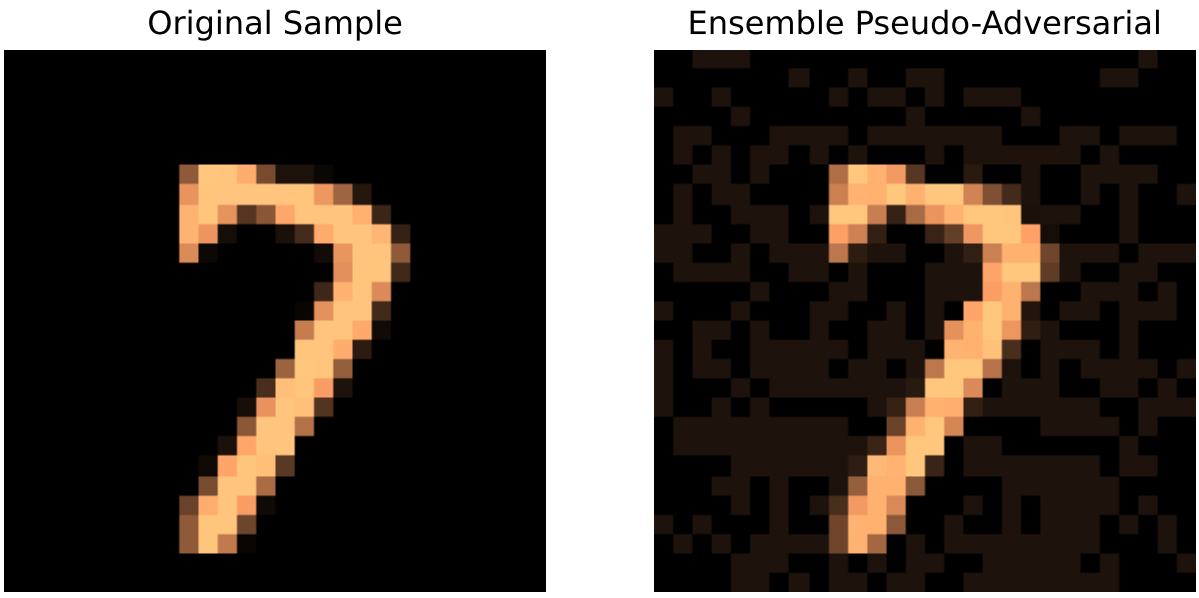


Figure 6.1: Original MNIST sample (left) and its adversarial. Differences between the original and the adversarial have been enlarged to make them visible. The original sample is predicted as normal. The pseudo-adversarial *p-adv* on the other hand is predicted as anomalous even though it is nearly indistinguishable from the original.

by a divide-and-conquer approach. By splitting it up into its submodels, solving a verification problem for each submodel and merging the intermediate results, we obtain an upper and a lower bound for the largest anomaly score in a predefined region.

Moreover, the intermediate results allow us to distinguish between submodels that either do or do not harm the ensemble's robustness. In a second step, we can create a new, robust ensemble model using only the non-problematic submodels. Thus we obtain a post-processing method that - within certain bounds - robustifies any such ensemble to a desired degree.

Beyond post-robustification, our method is also the first to produce a so-called pseudo-adversarial for such an ensemble method, as shown in Figure 6.1. These are inputs to the ensemble that are most likely being predicted as anomalous by the ensemble model and narrow down the approximation gap between the upper and lower bound of the anomaly score.

Using ensemble methods alleviates another major issue of neural network's formal verification: scalability. As shown by (Xiao et al., 2019), the runtime of a verification method increases exponentially with the complexity of neural networks. However, the use of abstract ensemble methods allows us to elegantly circumvent this problem as it is much faster to

verify many small neural networks compared to one large neural network.
We highlight the use of our robustification method on the *DEAN* ensemble method as described in Chapter 4. This model combines several advantageous properties: since it employs feature bagging with a large set of simple submodels, we can scale up the verification to datasets of any dimension. Moreover, due to its simple architecture, any particular submodel can be verified quickly and without approximation loss by an SMT solver.
With our experiments, we highlight several properties of post-robustification. As a first result, we show that we can successfully robustify a given *DEAN* ensemble without impairing its predictive performance. Moreover, we show that any local robustification guarantees not only local robustness but also increases overall robustness for all samples. Therefore, we can even robustify multiple samples simultaneously while still retaining a large portion of the original submodels.
We furthermore notice a surprisingly small approximation gap we inevitably introduce by our divide-and-conquer approach. Even though we do not jointly analyze the submodels, the relative error between our approximation and the actual largest anomaly score never exceeds 2%.
Finally, we compare the robustness of the *DEAN* model to other, well-known deep anomaly detectors. As a result, we strongly suggest using *DEAN* for verifiable anomaly detection because it yields the best performance both in terms of scalability and robustness.

6.2 Related Work

6.2.1 Neural Network Verification

Formal verification of neural networks can be categorized into SMT solver approaches ((Ehlers, 2017; Katz et al., 2017)) and abstract-interpretation-based approaches ((G. Singh et al., 2018; Weng et al., 2018; H. Zhang et al., 2018)). For our purposes, we will make use of SMT solvers as this allows us to obtain two types of results for a predefined region: an upper bound on the anomaly score as well as a lower bound derived from a pseudo-adversarial. Thus we can estimate the approximation gap to the true largest anomaly score in that region. Abstract-interpretation based approaches on the other hand, could only provide us with an upper bound and do not yield a pseudo-adversarial for ensembles.
However, SMT solvers are notorious for being much slower than abstract-interpretation-based methods. We counteract this deficiency through the feature bagging and limiting the

complexity of each DEAN submodel. Overall this leads to an acceptable runtime while producing well-approximated robustness results.

6.2.2 Robustifying against Adversarials

Other approaches to defend against adversarial attacks have already been proposed. Most notably, seminal papers by Madry et al. (Madry et al., 2018) and Goodfellow et al. (I. J. Goodfellow et al., 2015) introduce the technique of adversarial training. However, they assert that their method is only based on testing against adversarials instead of verifying a predefined region. Moreover, it has been shown that any attempt to prevent adversarial attacks by hiding the network structure through, e.g., gradient masking, is doomed to fail because blackbox adversarial attacks suffice (Papernot et al., 2017), (Athalye et al., 2018). Therefore networks must be made provably resilient against adversarial attacks.
Most existing methods, including (Croce et al., 2019; I. J. Goodfellow et al., 2015; Madry et al., 2018; H. Zhang et al., 2019), try to incorporate robustness into their training methods. However, if this training does not yield a robust model, one can only retrain it without the guarantee of obtaining a robust model thereafter. Instead, we aim to adjust an already trained network to become provably robust.

6.2.3 Model Selection/Simplification

Other model selection methods exist that focus on improving performance (Chiang et al., 2016) or try to increase the importance of bounding points (Z. Zhao, 2017). We, on the other hand, select submodels based on the notion of robustness and verification. Beyond model selection for ensemble methods, network pruning (Blalock et al., 2020) is another approach to simplify networks: it reduces model complexity and makes them applicable with low computational costs by erasing weights/neurons. However, none of these approaches aim to robustify the network in the sense we propose.

6.3 Problem setting

There are several notions of robustness both for deep anomaly detectors (Najari et al., 2021; C. Zhou & Paffenroth, 2017) and for neural networks in general (S. Zheng et al., 2016). While a lot of emphasis has been put on training models to be robust, once a model exists it can only be shown or measured whether it is robust. However, given a non-robust model it would be useful to just slightly adapt it in order to make it robust, instead of retraining a

new model from scratch.
Therefore this section poses the challenge to post-process a given model such that it becomes robust. To this end, we will formally introduce the post-robustification problem and provide the necessary definitions.

6.3.1 Deep Anomaly Detectors

First, we need to define what we mean by a deep anomaly detector. It consists of a (set of) neural network(s) F and a function G mapping the output of F to an anomaly score. Thus we can define $Anom(x) = G(F(x), x)$.
This score represents the degree of anomalousness for the input x: the larger $Anom(x)$ is, the more likely it is to be an anomaly according to the model. Moreover, to eventually distinguish between normal and abnormal points, a threshold τ needs to be set such that the anomaly detector becomes a binary classifier: if $Anom(x) > \tau$, we consider x to be an anomaly, else we consider it to be normal. For the models we train, we choose τ so that $Anom(x) \leq \tau$ for 80% of training data points.

6.3.2 Post-Robustification

Inspired by the robustness against adversarial samples in the realm of supervised learning, we want to locally post-robustify models for anomaly detection around a given input x^*. More precisely, this chapter addresses the following problem:

Problem 1 (Post-Robustification)**.** *Given an evaluation metric m and a non-robust model-input pair (D, x^*), create a new model D^* such that (D^*, x^*) is robust and $m(D^*) - m(D)$ is maximized.*

This problem definition reflects that we do not only want to robustify our model on x^*, but that we also do not want to trade off too severely with respect to a given evaluation metric. Otherwise post-robustification might result in a degenerate model that, e.g., maps all inputs to the same output: a model that is very robust but not at all useful.
Note that, assuming a monotone m, we do not want to minimize $|m(D^*) - m(D)|$: if D^* performs even better according to the evaluation metric, we do not consider this a problem.
To make this problem statement more concrete, we need to give the model, the evaluation metric and define a precise notion of robustness: we choose the previously defined *DEAN*

model evaluated by the AUC score which reflects the predictive capability in the task of anomaly detection. The precise definition of robustness will be given in the next section.

6.3.3 Adversarial Robustness

Our robustness definition is the direct adaptation of adversarial robustness from supervised learning to anomaly detection. In essence, we consider the anomaly detector as a binary classifier and apply the definition of (Madry et al., 2018) to it.

Defnition 1 (ε-adv-rob)**.** *Let D be a deep anomaly detector, $dist$ a distance function and x^* a normal sample, i.e. $Anom(x^*) < \tau$. We say that D is ε-adversarial-robust at x^* if and only if for all inputs in $x^* \pm \varepsilon := \{x \in \mathbb{R}^n : dist(x, x^*) \leq \varepsilon\}$ the **L**argest **A**nomaly **S**core is less than τ:*

$$LAS(D, x^* \pm \varepsilon) := \max_{x \in x^* \pm \varepsilon} Anom(x) < \tau$$

According to this definition, a normal input x^* is robust if and only if all surrounding inputs are normal as well. Thus, if we can prove ε-adversarial-robustness, we know that there cannot be an adversarial sample in $x^* \pm \varepsilon$.
Typically, $dist$ will be given by an L_p distance such as L_1 or L_∞ and for the remainder of this chapter, we will work with the L_∞ distance.
Please note that this is a much stronger notion of robustness than e.g. testing against a finite set of adversarial samples. In contrast, we aim to verify the model against infinitely many points defined by the ε environment of x^*. It corresponds to the same notion of robustness against adversarial attacks in supervised learning (Madry et al., 2018).
We have now defined the problem we want to solve for the *DEAN* model. However, we need a further subproblem which we will solve for each submodel of *DEAN* to establish robustness on a given *DEAN* model.

6.3.4 Worst-Case-Error Problem

Given that *DEAN*, among other deep anomaly detectors, bases its anomaly score on the deviation to a desired output, we introduce a slightly adapted version of the worst-case-error problem given in (Böing et al., 2020). If we know the worst-case error of a given submodel in the region of interest $x^* \pm \varepsilon$, we also know whether the model will ever predict *anomalous* for any input in that region.

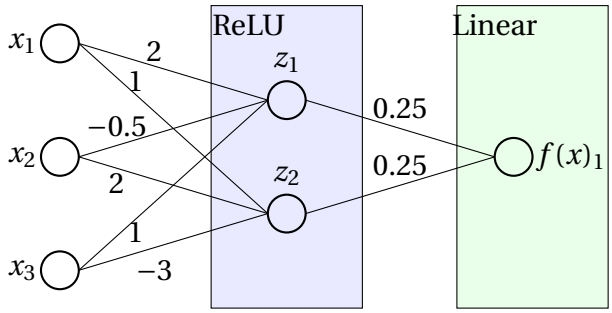


Figure 6.2: Exemplary *DEAN* submodel for explaining the construction of SMT formulas.

Defnition 2 (wce)**.** *Let $f : \mathbb{R}^b \to \mathbb{R}^1$ be a DEAN submodel, $A \subset \mathbb{R}^b$ an input region and q the target value of f. The worst-case-error is given by:*

$$wce(f, A) = \sup_{x \in A} \|q - f(x)\|_\infty$$

Moreover we want to calculate an input that realizes wce up to a predefined accuracy. Sticking to the notation introduced in (Böing et al., 2020) these are so-called *weakly supervised adversarials.*

Defnition 3 (w-adv)**.** *In the context of definition 2 we define an weakly supervised adversarial (denoted w-adv) as an input x such that*

$$\left| \|q - f(x)\|_\infty - wce(f, A) \right| < acc$$

where acc is a predefined accuracy.

Now that we have all the required definitions, we can build our way backwards to solve each of these problems. This is the content of the next section.

6.4 Solution Framework

In this section, we provide a framework for post-robustifying a deep anomaly detection ensemble. Reversing the structure of the previous section, we start by calculating wce on a *DEAN* submodel followed by verifying the *DEAN* ensemble and finally robustifying it.

6.4.1 Calculating wce for a DEAN submodel

To calculate wce for a given *DEAN* submodel we will slightly adapt the solution provided in (Böing et al., 2020) and make use of so called SMT solvers. In essence $wce(x^*, x^* \pm \varepsilon)$ is

obtained by repeatedly checking existence of an input x in $x^* \pm \varepsilon$ such that $\|f(x) - q\| > \delta$. Using this δ-check as a subroutine we can approximate $wce(x^*, x^* \pm \varepsilon)$ with a predefined accuracy using binary search over δ. Therefore, in the following, we only need to describe how to use an SMT solver for a particular value of δ.

While an SMT solver can be used in a very versatile way, we will treat it mostly as a black-box, explaining only what we need to provide and what in return we obtain as a solution. For a more detailed description of SMT solvers, please refer to (Böing et al., 2020) or (Ehlers, 2017).

For our specific use case, we will provide a boolean combination ($\wedge$, $\vee$, $\neg$) of linear inequalities, a so-called 'formula', as input to our SMT solver of choice ((Katz et al., 2019)). These linear inequalities will encode the *DEAN* submodel as well as the δ-check. The variables in each inequality will be given by the submodel's inputs/outputs, as well as the intermediate results in each layer.

As a result of the verification process we will obtain either $unsat$ proving that no input resulting in a distance to q exceeding δ exists, or sat together with a sample $x \in x^* \pm \varepsilon$ such that $\|q - f(x)\|_\infty > \delta$. We will keep track of the last sample obtained during the binary search as this will be the *w-adv*.

A small drawback of this approach is that SMT solvers allow only for piecewise affine functions. Therefore the submodels can consist only of linear, ReLU, convolutional and pooling layers. However, while this limits the generality of this approach, *DEAN* models only require linear and ReLU layers.

To make things more precise, consider a *DEAN* submodel $f = h_1 \circ h_2 \circ \ldots \circ h_L$ where each h_i denotes one layer. Thus f is computed recursively for all layers $k \in 1, \ldots, L-1$ via

$$x_k = h_k(x_{k-1}) = \mathrm{ReLU}(A_k x_{k-1})$$

where A_k is the weight matrix between the fully connected layers $k-1$ and k. The last layer is calculated similarly, except that no ReLU activation is used.

To translate the submodel to a formula for the SMT solver, each neuron is encoded by adding inequalities to the formula using the $\wedge$-conjunction. For the sake of simplicity, we write equalities as well as 'ite' - short for 'if-then-else' - which can all be implemented using boolean combinations of inequalities. For a Linear neuron $x_{k,j}$ - the jth neuron in the kth layer - the subformula encoding it would be given by:

$$\psi_{k,j} := [x_{k,j} = [\sum_{i=1}^{l_{k-1}} A_{k,i,j} x_{k-1,i}]]$$

where l_k denotes the number of neurons in layer k.
In case of a ReLU neuron we add the following subformula:

$$\psi_{k;j} := \Big[[S_{k;j} = [\sum_{i=1}^{l_{k-1}} A_{k,i,j} x_{k-1,i}]] \\ \wedge [x_{k,j} = ite(S_{k,j} < 0, 0, S_{k,j})] \Big]$$

To encode the entire submodel we simply concatenate all subformulas for all neurons via $\wedge$:

$$\phi_f = \bigwedge_{2 \le k \le L} \bigwedge_{1 \le j \le l_k} \psi_{k,j}$$

Beyond encoding the submodel, we need to encode the δ-check in the language of SMT solvers. They are given by a simple formula encoding the L_∞ distance between q and the submodel's output:

$$\phi_\delta = [f(x) \ge q + \delta] \vee [f(x) \le q - \delta]$$

Finally, we need to encode that the input is in an L_∞ ball of radius ε around the input:

$$\phi_\varepsilon = \bigwedge_{1 \le k \le N} [x_k \le x_k^* + \varepsilon] \wedge [x_k \ge x_k^* - \varepsilon]$$

Basically the formula ϕ_δ is satisfied if and only if the error between $f(x)$ and q is large enough, while ϕ_ε restricts the SMT solver to search for a point x in an $\varepsilon-$environment of x^*. Eventually, the formula presented to the SMT solver is given by $\phi_{tot} = \phi_f \wedge \phi_\varepsilon \wedge \phi_\delta$.
If the SMT solver returns $unsat$ for this formula, we know that for all points in $\{x \in \mathbb{R}^n : \|x - x^*\|_\infty < \varepsilon\}$ the distance between $f(x)$ and q is less than δ. If on the other hand, the solver returns sat and a solution $\tilde{x}$ which satisfies $\|x - x^*\|_{L_\infty} < \varepsilon$ and $\|f(x) - q)\|_{L_\infty} > \delta$, we directly obtain an example resulting in a large deviation.
To further ease understanding we will present a small example: For the simple *DEAN* submodel given in Figure 6.2 the SMT formula looks as follows:

Example 1.

$$\begin{aligned} \phi_f = &(z_1 = 2x_1 - 0.5x_2 + x_3) \\ &\wedge (z_2 = x_1 + 2x_2 - 3x_3) \\ &\wedge (\tilde{z}_1 = ite(z_1 < 0, 0, z_1)) \\ &\wedge (\tilde{z}_2 = ite(z_2 < 0, 0, z_2)) \\ &\wedge (f(x) = 0.25\tilde{z}_1 + 0.25\tilde{z}_2) \end{aligned}$$

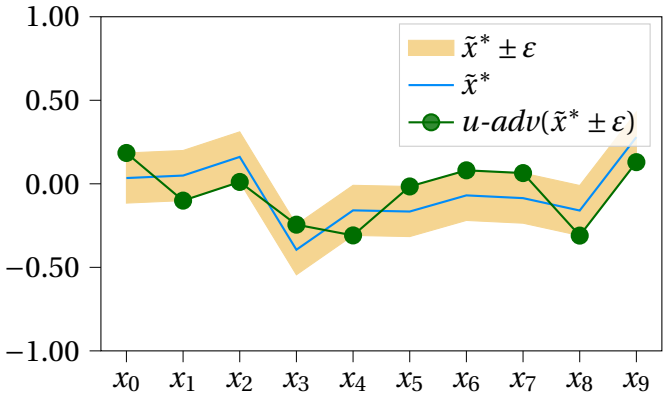


Figure 6.3: Visualization of *u-adv*. For each dimension (x-axis) *u-adv* is usually given by either $\tilde{x}^* - \varepsilon$ or $\tilde{x}^* + \varepsilon$. Thus *u-adv* is a 'corner' of $\tilde{x}^* \pm \varepsilon$

For given $x^* = (1,2,1)$, $f(x^*) = (1)$, *and* $\varepsilon = 0.1$ *we furthermore have a formula*

$$\begin{aligned}\phi_{\varepsilon=0.1} =&((x_1 \geq 0.9) \wedge (x_1 \leq 1.1))\\ &\wedge((x_2 \geq 1.9) \wedge (x_2 \leq 2.1))\\ &\wedge((x_3 \geq 0.9) \wedge (x_3 \leq 1.1))\end{aligned}$$

and lastly for $\delta = 0.1$ *we add*

$$\phi_{\delta=0.1} =((f(x) \geq 1.1) \vee (f(x) \leq 0.9))$$

The solution for this instance is sat *with, e.g., the following variable assignments:*
$\{x_1 = 1.1, x_2 = 2.1, x_3 = 0.9, z_1 = \tilde{z}_1 = 2.05, z_2 = \tilde{z}_2 = 2.6, f(x) = 1.1625\}$
Note, however, that for $\delta = 0.2$, *no solution exists, proving that* wce *must be between* 0.1 *and* 0.2.

6.4.2 Robustness Verification

Having established a procedure to calculate wce on a *DEAN* submodel, we now show how to verify the entire *DEAN* model. The general procedure of our verification framework is given by splitting up the ensemble model, calculating wce and *w-adv* on each submodel, and merging the results. In the following section, we provide details on these steps.

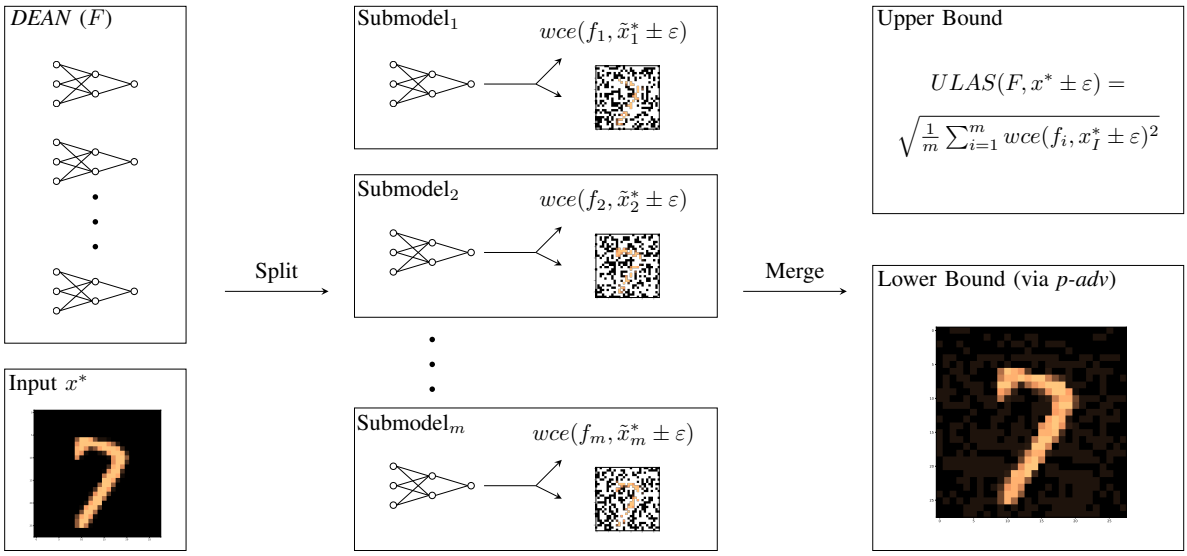


Figure 6.4: Verification Process of the DEAN model. We first split DEAN and the input (left) up into the submodels and their features. Thereafter on each submodel both the wce and the *u-adv* are calculated (middle) followed by merging the results into the upper bound and the pseudo-adversarial *p-adv* (right). Note that due to feature bagging each submodel's *u-adv* does not cover all input dimensions.

6.4.2.1 Splitting

We split the *DEAN* model into each of the submodels it consists of. Thus, if $F = (f_1, ..., f_m)$ is the *DEAN* model we consider each f_i separately in the next step. Note also that for the next step, we need to respect the feature bagging. If we want to check robustness for a given input x^* each submodel f_i is verified on $\tilde{x}_i^* \pm \varepsilon$ being the projection of x^* onto the features of f_i.

6.4.2.2 Submodel Verification

For each submodel f_i we solve the adapted version of the worst-case-error problem posed by (Böing et al., 2020) as described in section 6.4.1. Thereby, we obtain two outputs. On the one hand, we obtain a value wce of each submodel, and on the other hand, an input *w-adv* that realizes wce up to a predefined accuracy.

6.4.2.3 Merging Outputs

From the two outputs obtained for each submodel, we will extract an upper and a lower bound for $LAS(F, x^* \pm \varepsilon)$. Recall that if the upper bound is lower than the anomaly threshold τ we prove local robustness. The lower bound on the other hand is used to estimate the

approximation gap to $LAS(F, x^* \pm \varepsilon)$.
We calculate the **U**pper bound of the **L**argest **A**nomaly **S**core by:

$$ULAS(F, x^* \pm \varepsilon) = \sqrt{\frac{1}{m} \sum_{i=1}^{m} wce(f_i, \tilde{x}_i^* \pm \varepsilon)^2}$$

We replace the error of each model with the largest error that can possibly manifest for each submodel. This is an overapproximation of $LAS(F, x^* \pm \varepsilon)$ because different submodels might realize this largest error on different inputs. However the *DEAN* model must, of course, be evaluated on a single input only. If the upper bound is low enough, it serves as a proof that the ensemble model is robust.
We construct the lower bound by combining the adversarials *w-adv* obtained for each submodel into an input point for *DEAN*. To this end we leverage a property of the adversarials *w-adv* obtained by our subroutine: usually they are at a corner of the input space $\tilde{x}^* \pm \varepsilon$. Thus in each dimension k they are given by $y_k \in \{\tilde{x}_k^* + \varepsilon, \tilde{x}_k^* - \varepsilon\}$ (c.f. Figure 6.3).
Taking the perspective of a particular dimension k, there are several submodels that have this dimension as input due to feature bagging. To combine the adversarials, we simply employ a majority vote among these submodels to determine which side of the corner ($x_k^* + \varepsilon$ or $x_k^* - \varepsilon$) we choose.
Thus, let J be the index set of submodels using feature k as input and $\{y_k^j : j \in J\}$ be the corner points obtained for dimension k by their respective weakly supervised adversarials. Then, we construct a pseudo-adversarial for the ensemble as

$$p\text{-}adv(x^* \pm \varepsilon)_k = \text{mode}\{y_k^j : j \in J\}$$

From this pseudo-adversarial we extract a lower bound for $LAS(D, x^* \pm \varepsilon)$ by simply calculating $Anom(p\text{-}adv(x^* \pm \varepsilon))$. Essentially, we try to combine the adversarials of each submodel in such a way that each of the submodel's errors become large, thereby tailoring a pseudo-adversarial for the ensemble model.
Even though this results in a lower bound for the *DEAN* model, we will experimentally show that it is close to the upper bound. Thus we achieve a small approximation gap.

6.4.2.4 Properties of Ensemble Verification

Robustness verification of an ensemble comes with both advantages and caveats. One major advantage is with respect to scalability. Using an ensemble of simpler models allows us to break down the difficult problem of verifying a large neural network into smaller subproblems, each of which can be solved in a much shorter time. Since verification time increases exponentially in the number of ReLU nodes (Xiao et al., 2019), the size of networks

Algorithm 1: Robustify

Input: $L = \left((f_i, wce(f_i, \tilde{x}_i^* \pm \varepsilon))\right)_{i=1,\dots,m}, \tau$

$L_{sort} = sort(L)$ by wce
while $\sqrt{\frac{1}{|L_{sort}|}\sum_{f_i \in L_{sort}} wce(f_i, \tilde{x}_i^* \pm \varepsilon)^2} > \tau$ **do**
  $L_{sort} = L_{sort} \setminus max(L_{sort})$
end
$RemainingModels = models(L_{sort})$
return $RemainingModels$

that can be verified is limited. Thus ensemble methods of simpler models are a natural aid for scalability.
Secondly, feature bagging - a method only applicable to ensemble methods - allows us to scale up the verification procedure to datasets of any dimension.
Finally, as we will see in the experiments, the *DEAN* ensemble is more robust in the previously defined sense than other deep anomaly methods making them the go-to choice for robust anomaly detection.
Still there is a trade-off we take by verifying each of the submodels independently: we only approximate the largest anomaly score on the entire net, because we are not able to consider inputs to different submodels jointly. We will see empirically though that our approximation is very close.

6.4.3 Robustify

Endowed with the capability to determine robustness, we will now present a simple procedure with which we can post-process a non-robust model-input-pair based on each submodel's wce such that it becomes robust: we sort all submodels by their wce and remove them one by one starting with the largest wce until the upper bound on $LAS(F, x^* \pm \varepsilon)$ of the remaining models is below the anomaly threshold τ. This way we can guarantee that in $x^* \pm \varepsilon$ the resulting ensemble will never predict *anomalous* ensuring robustness.
The exact procedure is given in Algorithm 1 and is based on the assumption that there are a few submodels whose wce exceed τ by a substantial margin (c.f. Figure 6.5). In that case, we only need to remove a small portion of the submodels, thus retaining a lot of the ensemble's predictive capability.
This seemingly simple procedure has to be executed with care given the following caveats: first, by reducing the number of models we might impair the predictive capability of the ensemble. We will show experimentally that this trade-off is not severe, but of course this

depends on the level of robustness one wants to achieve. Secondly there is a limit of robustness that we cannot overcome simply given by the smallest worst-case error of any submodel. Thus, if ε is too large (i.e. if we want too much robustness) post-robustification by removing submodels becomes impossible and Algorithm 1 returns an empty list.

6.5 Experiments

This section highlights the use of post-robustification for a given *DEAN* model. We will start by looking into a single ensemble model trained on MNIST, showing what useful results can be obtained with our method. Thereafter we will compare *DEAN* with a) an autoencoder ensemble (*RandNet*) (J. Chen et al., 2017) and b) the *DeepSVDD* (Ruff et al., 2018) model on 8 other real-world datasets highlighting that *DEAN* models are more robust from the beginning, allow for post-robustification on each of these datasets, and - in contrast to RandNet and DeepSVDD - keep a constant runtime across all datasets.

6.5.1 Deep Dive MNIST

Our first experiments are conducted on the MNIST dataset. Here we train an ensemble of 1000 *DEAN* submodels with feature bagging of size 32 to highlight properties of our *Robustify* method on a single ensemble model.

6.5.1.1 Remaining Predictive Capability

Assuming that we started with a powerful predictor, our first experiment addresses whether the model loses its predictive capability by robustifying. As shown in Figure 6.5, after post-robustification, we can keep 856 of the original models and thereby sacrifice almost no AUC. Indeed, we could have deleted more than 50% of the submodels before witnessing a severe drop in AUC score.

6.5.1.2 Approximation of the Largest Anomaly Score

As we cannot directly calculate LAS, we must approximate it. Recall that we obtained an upper bound on the LAS by aggregating over each submodel's wce and a lower bound by combining each submodel's *w-adv* into a pseudo-adversarial for the *DEAN* model. We can use both bounds to test how accurate the approximation of LAS for the *DEAN* ensemble is.

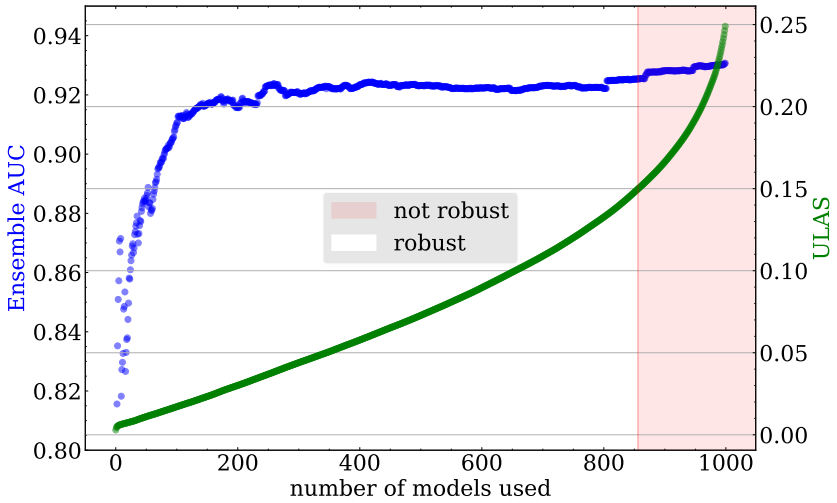


Figure 6.5: ROC-AUC Score and $ULAS$ of the *DEAN* Ensemble as a function of the number of models used to generate it. Here we use only the submodels with the lowest error and consider an ensemble with $ULAS \leq 0.15$ to be robust as defined by τ. Calculating these individual errors does not require labels, so our process is still unsupervised/weakly supervised.

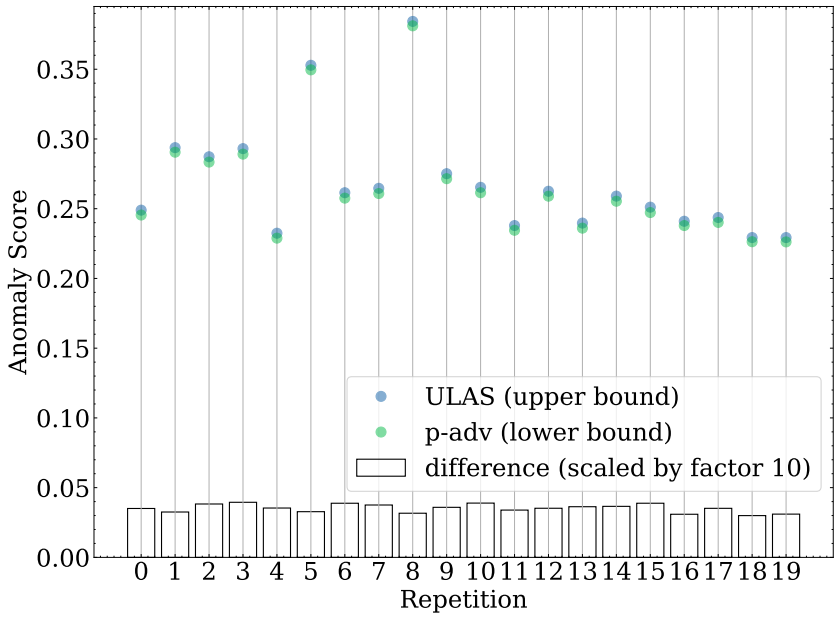


Figure 6.6: LAS Approximations (upper bound through averaging submodel's *wce* and lower bound through combining *u-adv*) for 20 different samples. For every sample tested, we know the value of LAS up to a relative error of less than 2%.

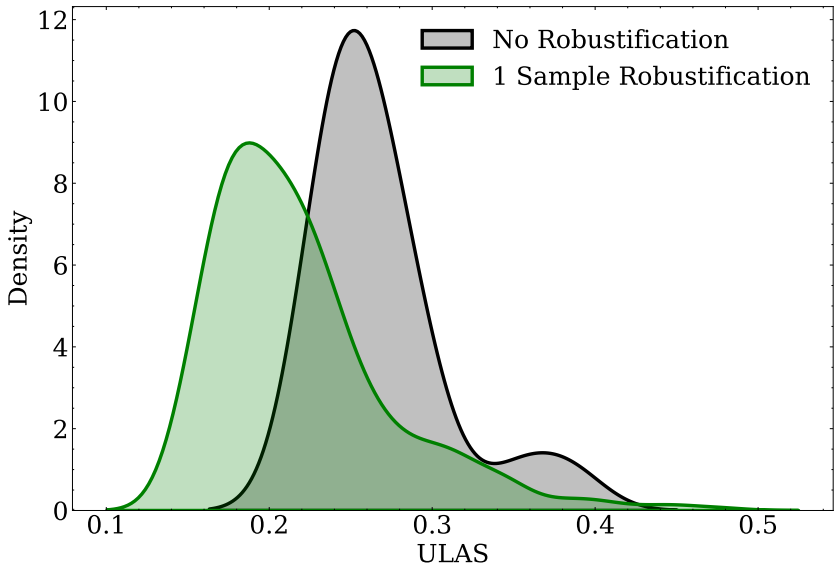


Figure 6.7: Gaussian KDE plot of $ULAS$ of 20 samples before (black) and after (green) application of Robustify. After robustifying on a particular sample we calculated $ULAS$ on different samples to obtain the green density.

Since we have no theoretical guarantee on the size of the approximation gap, we empirically evaluate it for a given *DEAN* model on 20 subsamples in Figure 6.6. Surprisingly, the relative error between the approximation gap and the actual LAS is always less than 2%, showing that our approximation scheme is very precise.

6.5.1.3 Local vs. Global Robustness

Next, we investigate the global effects of local robustification.

Figure 6.7 shows that we significantly decrease $ULAS$ for other normal samples by local robustification. It appears that the models we delete by robustifying one sample also cause high wces on other samples. Therefore, our method increases robustness not only locally but also globally.
However, an important remark is that in order to be provably robust at multiple points, one needs to successively apply the robustifying step in each of which the trade-off with predictive capability is made. As seen in Figure 6.9 though, the trade-off becomes less severe with a growing number of local robustifications.

Dataset	Features	RandNet	DEAN	DeepSVDD
$pageblocks$	10	0.9231	0.9577	0.8748
$segment$	18	0.9982	0.9998	0.9981
$steelplates$	27	0.7521	0.7329	0.718
wbc	30	0.941	0.9751	0.9336
$satellite$	36	0.8321	0.8196	0.8233
$qsarbiodeg$	39	0.8734	0.7425	0.821
$gasdrift$	128	0.9791	0.9319	0.9562
har	561	0.9786	0.9529	0.9371
$Average$		0.9097	0.8890	0.8828

Table 6.1: ROC-AUC Scores on the 8 datasets used in this chapter.

6.5.2 Comparison to RandNet and DeepSVDD

This section compares *DEAN* to two representative, alternative models: *RandNet* (J. Chen et al., 2017) and *DeepSVDD* (Ruff et al., 2018). While *RandNet* consists of an ensemble of autoencoders, thereby being directly comparable to *DEAN*, *DeepSVDD* consists of a single, large neural network. Thus even though we cannot directly apply our post-processing method to *DeepSVDD*, we highlight how *DEAN* outperforms it with respect to runtime and robustness.

6.5.2.1 Datasets

We choose eight different datasets with varying number of features. These are chosen from (Rayana, 2016) and (Vanschoren et al., 2013), such that each algorithm achieves a similar AUC score as shown in Table 6.1 and can thus not be used to compare the algorithm's performance.

6.5.2.2 Verification of RandNet and DeepSVDD

For both models, we need to define how to verify their robustness. To this end we need to upper bound their anomaly score (via $ULAS$) to thereafter compare this bound to τ.
For the *RandNet* model $RN = (ae_1, \ldots, ae_m)$ we proceed similar to the *DEAN* model: we calculate wce for each submodel by

$$wce(ae_i, x^* \pm \varepsilon) = \max_{x \in x^* \pm \varepsilon} \|x - ae_i(x)\|_\infty$$

and join the results using the median as their aggregation function

$$ULAS(RN, x^* \pm \varepsilon) = \underset{i \in 1,\ldots,m}{\text{median}}(wce(ae_i, x^* \pm \varepsilon)).$$

Note that we need to choose the L_∞ norm to calculate wce, because SMT solvers cannot handle the L_2 norm.
For a *DeepSVDD* model DS the anomaly score is given by the euclidian distance (L_2-norm) to a (given) parameter c. Similar to *RandNet* we instead need to calculate the L_∞ norm as wce:

$$wce(DS, x^* \pm \varepsilon) = \max_{x \in x^* \pm \varepsilon} \|DS(x) - c\|_\infty$$

We can ensure robustness only via the largest L_∞-ball fully contained in the $L2$-ball of radius c. Therefore the upper bound of the anomaly score is given by

$$ULAS(DS, x^* \pm \varepsilon) = \sqrt{p \cdot wce(DS, x^* \pm \varepsilon)^2}$$

where p is the model's output dimension.

6.5.2.3 Ratio Comparison

As the anomaly scores of different models can have different scales, Equation 6.1 defines the so-called Change Ratio (CR) to make the results on different models comparable. It indicates how much $ULAS$ needs to be reduced to obtain a robust model: the model is already robust for $CR \leq 1$ while for $CR > 1$ we need to reduce $ULAS$ by a factor CR.

$$CR := \frac{ULAS(D, x^* \pm \varepsilon)}{\tau} \tag{6.1}$$

The upper part of Figure 6.8 shows that for every dataset, *DEAN* is already the most robust, often having a $CR \leq 1$. While *DeepSVDD* is already less robust, *RandNet* has change ratios more than 100 times higher than those of *DEAN*. At least partly, this is due to a larger approximation error of wce as we have to replace the L_2-norm with the L_∞-norm for both *RandNet* and *DeepSVDD*. *DEAN* on the other hand calculates wce precisely showing that it is best suited for our approach.
One measurement, showing CR of the *RandNet* on the *har* dataset (561 Features), is missing. This is because the model's complexity caused the verification to time out.
The bottom part of Figure 6.8 shows the runtime of the verification algorithm. As the x-axis is sorted by the number of features in each dataset, we show the drastic increase in required verification time both for *RandNet* and, to a lesser extent, also for *DeepSVDD*. In contrast,

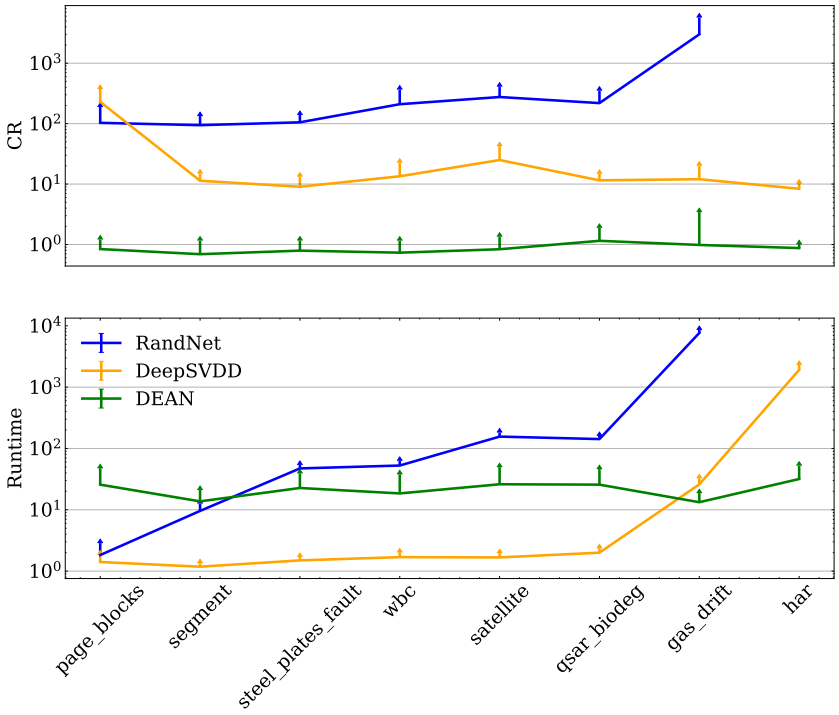


Figure 6.8: Change Ratio and runtime on eight datasets averaged over 10 runs and 20 samples each. The computational effort for *RandNet* and *DeepSVDD* increases with the number of features (note that the datasets are sorted by their number of features). For the highest dimensional dataset "har" with 561 features, it was impossible to verify the autoencoder due to timeout. Notice that in contrast to the other algorithms, DEAN has constant runtime and the lowest Change Ratio.

the required verification time for the *DEAN* model stays constant over all datasets as feature bagging allows to keep the number of ReLU nodes in each submodel the same.
Even though *DEAN* seems to have a larger runtime on lower-dimensional datasets, note that - unlike *DeepSVDD* - its verification process can be parallelized along the submodels. Moreover, it depends linearly on the number of submodels verified. Therefore, the eventual runtime of *DEAN* verification can be controlled both with the number of submodels to be verified and with the number of CPU cores available.

6.5.3 Towards Global Robustness

Finally, we want to build on the results obtained in section 6.5.1.3. As we have seen, local post-robustification impacts robustness globally. Therefore Figure 6.9 shows the number of remaining models after *repeatedly* applying post-robustification on different samples. It appears that on all datasets we can robustify up to 10 different samples without sacrificing

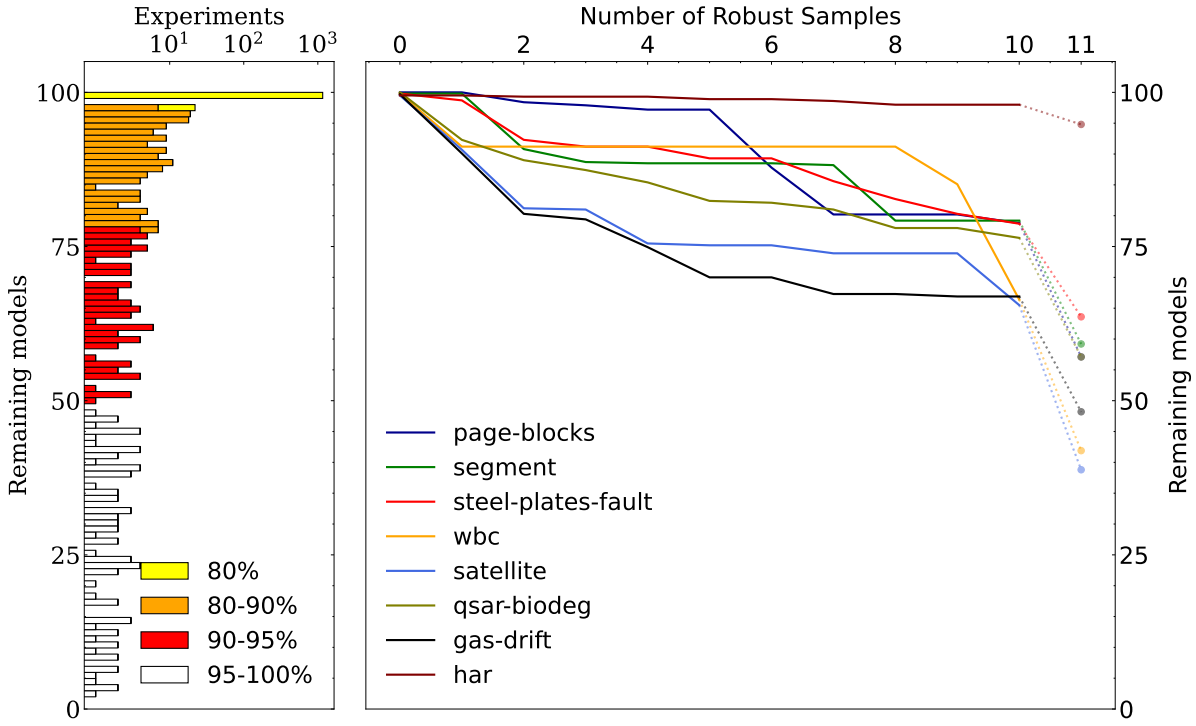


Figure 6.9: Number of models deleted by robustification (left) and number of remaining models after *repeated* robustification. To reach robustness around a single sample, we must remove less than 10% of models on average. As we show on the left side of the figure, for 90% of verified samples, we need to remove at most 22% of submodels. Just a few samples are hard to robustify and require removing more submodels. Using these insights, we test if we can robustify a given ensemble for multiple samples. We show the number of remaining models on the right side after robustifying up to 10 samples averaged over 10 different ensembles and samples. As only a few hard-to-verify samples affect this curve strongly, we choose the 10 samples that are easiest to robustify for every model. However, we show the effect of also including the final sample as point 11.

too many models.

However, there are a few samples that would delete more than 50% of the models, as can be seen on the left-hand side. For these cases, one has to closer evaluate the remaining AUC score, which is why we exclude them in our experiment. In future work, we may also investigate how to train more submodels focusing on these particular inputs.

6.6 Conclusion

In this chapter, we are the first to study the formal verification of ensembles for anomaly detection. We show that especially that an abstract ensemble approach, like the *DEAN* model

seems suitable for this task, as it outperforms common alternatives in terms of robustness. Also, using feature bagging, *DEAN* achieves a verification time independent of the number of features of the dataset. To our knowledge, this is the only algorithm that achieves sub-exponential verification time.
We also introduce *Robustify*, a method allowing to further improve the robustness of the ensemble. We show using *DEAN* that this method allows to drastically increase the local robustness of the model, while maintaining the same anomaly detection capability.
Using *Robustify*, we are able to generate an ensemble that is robust in any area locally. Using repeated application of this algorithm, we achieve robustness in multiple points. However, to achieve global robustness, we require a more powerful algorithm. This could be a method similar to boosting, scaling the impact of less robust submodels differently and thus reaching global safety against adversarial attacks. We would also like to explore how to apply *Robustify* to supervised ensembles and especially to binary classification tasks.

Recently, some people propose using libschitz safe neural networks (Gouk et al., 2021) to guarantee a form of adversarial safety for all points as an alternative to manual verification. Some initial experiments done by a thesis student (Wickes, 2024) show that such an approach can also work in anomaly detection, but also that unsupervised and weakly supervised tasks require special care and also have a reduced performance when using libschitz safe neural networks.

CHAPTER 7

EFFICIENT EXPLANATIONS USING SHAPLEY VALUES

This chapter follows: Klüttermann, S., Balestra, C., & Müller, E. (2024). On the efficient explanation of outlier detection ensembles through shapley values. In D.-N. Yang, X. Xie, V. S. Tseng, J. Pei, J.-W. Huang, & J. C.-W. Lin (Eds.), , *pacific-asia conference on knowledge discovery and data mining (pakdd), advances in knowledge discovery and data mining* (pp. 43–55). Springer Nature Singapore. https://doi.org/https://doi.org/10.1007/978-981-97-2259-4_4

7.1 Introduction

For many applications, simply finding anomalies is not enough, and we would also like to be able to explain why a sample is anomalous. Consider, for example, predictive maintenance, where an explanation might help an engineer fix a fault faster (Langone et al., 2020; Vollert et al., 2021).

Shapley values (Shapley, 1953) have emerged as a promising technique for interpreting the contributions of individual features in black-box models. They offer mathematical guarantees of fairness that make them an attractive choice for anomaly detection as well. However, their practical application poses a significant challenge due to the requirement of training an exponentially large number of models. While significant progress has been made in anomaly detection interpretability (Z. Li et al., 2023), challenges persist. The trade-off between interpretability and model complexity transferred to the computational complexity of the feature importance scores, thus remains an interesting topic of investigation.

In response to this challenge, we propose an innovative approach that leverages attributes of our abstract ensemble approaches to approximate Shapley values efficiently and makes anomaly detection methods based on feature bagging interpretable. First, we delve into the details, defining the *bagged Shapley values* and presenting a theoretical proof of our approach. The experimental results demonstrate our method's effectiveness in achieving efficient interpretability in anomaly detection tasks with complex, high-dimensional data.

7.2 Shapley Values and Intepretability

Shapley values (Shapley, 1953) originate from Cooperative Game Theory. Since their first applications, they gained prominence as a powerful tool for increasing the interpretability of machine learning black-box models (Lundberg & Lee, 2017b; Ribeiro et al., 2016; Strumbelj & Kononenko, 2010). Shapley values offer a theoretically sound framework for quantifying the impact of each feature or factor in a model's prediction; the scores, being the average marginal contribution across all possible feature combinations, are robust and interpretable. Attributing the contributions of the individual features has revealed helpful for anomaly detection (Takahashi & Ishiyama, 2014; Tallón-Ballesteros & Chen, 2020), where Shapley values provide valuable insights into the importance of features in identifying anomalies. However, their practical use is contrasted by one significant challenge, namely their computational complexity. The exact computation of Shapley values requires evaluating a value function for every possible subset of players. Thus, the consequent exponential blow-up in computational cost soon renders their use for high-dimensional contexts infeasible. An exception to this are tree-based models, for which an exact polynomial time algorithm exists (Lundberg & Lee, 2017a).

However, in general, approximation techniques need to be implemented to make Shapley values more accessible; These include Monte Carlo sampling, stratification of players, and kernel approximations (Burgess & Chapman, 2021; J. Castro et al., 2009; Lundberg & Lee, 2017b; Štrumbelj & Kononenko, 2014; van Campen et al., 2018). Each method addresses the efficient computation of Shapley values differently, with potential accuracy and computational cost trade-offs.

Interpreting anomaly detection methods is essential for understanding *why* single data points are considered anomalous, e.g., in safety-critical applications. Feature importance analysis plays an essential role (Z. Li et al., 2023): Techniques such as feature attribution (Dissanayake et al., 2021) are employed to highlight which features have the most significant

impact on the detection. Additionally, we find rule-based models (E. Müller et al., 2012), decision trees (Park & Kim, 2021), and model-agnostic techniques like LIME (Ribeiro et al., 2016) and SHAP (Lundberg & Lee, 2017b) to shed light on the decision-making process of anomaly detection models. Furthermore, visualizations are essential for enhancing trust in complex scenarios. Examples are heatmaps, scatter plots, and time-series representations (Balestra et al., 2023; Kadir & Brady, 2001).

7.3 Anomaly detection ensembles

In our context, a set $X \subseteq \mathbb{R}^N$ of data points can be parted into two subsets: the set of *normal observation* indicated with X_{nor}, and the set of abnormal observations, indicated with X_{abn}. In unlabeled data, distinguishing normal from anomalous data is not always straightforward. We consider a model for anomaly detection f, that aims at classifying each data point $x \in X$ as either *normal* or *anomalous*. Among the various anomaly detection methods, we focus on methods that provide to each data point a score measuring its likelihood of being *anomalous*.

Defnition 4. *Given a set of data points X, we call* model *a function $a : X \mapsto \mathbb{R}$ where $a(x)$ represents the anomaly score assigned by a to the sample x.*

The higher the value $a(x)$, the more likely x is considered to be an anomaly compared to the set X. On the same set X, various anomaly detection models can be constructed. We indicate with $\mathscr{M}_X$ the set of models constructed on X.

Defnition 5 (Ensemble)**.** *Given a set of (sub)models $\mathscr{M}_X$, an ensemble is a function $A_{\mathscr{M}_X} : X \mapsto \mathbb{R}$ that assigns to each $x \in X$ its average anomaly score, i.e.,*

$$A_{\mathscr{M}_X}(x) = \frac{1}{\|\mathscr{M}_X\|} \sum_{a \in \mathscr{M}_X} a(x). \tag{7.1}$$

The ensemble prediction is the average submodel prediction in the set $\mathscr{M}_X$.

Using the trick of projected data points in lower dimensional spaces, we reach the definition of bagging. We indicate with $\mathscr{N}$ the set of coordinates of X and with X_I the set of data points in X projected only on the $I \subseteq \mathscr{N}$ coordinates (or *features*), i.e., given $x \in X$ the corresponding point $x_I = (x_i)_{i \in I}$ and $I \subset \mathscr{N}$. Now we can define a subset $\mathscr{M}_{X_I} \subseteq \mathscr{M}_X$ as the set of submodels that belongs to $\mathscr{M}_X$ trained only on X_I.

The bagging procedure is meant to randomly cover the information in X, considering only the projection of X in smaller-sized subsets. We refer to the size of the data points in

the projection as *bag*. Having $X \subseteq \mathbb{R}^N$ fixed and bag $\leq N$, we can get $\binom{N}{\text{bag}}$ different subsets of size bag from the N features.

Defnition 6 (Bagging)**.** *After fixing the* bag*, the* bagging procedure *consists in defining the model $b_{S,a} \in \mathscr{M}_S$ such that $b_{S,a}(x)$ is the result of a model a when trained on the data set X_S and S is a subset of N whose size is $|S| =$ bag.*

The bagging procedure does not fix either the model a from $\mathscr{M}_X$ or the set $S \subseteq \mathscr{N}$, thus potentially covering, using sufficiently many random seeds, all the information contained in X. We write $b_{S,a|\text{seed}}$ for the specific *bagging submodel* resulting after we fixed the seed for the random sampling of S and the model a. Finally, we can construct the so-called *feature bagging* ensemble based on the bagging technique.

Defnition 7 (Feature Bagging)**.** *Given a dataset X and a set of models $\mathscr{M}_X$, we define the function $f_{\mathscr{M}_X} : X \mapsto \mathbb{R}$ such that it assign to each $x \in X$ the score defined as*

$$f_{\mathscr{M}_X}(x) = \lim_{n\to\infty} \frac{1}{n} \sum_{j=0}^{n} b_{S,a|seed[j]}(x). \tag{7.2}$$

where seed is an eventually infinite vector of randomly drawn seeds.

A similar definition could also be made for non-anomaly detection ensembles as long as the output is a linear combination of the submodel predictions. Still, feature bagging is more commonly used in anomaly detection.

7.4 The bagged Shapley values

A cooperative game is a pair $(\mathscr{N}, v)$ where $\mathscr{N}$ represents the set of *players*, and v is a function over the subsets of $\mathscr{N}$. v assigns to each *coalition* of players a worth, i.e., a positive real number representing the score obtained by the players as a team. Usually, the monotonicity of the value function is assumed, i.e., $v(\mathscr{A}) \leq v(\mathscr{B})$ if $\mathscr{A} \subseteq \mathscr{B} \subseteq \mathscr{N}$.

The Shapley values are a *fair* assignment of weights to the single players that consider the role of the single players in any single coalition. Given a game $(\mathscr{N}, v)$, $\phi_v(i)$ represents the Shapley value of player i:

Defnition 8 (Shapley Value)**.** *Given a game $(\mathscr{N}, v)$, the Shapley value $\phi_v(i)$ of player i is defined as*

$$\phi_v(i) = \sum_{\mathscr{S} \subseteq \mathscr{N}, i \notin \mathscr{S}} \frac{|\mathscr{S}|! \cdot (|\mathscr{N}| - |\mathscr{S}| - 1)!)}{|\mathscr{N}|!} [v(\mathscr{S} \cup \{i\}) - v(\mathscr{S})] \tag{7.3}$$

We refer to $v(\mathcal{S} \cup \{i\}) - v(\mathcal{S})$ as the *marginal contribution of i to $\mathcal{S}$*. Shapley values have a flexible and straightforward definition, depending only on v and the number of players; this made them the object of study in various circumstances and applications. However, their computation results in an NP-hard problem that approximation approaches can only partly solve. We show that the exact computation of Shapley value-similar scores for feature bagging ensembles can be easily reduced to a polynomial time.

We introduce the *bagged Shapley values*; their definition perfectly aligns with the impossibility of training an ensemble method with less than *bag* features. We rewrite the definition of Shapley values from Equation (7.3) $\phi_{f_{\mathcal{M}_X}(x)}(i)$ for feature bagging ensembles, where $x \in X \subseteq \mathbb{R}^N$ is a data point, $f_{\mathcal{M}_X}$ is the feature bagging model and we are interested in assigning to the coordinate i of X an importance score in predicting the overall anomaly score $f_{\mathcal{M}_X}(x)$. We define the bagged Shapley values:

Defnition 9 (bagged Shapley Value)**.** *Given a set of data points $X \subseteq \mathbb{R}^N$, a set of (sub)models $\mathcal{M}_X$ and a feature bagging model $f_{\mathcal{M}_X}$ defined over $\mathcal{M}_X$, the* bagged Shapley values *are the values*

$$\tilde{\phi}_{f_{\mathcal{M}_X}(x)}(i) = \sum_{S \subseteq \mathcal{N}, i \notin S, s \geq bag} \frac{N}{N - bag} \frac{s! \cdot (N - s - 1)!)}{N!} \left[f_{M_{X_{S \cup \{i\}}}}(x) - f_{M_{X_S}}(x) \right] \tag{7.4}$$

This equation removes terms with magnitude $\propto \frac{\text{bag}}{N}$, a necessary step, as defining an ensemble model with less than *bag* features is not possible. Notice that the higher the dimension of the data points in X is, the smaller the difference between $\tilde{\phi}_{f_{\mathcal{M}_X}(x)}(i)$ and $\phi_{f_{\mathcal{M}_X}(x)}(i)$. To somewhat correct for this difference, we add a factor $\frac{N}{N - \text{bag}}$ to compensate that we are summing over fewer subsets of $\mathcal{N}$.

7.5 Theoretical guarantees for the approximation

The main result of our studies regards the chance to express Shapley values with a limited number of selected bagging submodels, thus avoiding the exponential computational costs of Shapley values.

Theorem 1. *The* bagged Shapley values *can be expressed using a selection of submodels involved in the feature bagging ensemble $f_{\mathcal{M}_X}$. In particular, it holds*

$$\tilde{\phi}_{f_{\mathcal{M}_X}(x)}(i) \propto f_{\mathcal{M}_X}(x) - f_{\mathcal{M}_{X_{\mathcal{N} \setminus i}}}(x).$$

Proof. To increase readability, we use the notation

$$k(S, N) = \frac{N}{N - \text{bag}} \frac{s!(N - s - 1)!}{N!}$$

where $s = |S|$ and $N = |\mathcal{N}|$. For abuse of notation and readability, we write S instead of X_S throughout the whole proof.

Now, we can rewrite the *bagged Shapley values* in the following way $b_{S,a|\text{seed}}$ and substitute it with $b_{|\text{seed}} \in \mathcal{M}_S$

$$\begin{aligned}\tilde{\phi}_{f_{\mathcal{M}_X}(x)}(i) &= \sum_{S\subseteq\mathcal{N}, i\notin S, s\geq \text{bag}} k(S,N)\left[f_{S\cup\{i\}}(x) - f_S(x)\right] \\ &= \lim_{n\to\infty} \sum_{S\subseteq\mathcal{N}, i\notin S, s\geq \text{bag}} k(S,N) \\ &\cdot\left(\frac{\sum_{j=0,\dots,n, b\in\mathcal{M}_{S\cup\{i\}}} b_{|\text{seed}}(x)}{\|\mathcal{M}_{S\cup\{i\}}\|} - \frac{\sum_{j=0,\dots,n, b\in\mathcal{M}_S} b_{|\text{seed}}(x)}{\|\mathcal{M}_S\|}\right)\end{aligned}$$

where $\mathcal{M}_K = \{a \in \mathcal{M}_X \mid a \text{ restricted to features in } K\}$ is the subset of models that contain only features included in K.

From the previous equation, we see that $\tilde{\phi}_{f_{\mathcal{M}_X}(x)}(i)$ is a sum over the same bagging models multiple times, as they are part of various subsets. We can simplify the writing to evaluate each model only once but weight them.

$$\tilde{\phi}_{f_{\mathcal{M}_X}(x)}(i) = \lim_{n\to\infty} \frac{1}{\|\mathcal{M}_X\|} \sum_{b\in\mathcal{M}_X} \alpha_b \cdot b_{|\text{seed}}(x) - \frac{1}{\|\mathcal{M}_{\mathcal{N}\setminus i}\|} \sum_{b\in\mathcal{M}_{\mathcal{N}\setminus i}} \beta_b \cdot b_{|\text{seed}}(x) \tag{7.5}$$

Noting that we can shuffle our feature labels without changing Equation 7.5, $\alpha_b = \alpha$ and $\beta_b = \beta$ have to be independent on the specific model $b_{|\text{seed}}$. By the same argument, α and β can not depend on the model outputs $b_{|\text{seed}}(x)$. This allows us to choose any model $b(x)$ to compute them; we pick here

$$b(x) = \begin{cases} 1 \text{ if model } b \text{ considers feature i} \\ 0 \text{ else} \end{cases} . \tag{7.6}$$

Using the proposed $b(x)$, the β term disappears, thus we can write α as:

$$\begin{aligned}\alpha &= \lim_{n\to\infty} \frac{\sum_{S\subseteq\mathcal{N}, i\notin S, |S|\geq \text{bag}} k(S,N) \frac{\|\mathcal{M}_X\|}{\|\mathcal{M}_{X_{S\cup\{i\}}}\|} \sum_{b\in\mathcal{M}_{X_{S\cup\{i\}}}} b(x)}{\sum_{b\in\mathcal{M}_X} b(x)} \\ &= \lim_{n\to\infty} \frac{\sum_{S\subseteq\mathcal{N}, i\notin S, |S|\geq \text{bag}} k(S,N) \cdot \frac{\text{count}(\mathcal{M}_{X_{S\cup\{i\}}})}{\|\mathcal{M}_{X_{S\cup\{i\}}}\|}}{\frac{\text{count}(\mathcal{M}_X)}{\|\mathcal{M}_X\|}}\end{aligned}$$

where count($\mathcal{M}_{X_K}$) is the number of models in $\mathcal{M}_{X_K}$ that contain one specific feature in K. We can use $\lim_{n\to\infty} \frac{\text{count}(\mathcal{M}_{X_K})}{\|\mathcal{M}_{X_K}\|} = \frac{\binom{|K|-1}{\text{bag}-1}}{\binom{|K|}{\text{bag}}} = \frac{\text{bag}}{|K|}$ thus getting

$$\begin{aligned}\alpha &= N\frac{N}{N-\text{bag}}\sum_{s=\text{bag}}^{N-1}\binom{N}{s}\cdot\frac{s!(N-s-1)!}{N!}\cdot\frac{1}{s+1}\\ &= \frac{N}{N-\text{bag}}\sum_{s=\text{bag}}^{N-1}\cdot\frac{1}{s+1}\\ &= \frac{N}{N-\text{bag}}\cdot(\psi^0(N+1)-\psi^0(\text{bag}+1))\end{aligned}$$

with the digamma function ψ^0.

When instead of choosing $b(x)$ to be independent of i, we find that $\tilde{\phi}_{f_{\mathcal{M}_X}(x)}(i) \propto (\alpha-\beta)$. But since the feature is designed not to have any effect, we know that $\tilde{\phi}_{f_{\mathcal{M}_X}(x)}(i) = 0$ and thus $\alpha = \beta$. This concludes the proof.

■

The results not only show that the bagged Shapley value is proportional to the difference of two feature bagging, respectively defined on $\mathcal{M}_X$ and $\mathcal{M}_{X_{\mathcal{N}\setminus i}}$, but also that when using bagging models, we can calculate the bagged Shapley values in polynomial time. This is because for deterministic submodels, instead of using ∞ of them, we only need to train $\binom{N}{bag} < N^{bag}$ submodels.

7.6 Experiments

We evaluate our approach on various freely available real-world datasets with varying numbers of features (Deng, 2012; Z. Liu et al., 2015; Triguero et al., 2017). We conduct experiments on the correctness of the approximation (Section 7.6.1), the effectiveness of our explanation (Section 7.6.2), and the scalability (Section 7.6.3) of our approach.

7.6.1 Quality of the Approximation

To fairly investigate the approximation accuracy of the bagged Shapley values, we use a low-dimensional dataset, i.e., the five-dimensional phoneme dataset (Triguero et al., 2017), thus requiring the training of feature bagging ensemble models only $2^5 = 32$ times to calculate accurate bagged Shapley values. The low dimensionality of the dataset allows us to compute the non-approximated version of Shapley values without incurring extremely long runtimes. We train isolation trees from (F. T. Liu et al., 2008) with a bagging size of 2

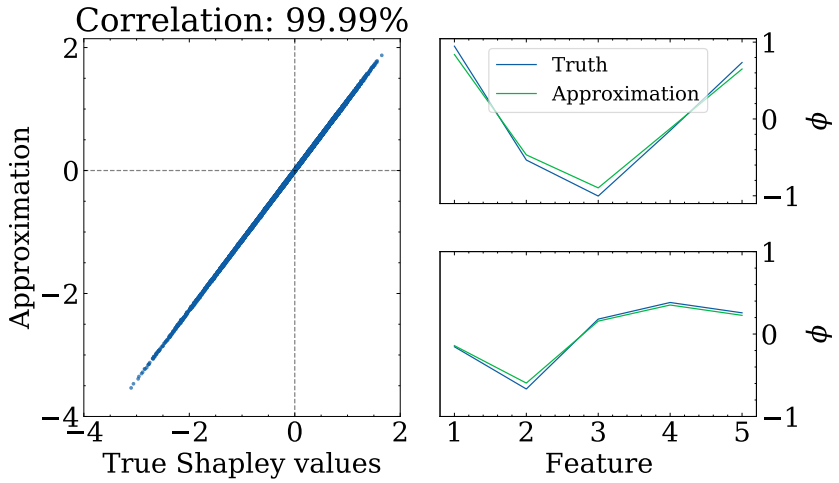


Figure 7.1: Left: Plot of the bagged Shapley values against the exact Shapley values for each data sample in the phoneme dataset. Right: Shapley values and their approximation for two example samples. The color-coding of the features is represented in the legend.

and simplify the anomaly score from (F. T. Liu et al., 2008) to fit our methodology by using the negative average path length over all trees as an indicator of anomalies. We train one million submodels and average the obtained results to guarantee consistent and robust results. The total training takes about 70min of CPU time[1]. The ROC-AUC score is 0.733.

We separate the trained models into ensembles for each subset of them and compute the exact Shapley values and the bagged Shapley values. We combine the values obtained into Figure 7.1. As the mapping lies on the diagonal line, we conclude that the approximation works well on all data points.

It might be possible to change the feature bagging method to not simply choose random features, improving the accuracy further when using significantly fewer submodels. Some initial ideas about this have been explored by a thesis student (H. P. Tang, 2022).

7.6.2 Effectiveness

We can compute the bagged Shapley values for datasets whose dimensions are too high for an exact computation. We focus on the MNIST dataset (Deng, 2012), a collection of images of hand-written digits usually used to train image-recognition models. Following the approach of (Ruff et al., 2018), we consider normal all images representing a handwritten 7, and anomalous the images representing other digits. Each image has a resolution of $28 \cdot 28$, i.e., we handle 784 features in each image. Computing the exact Shapley values for

[1] All experiments were performed on Intel Xeon E5 CPUs. In this chapter, we stick to CPUs over GPUs also when we use neural network submodels; the choice is justified by the higher amount of parallelization they allow.

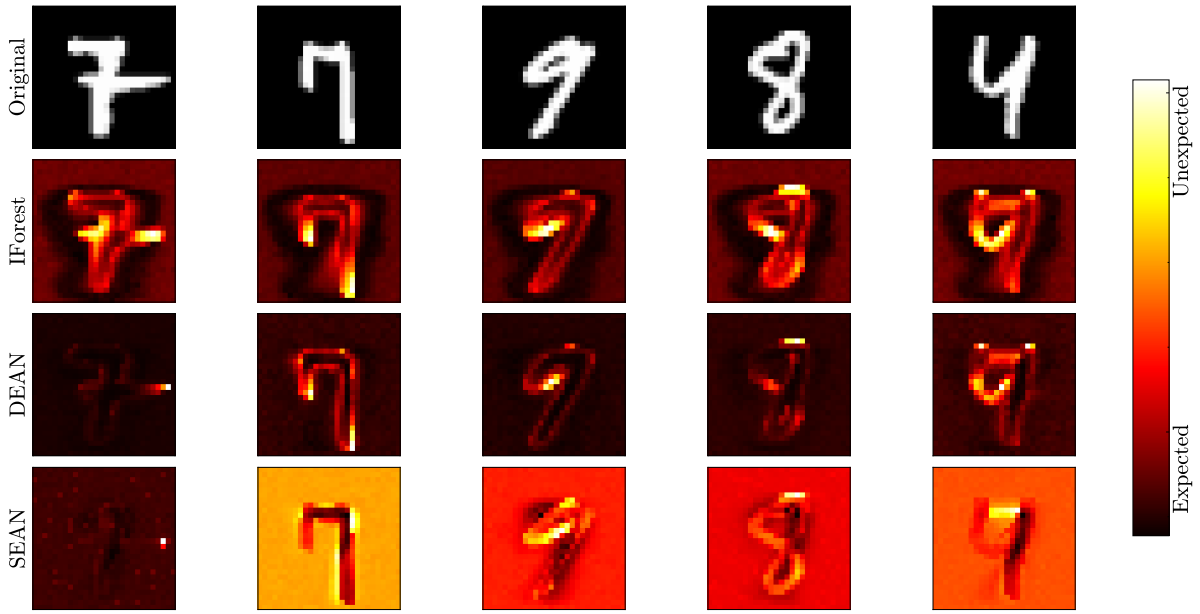


Figure 7.2: MNIST dataset. The original images are in the top row. The bottom rows contain the derived bagged Shapley value heatmap for IFOR, DEAN and SEAN. We rescaled the color legend to the upper and lower bound of the Shapley values in each plot.

the single pixels requires $2^{784} \approx 10^{237}$ evaluations, a number significantly larger than the computational power available.

For the bagged Shapley values, we use the bagging size bag = 32. We train three models: we use DEAN, SEAN, and a shallow isolation forest (F. T. Liu et al., 2008). The training time of DEAN is significantly longer than using IForest or SEAN[2]. Note that we do not only train a model on each possible subset, as the number of subsets is still $\binom{768}{32} \approx 4 \cdot 10^{32}$. Instead, we train on random subsets until the result converges. This also helps deal with the random nature of our algorithms.

Figure 7.2 represents the plots of the Shapley values for five representative samples in the form of heatmaps; bright colors represent high score, i.e., features highly increasing the anomaly score. Each heatmap, both for DEAN, IFOR and SEAN, highlights the changes to the original input that would make it closer to a normal observation by highlighting the erroneous regions. From the left to the right side, the first two input images are labeled as normal; however, they still contain features that are not expected, e.g., the middle horizontal line in the first image. These unexpected features are highlighted in bright red/yellow. Similarly, the other three images obtain high anomaly scores, although they contain typical

[2]The isolation forest takes about 220min of CPU time, and SEAN is even faster, finishing 1 million submodels in 190min. However, DEAN requires about 113days; This is somewhat less problematic since such independent ensembles are easy to parallelize, and less accurate results can already be achieved with ten thousand submodels (27hours).

features for normal input images. These features are also unexpected by the model and thus result in high Shapley values. Examples are represented by the nine and the four; removing the lower line from the circle would make the nine more similar to a normal observation, while adding a horizontal line to the top would make the four more similar to a seven.

Comparing the algorithms, we see how the understanding of the normal concept, i.e., the digit "seven", of the shallow methods (isolation forest and SEAN) is too simple to explain the predictions entirely. In the second column of Figure 7.2, we see that the isolation forest expects the tail of the seven to bend instead of going straight down. On the other hand, DEAN, based on a deep learning method, has less difficulty in learning a broader concept of seven. This is also reflected in the anomaly detection performance: While DEAN reaches a ROC-AUC of 0.9698 on the dataset, the isolation forest and SEAN only reach lower scores of 0.9118 and 0.8591.

We strongly believe that the bagged Shapley images provide useful insights into what the model understood and learned from the training data, as well as better performance measured by the ROC-AUC metric.

7.6.3 Scalability

We select the celebA dataset (Z. Liu et al., 2015) to study how the approach scales to larger datasets. celebA contains images with $218 \cdot 178 = 38804$ pixels, which we convert to grayscale to simplify the plotting later. In the previous section, we showed how complex patterns can overwhelm anomaly detection ensembles that struggle to learn a proper schema for normal and abnormal data points. Thus, we aim to maximize the separation between normal and abnormal classes in order to simplify the learning task. We divide the dataset into normal and anomalous instances, where we characterize a normal observation being labeled with the attributes "female", "young", "attractive", and "not bald". The inverse attributes characterize an abnormal observation. Here, the choice of attributes was only guided by the distribution of attributes in the dataset, and similar results would likely have followed any other choices for the anomalous and normal classes. We only trained the DEAN ensemble on the dataset, as the model proved to handle complicated attributed data best. We represent the obtained bagged Shapley values as heatmaps on five different images in Figure 7.3. The first row is the input image, while the second contains the corresponding Shapley values. The images resulting from the bagged Shapley values plotting have high resolution and show some features as more anomalous; However, the designed features do not match the designed separation in normal and abnormal images. This can also be seen in the ROC-AUC score of 0.6184. The most anomalous features seem to be (from left to

Figure 7.3: Analysis on the celebA dataset. Heatmaps show the bagged Shapley value; Brighter colors indicate higher values.

right) the bindi, the partially covered forehead, the shirt collar, the laugh lines, and the skin paint transition. These are rare features in the images of young women in celebA, thus considered anomalous by the model. Still, the complexity of the separation is likely too big for the available samples (≈ 72000), and thus, the learning, as shown by the ROC-AUC, is inaccurate. Although the features outlined are not the expected ones from our understanding of the separation between the two classes, it is worth noticing how the bagged Shapley value maps can be used to understand and possibly also to fix the functionality of our anomaly detection model. The runtime of the training procedure for one million DEAN submodels is ≈ 468 days. We use 4 millions of submodels in our training setup, under parallelization assumption, and set up the bagging size to be bag = 32. A different bagging size might have achieved more accurate results, but we did not optimize it since, in most contexts, the anomaly detection task sets the bagging size.

We finally want to characterize the minimum number of submodels needed for our methodology to perform well. For this, we calculate the bagged Shapley values maps so that each feature is used on average 10, 100, and 1000 times. The corresponding maps for the center image from Figure 7.3 are shown in Figure 7.4. While some features are already visible at about 12000 submodels, because the noise level being still very high, facial features are undetectable; with about 10 times more submodels, those become visible while after extending the number of submodels to about 100 times more, they have become clear. As a rule of thumb, we suggest training $10 \cdot \frac{N}{bag}$ features to visualize the basic features and to train $10 \cdot (\frac{N}{bag})^{\frac{3}{2}}$ for clear images.

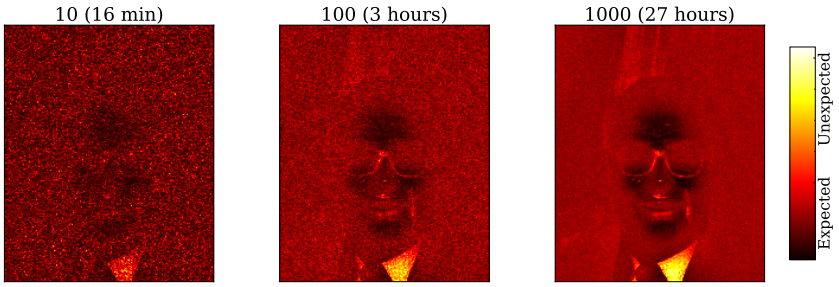


Figure 7.4: Shapley value maps of the DEAN ensemble for different numbers of submodels. Here we use 12127, 121263 and 1212625 submodels, so that each feature is approximately sampled 10, 100 and 1000 times. The times stated assume a parallelization with 500 CPUs.

7.7 Conclusions

Detecting and explaining anomalies can be highly complicated. Shapley values, by their side, offer a flexible definition, easily applicable to this context; However, their high computational costs represent an often insurmountable downside that makes their exact computation often unfeasible.

We combine Shapley values with abstract ensemble techniques, specifically focusing on feature-bagging ensembles for anomaly detection. The *bagged Shapley values* offer an advantageous reduction of the computational costs, giving the chance to compute importance scores for settings with tens of thousands of features. Furthermore, we showed the value of highlighting anomalous features in images to obtain insights into the features learned by the anomaly detection method.

We believe that combining Shapley values with abstract ensemble methods can boost the use of Shapley values in the Machine Learning community, showing advantages from a computational and interpretability point of view, as well as lead to better, more reliable, anomaly detection models.

CHAPTER 8

USING TEST TIME TRAINING TO IMPROVE ANOMALY DETECTION PERFORMANCE

This chapter follows: Klüttermann, S., & Müller, E. (2024). About test-time training for outlier detection [Under review]. *2025 IEEE International Conference on Big Data (BigData)*. https://arxiv.org/abs/2404.03495

8.1 Introduction

Because labeling rare anomalies is an often very expensive task, most anomaly detection algorithms work either unsupervised (without any labels) or in a weakly supervised setting (with only normal labels) (S. S. Khan & Madden, 2010; Olteanu et al., 2023). Some algorithms (Breunig et al., 2000; F. T. Liu et al., 2008) are introduced in an unsupervised setting, while other algorithms (Bounsiar & Madden, 2014; J. Chen et al., 2017; Ruff et al., 2018), including the one introduced in this chapter, follow the weakly supervised setting, either explicitly or implicitly.

But in practice, the difference between both is often ignored, as anomalies are rare and most algorithms also work with slightly contaminated training data (Qiu, Li, et al., 2022). Therefore, algorithms are trained similarly in both cases. We think that this leaves out important information in the weakly supervised setting that can be used to improve the performance of anomaly detection algorithms significantly.

An example of this unused information can be seen in Figure 8.1: Since training and test

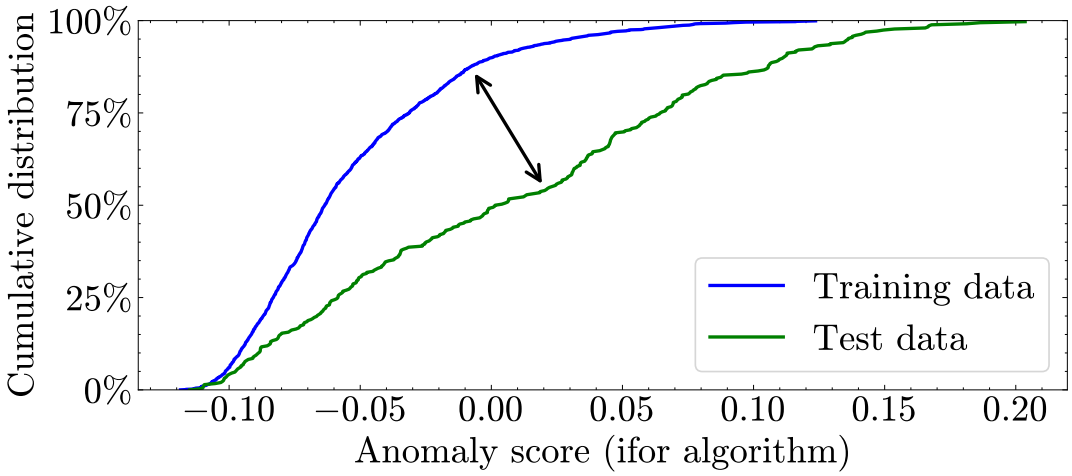


Figure 8.1: Cumulative distribution of anomaly scores of an isolation forest (F. T. Liu et al., 2008) on cardio data (Halder, 2020). There is a difference between training data (100% normal) and test data (50% anomalies), which our approach tries to maximize.

data are generated by partially different processes, we can measure a difference between their distributions. The higher this difference is, the easier the separation between normal and abnormal data becomes. This chapter presents *DOUST*, our method that searches for a data representation explicitly maximizing this difference.

DOUST uses the contaminated test data it is applied to, to specialize a simple DEAN-like anomaly detector to work better at finding anomalies in the same test data (this is sometimes called test-time training (Y. Sun et al., 2020) in supervised machine learning). This approach can be seen as the benefit of knowing the questions of an exam already while studying for it: While the pedagogical use might be questionable, using all available information tends to increase the quality of the answers. Similarly, we reach a drastically superior anomaly detection performance that is comparable to some supervised classification methods, even when no labeled anomalies are given. We believe that needing only normal samples to reach compelling classification performances makes our method applicable to many scenarios. Examples of such situations could include provably faultless historical data (Miljković, 2011) or tasks where simulated data exists (Mikuni et al., 2022).

8.2 Related Work

8.2.1 Test time training

Test time training is a relatively new concept in supervised learning (Y. Sun et al., 2020) (Gandelsman et al., 2022). It tries to solve the problem that the distribution generating the training data often differs from the distribution generating the test data, especially from data used in production environments. For example, a face recognition system might break when everybody suddenly starts to wear face masks. The original approach combines a supervised algorithm, which could not be trained on test data since the test labels are missing, with an unsupervised algorithm that could still be trained without labels. This can allow them to refine learned hidden features to better match test data, even under a possible distribution shift (Liang et al., 2023), making it a subsection of unsupervised domain adaption (X. Liu et al., 2022).

This approach is not susceptible to overfitting even though test samples are used since no access to the test labels is given.

In supervised test-time training, the primary difficulty is that there might be a misalignment between the supervised and the unsupervised optimization goals. However, in the weakly supervised setting studied here, we do not need to combine two models, as our training is independent of supervised labels.Interestingly, many older anomaly detection papers also do not separate between training and test data (Breunig et al., 2000). And some algorithms (F. T. Liu et al., 2008) even work slightly better when using anomalies in their training data.

While we need to optimize our model not only during training time but also during test time, this is no different from how, e.g., a k-nearest neighbor anomaly detection algorithm (Gu et al., 2019) searches for anomalies: Except maybe for some pretraining to increase the efficiency, most computation time is required during testing, by evaluating the difference between training and test samples.

DOUST extends these ideas to modern deep learning methods, explicitly searching for differences between distributions during test time.

8.2.2 Positive unlabeled learning

In positive unlabeled (PU) learning, a machine learning method considers both a set with samples from a positive class and a set of unlabeled samples (Bekker & Davis, 2020). This is usually studied in supervised learning (For example, a missing medical diagnosis only

indicates that a patient could be healthy or sick), but can also be extended to anomaly detection. The most common application is semi-supervised learning (Ruff et al., 2019), where an unlabeled set of both normal and abnormal samples and a positive set of a limited number of known anomalies is given. This can be seen as a restricted version of supervised learning and tends to often work better than unsupervised methods even when only a single anomaly is given (Han et al., 2022; Ruff et al., 2021). Still, this only works as long as the anomalies are similar to those used for training.

PU learning can also be used when a set of normal and a set of unlabeled samples are given, even though this is less developed. For example, PU learning can be used similarly to a modern reject option (X. Li et al., 2007), removing anomalies from a classification task. Other papers focus on indirect optimizations to improve existing anomaly detection methods, like threshold selection (Tian et al., 2019), density estimation (Hido et al., 2011), or model selection (Baudzus et al., 2023). A more direct method is studied in (Mu et al., 2021; J. Zhang et al., 2017) , by trying to transfer the PU learning task to a usual classification task, through an iterative approach. This approach selects the most anomalous samples from the unlabeled set, and trains a supervised algorithm on this set, and uses this supervised algorithm to find the most anomalous samples in the following steps. Still, this has a drawback similar to the semi-supervised approach: Only anomalies similar to these most anomalous samples can be found. Also, the iterative approach can allow singular mistakes in the original assessment of anomalousness to snowball into a misalignment between the type of samples found and the anomalies searched for.

In contrast, DOUST tries to directly maximize the difference between both sets, making it less susceptible to an initial anomaly assessment. Additionally, through the use of test-time training, we do not require our anomalies to generalize, as we search for them directly on the test set. This also means that we do not need to collect additional unlabeled data for our training phase. And most importantly, we do not require any expensive labeled anomalies.

8.3 Anomaly refinement with Test time training

Anomaly detection is often done in a weakly supervised setting, where only normal data points are known. As shown in Figure 8.1, this implies that there is a measurable difference between the distribution of data used for training and that used for testing. Our idea is to create a one-dimensional data representation that maximizes the difference between training and test samples, and thus also between normal and abnormal ones (as shown in

Appendix A.2.2). This representation can thus be used as an anomaly score, where higher values represent more anomalous samples.

Our method, Deep OUtlier Selection with Test-time training (DOUST), contains two training steps. One pretaining step during training time, and one to refine our model on the test data.

After defining a neural network (for the sake of brevity, our specific hyperparameter choices can be found in Appendix A.1.1) to transform samples to a one-dimensional representation, we first train it to learn a constant value for each sample in the training set (similar to Chapter 4) using the loss function defined in Equation 8.1.

$$L_0 = \sum_{x \in X_{train}} (f(x) - \frac{1}{2})^2 \tag{8.1}$$

We use a special activation function (Equation 8.2) in the last layer of our neural network:

$$S^+(x) = S(x-1) = \frac{1}{1+e^{1-x}} \tag{8.2}$$

This shifted sigmoid fulfills $S^+(x) \in (0,1)$, simplifying later calculations, but also does not satisfy the trivial solution of $S(0) = \frac{1}{2}$, which would trivialize the loss in Equation 8.1.

After this first step, removing some of the uncertainty of random initialization, pretuning the network for anomaly detection, and setting all values close to the middle of possible distribution values, we will add a second step, pulling apart normal from abnormal samples. This second step is part of the evaluation since it requires access to our test data. It allows us to specialize our prediction to the specific anomaly detection task. This also means that when evaluating our algorithm on new data, the second training step needs to be repeated to achieve maximum performance.

In the second step, many different losses are possible, and we will study these further in Appendix A.4, but here we study a slightly modified mse loss (Equation 8.3). This loss requires each training sample representation to be as small and each test sample representation to be as high as possible.

$$L = \frac{1}{\|X_{train}\|} \sum_{x \in X_{train}} f(x)^2 + \frac{1}{\|X_{test}\|} \sum_{x \in X_{test}} (1 - f(x))^2 \tag{8.3}$$

Since many samples from the test set are considered normal and thus follow the same distribution as those from the training set, this does not separate training from test samples but instead separates normal from abnormal ones. To show this, we assume infinitely many

samples equally distributed over the training and test set: $\|X_{train}\| = \|X_{normal}\| \to \infty$. Also, we assume that the training data is sampled from a distribution p_{normal}, while the test data follows $(1-\nu)\cdot p_{normal} + \nu \cdot p_{abnormal}$ (with the fraction of abnormal data samples in the test set ν). Rewriting Equation 8.3 with these assumptions, we find:

$$L \propto \int_{\mathscr{R}^d} dx\, p_{normal}(x)\cdot(f(x)^2 + (1-\nu)\cdot(1-f(x))^2) + p_{abnormal}(x)\cdot\nu\cdot(1-f(x))^2 \tag{8.4}$$

This loss is minimal, when $f(x) = 1$ for all abnormal samples and $f(x) = \frac{1-\nu}{2-\nu} \leq \frac{1}{2}$ for all normal samples.

Proof. We consider it self-evident that $f(x) = 1$ is the minimum for abnormal samples. The minimum for normal samples is given when $(f(x)^2 + (1-\nu)\cdot(1-f(x))^2)$ is minimal. Thus $\frac{d}{dy}(y + (1-\nu)\cdot(1-y)^2) = 0$, which is equivalent to $1-(1-\nu)\cdot 2\cdot(1-y) = 0$ and thus $f(x) = \upsilon = \frac{1-\nu}{2-\nu} \leq \frac{1}{2}$. ■

While practical implementations have further complications, limiting this relation (Limited neural network complexity, sampling uncertainty, and contamination. These are partially studied in Appendix A.2.4), this relation allows DOUST to explicitly optimize finding samples without knowing what they look like.

During implementation, we found that using an ensemble (similar to the DEAN ensemble in Chapter 4, using 100 submodels) is slightly beneficial (Appendix A.2.5.1) but also that feature bagging (Lazarevic & Kumar, 2005) should not be used (In contrast to DEAN, see Appendix A.2.5.2). These kinds of optimizations were initially only done once on one unrelated dataset (Malyi, 2017), to not taint our shown results.

8.3.1 Experimental comparison

This we will now evaluate experimentally. For this, similar to the evaluation in Chapters 4 and 5, we follow a recent survey paper (Han et al., 2022). In this chapter, we limit ourselves here to the 47 datasets considered classical for computational reasons. We initially also only compare against the three best algorithms ADBench found as competitors. These are a k-neirest neighbor algorithm (Gu et al., 2019), an isolation forest (F. T. Liu et al., 2008), and cblof (Z. He et al., 2003) (a cluster-based extension of lof (Breunig et al., 2000)). We validate this choice of competitors later by also comparing it against the other competitors from Chapter 4. Additionally, we compare against a supervised random forest algo-

rithm (Breiman, 2001). As the datasets are highly imbalanced to work well with unsupervised algorithms, we only use the labeled test set of these algorithms for the random forest but employ cross-validation (Refaeilzadeh et al., 2009) to remove overfitting effects. Notice that this is a deliberately unfair comparison. While our anomaly detection competitors, as well as DOUST, have no access to labeled anomalies to train on and thus effectively have to separate every type of anomalies, the supervised random forest has access to labeled data and simply has to learn to separate two finite groups of samples. Considering that access to a singular labeled anomaly is often enough to outperform each unsupervised algorithm (Olteanu et al., 2023; Ruff et al., 2021), the random forest with access to all anomalies can be seen as close to an upper limit to the performance we are able to reach. We choose here to use a random forest over any more modern classification algorithm since tree-based ensemble methods are not strongly affected by hyperparameters or preprocessing steps and work on most dataset sizes (Roßbach, 2018). Likely, when optimizing, for example, a neural network-based classifier, a better performance could be achieved. But this would be an even more unfair comparison, and we would have to handle overfitting and overoptimization effects. More details about our competitors can be found in Appendix A.1.2.

To evaluate a model's performance, we will use the ROC-AUC score, as it is independent of the fraction of anomalies (ν in Equation 8.4). This is required for some of our analyses (See Appendix A.2.1) and thus we are not able to use the AUC-PR score (Boyd et al., 2013) in this chapter.

8.3.2 Handling erroneous models

Some training loops fail because TensorFlow is not built for the complex, multi-stage optimization task we propose here. And while there are always hyperparameters that could be changed (for example, each dataset is solved by at least one of the loss functions described in Appendix A.4), the high number of models we train, makes optimizing specific model impossible. Additionally, this might result in unfair comparisons when comparing differently trained algorithms.

Instead, we ignore these rare measurements by removing datasets on which not every used algorithm worked. This means that we will use a different number of datasets in each comparison, and numeric results are usually only comparable in each diagram. We will clarify how many datasets could be used for each result, but each result uses at least half of all datasets outside of Appendix A.4.

Erroneous models usually happen in one of three ways:

1. When a training loop only sometimes fails but also works sometimes, we will ignore this error. This is possible since each performance is the performance of an ensemble with (up to) 100 submodels, but these ensembles usually converge already at $5-10$ submodels. Thus, combining a few fewer models does not meaningfully affect the results.

2. If a training loop does not run successfully converge to a value except NaN at all, we will remove this dataset. This happens only rarely (2 of 47 datasets), but this is still non-neglectable, especially when comparing multiple settings.

3. In Section 8.5, we will modify our datasets by altering the fraction of anomalies ν. This can result in datasets without anomalies, which will also be removed. Because of this, a maximum of 31 datasets are used in Section 8.5.

Overall, this shows the biggest drawback of DOUST, as it is significantly less reliable than the methods introduced in earlier chapters.

8.4 Experimental results

We summarize the performance of DOUST in a critical difference plot in Figure 8.2. This critical difference plot compares the performance of each model to that of the other and marks differences that are not statistically significant (like ifor and cblof). For this, we use a Friedman test (Friedman, 1940) to see if there are any significant differences between the methods. Afterward, a Wilcoxon test (Wilcoxon, 1945) test determines which of the differences are significant. For this, we consider p-values below $p < 5\%$ (after a Bonferroni-Holm correction (Holm, 1979)) as significant. Further comparisons to other more modern, but worse algorithms can be found in Table 8.1 and validate our findings.

Even though our unsupervised competitors are some of the effective algorithms for finding anomalies on these datasets, the average rank of the DOUST algorithm is significantly better. The change in ROC-AUC is even more significant, with our algorithm performing more than twice as close to a perfect separation of 1 than each unsupervised competitor.

However, perfect separation is rarely possible since datasets are usually imperfect as distributions overlap. A more reasonable limit is the supervised random forest (An algorithm using unlabeled data should perform worse than one with access to labels (Olteanu et al., 2023; Ruff et al., 2021)). Here, our algorithm performs 99% as well as this limit, with

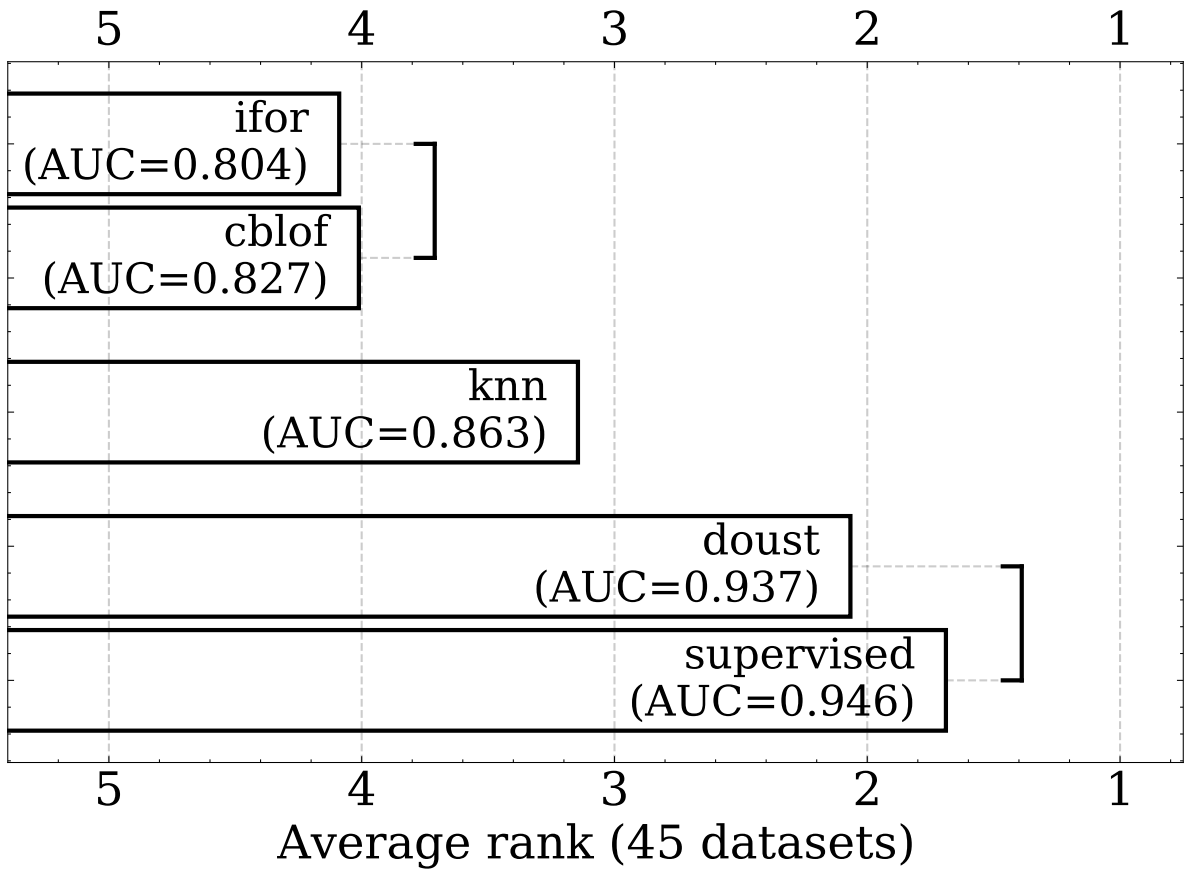


Figure 8.2: Critical difference plot. We also add the average ROC AUC of each algorithm. As expected, the three unsupervised algorithms perform worse than the supervised ones. But DOUST performs almost as well as the supervised random forest (there is no statistically significant difference between their performance at $p = 5\%$).

only an insignificant difference in our datasets. This is a significant discovery since it effectively means that we are able to find anomalies almost equally well, whether we have access to labeled anomalies or not.

8.5 Optimization misalignment

For most anomaly detection algorithms, the fraction of anomalies in the test set (ν) does not affect the predictions or the results (when using ROC AUC (Hanley & Mcneil, 1982) for evaluation). We have already seen in Section 8.3, that the optimal separation between normal and abnormal data is $\Delta = 1 - \frac{1-\nu}{2-\nu} = \frac{1}{2-\nu}$. This means the fever anomalies are in the test set, the smaller the separation between normal and abnormal samples. But this

Algorithm	Average performance
DOUST	**93.72%**
KNN (Ramaswamy et al., 2000)	86.31%
LOF (Breunig et al., 2000)	83.50%
CBLOF (Z. He et al., 2003)	82.72%
DEAN (Chapter 4)	81.57%
IFor (F. T. Liu et al., 2008)	80.43%
NeuTral (Qiu, Pfrommer, et al., 2022)	78.03%
AE (Sakurada & Yairi, 2014)	77.73%
PCA (Callegari et al., 2011)	77.36%
DeepSVDD (Ruff et al., 2018)	77.28%
HBOS (Goldstein & Dengel, 2012)	76.98%
OCSVM (Bounsiar & Madden, 2014)	76.04%
COPOD (Z. Li et al., 2020)	74.53%
ECOD (Z. Li et al., 2022)	74.30%
GOAD (Bergman & Hoshen, 2020)	73.57%
LODA (Pevný, 2016)	70.29%
DTE (Livernoche et al., 2024)	69.00%
NF (Rezende & Mohamed, 2015)	67.16%
DAGMM (Zong et al., 2018)	66.37%
VAE (Kingma & Welling, 2014)	64.12%

Table 8.1: Comparing other unsupervised competitors, including deep learning approaches to DOUST.

difference can often either be ignored (at $\nu = 1\% \rightarrow \Delta \approx 0.503$, while at $\nu = 5\% \rightarrow \Delta \approx 0.513$) or fixed (See Appendix A.2.4). Still, when evaluating the effect of ν on the performance of DOUST, Figure 8.3 shows a clear drop for smaller values of ν. Thus, a more devious effect must make our algorithm depend on ν. This effect will be explained in this section.

Our performance is very competitive for a fraction of $\nu = 50\%$ anomalies in the test set; it becomes more average at single-digit values of ν and seems to drop off further for even lower anomaly fractions. So, to have an algorithm that is also useful in real-world applications (anomalies are rarely as common as $\nu = 50\%$), it is vital to understand the effect driving this change.

Looking at Figure 8.3, it is worth noting that our competitor's performance also changes. This is because we remove anomalies from our test set until ν is approximately fulfilled (thus, we also average slightly different anomaly fractions and remove datasets with less than 200 anomalies). And while the resulting ROC-AUC performance does not change on average, it still fluctuates depending on whether we remove easy or hard-to-find anomalies. Thus, a better interpretation of Figure 8.3 is that the average AUC drops compared to our

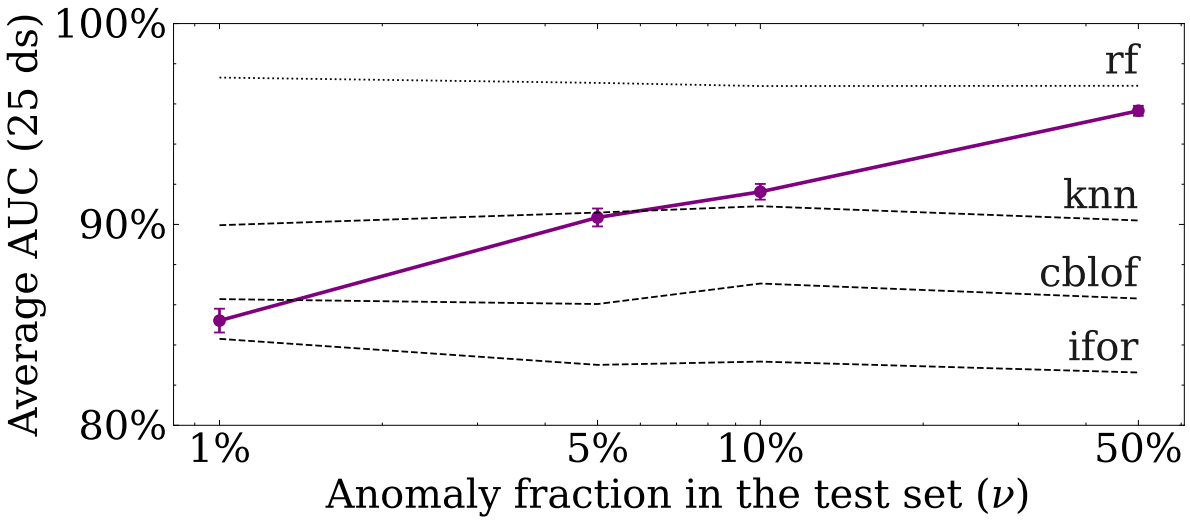


Figure 8.3: Comparison of the average AUC, of our algorithm (purple), compared to our competitors (black), as a function of the fraction of anomalies in the test set ν. To increase readability, we use a logarithmic x-axis. We are only able to use 25 of the available datasets since not every dataset contains enough anomalies to subsample the test set.

competitors.

To explain why this happens, we suggest a thought experiment. Given a normal distribution of samples, we would like to find anomalies that do not conform to this distribution. To do this, we can choose one of two thresholds: one right and one left of our data, and we choose the threshold that separates more points, as the additional anomalies should make it more likely for this side to contain more samples. We visualize this thought experiment in Figure 8.4.

We want to answer how likely it is that the right side is chosen, and thus the anomalies are found. For this, the right side needs to contain more samples than the left one: This will consistently happen when there are enough anomalies A, and the anomalies are found in close to 100% of cases, but if $A \ll N \cdot f$, then it becomes more likely that random fluctuations dominate the number of samples on both sides and thus the probability drops to close to 50%. We simulate this behavior in Figure 8.5.

At some point, it becomes likely that the random fluctuations of the normal distribution dominate our experiment. Thus, the anomalies of the normal samples are more significant than those that we consider anomalous, creating a misalignment. Because there are too few anomalies, these are not captured by our loss, and our algorithm prefers searching for differences between the two normal distributions.

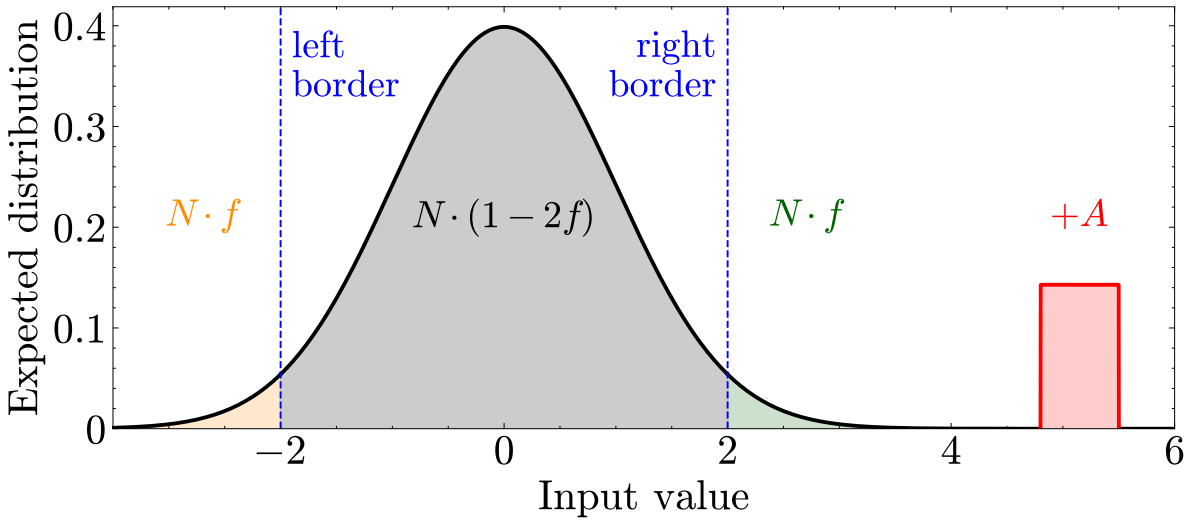


Figure 8.4: Visualization of our thought experiment. We consider two distributions. One that is normal is shown in grey (N samples), and one that is abnormal further away (red, A samples). To find this anomaly distribution, we consider one of the blue lines as a threshold, depending on which threshold separates more samples. Generally, also, some normal samples are considered anomalies. On average, these are $f \cdot N$ on each side, but this number is highly uncertain, giving rise to situations in which the wrong border seems favorable.

Alternatively, when always considering samples on both sides as anomalous, the performance might be worse for a high ν, but stays constant. Applying this to our original comparison, this can be seen as an algorithm not specified on a specific anomaly detection dataset but used to find each type of anomaly (like our unsupervised competitors). While the ROC-AUC performance is competitive, consider that this algorithm consistently has twice the number of false positives, resulting in its making more mistakes than the blue line even for a high anomaly fraction ν (See Appendix A.2.1.1), limiting the effectiveness of this type of anomaly detector.

Mathematically, we can find a region in our thought experiment in which the algorithm searching for the right separation always reaches the right conclusion. Namely, we search for the region in which the uncertainty of the number of normal points is significantly lower than the number of added anomalies. We assume the number of normal points to follow a binomial distribution with mean $N \cdot f$. Using the uncertainty of a binomial distribution $\sqrt{N \cdot f \cdot (1-f)}$, we can write this condition as:

$$\sqrt{N \cdot f \cdot (1-f)} \ll A \leftrightarrow f \cdot (1-f) \ll A \cdot \frac{A}{N} \tag{8.5}$$

Here f represents the separation between normal and anomaly distribution. The higher

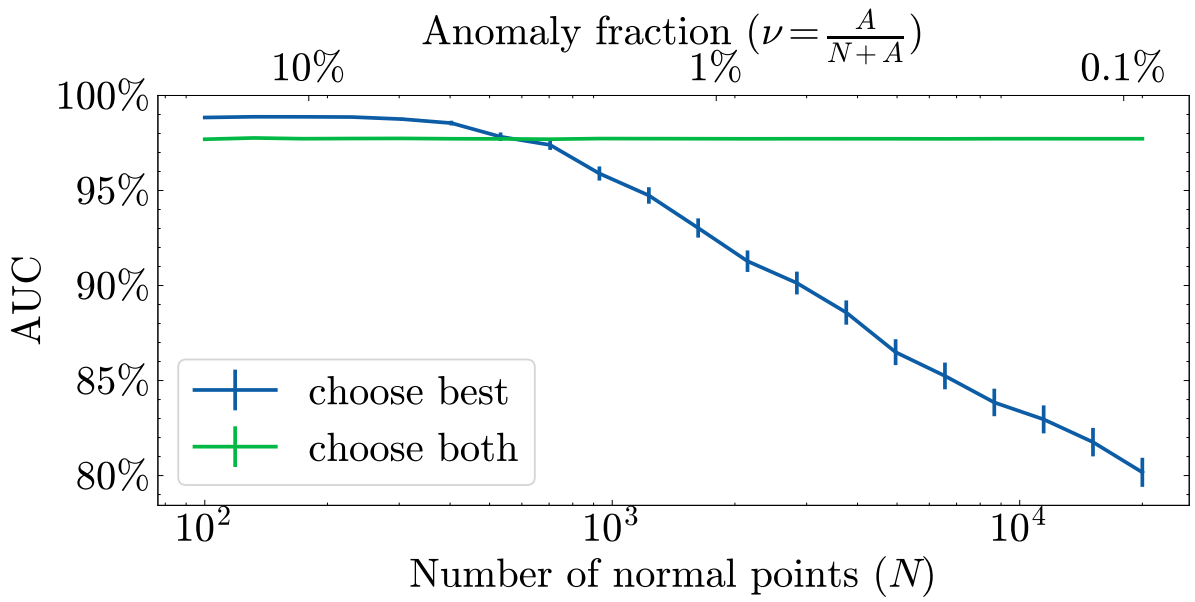


Figure 8.5: Experimental results of our thought experiment (in blue). When increasing the number of normal samples considered and thus decreasing the anomaly fraction ν, the ROC-AUC drops, representing a worse separation. This is because it is less likely that the right threshold will be chosen. For comparison, the green line represents a model that considers both samples outside both thresholds as anomalies, similar to a classical unsupervised anomaly detector.

the separation, the lower f and the easier it is to fulfill the condition. We usually don't know f, but we can limit it by its maximum value, as $f \in (0,1)$ and thus $f \cdot (1-f) \leq \frac{1}{4}$.

As expected, Equation 8.5 becomes more challenging to fulfill when anomalies are rare (Assuming $A \ll N$, $\frac{A}{N} \approx \nu \to 0$), but it also becomes more accessible to fulfill when there are more anomalies ($A \to \infty$). This implies that for a fixed anomaly fraction ν, the more measurements there are, the more likely the separation succeeds.

As a rule of thumb, this requires approximately $N \gg \frac{1}{\nu^2}$ samples.

Because of our downsampling strategy, we effectively decrease the performance in Figure 8.3 twice, as we do not only decrease the fraction of anomalies $\nu = \frac{A}{N(+A)}$, but also the total number of anomalies (Also see Appendix A.2.3).

To test whether this effect generalizes for more than our thought experiment, we have to rely on simulated data to change the anomaly fraction without downsampling. This we do in Figure 8.6; we apply our method to find anomalies on ten-dimensional simulated data, where normal and abnormal distributions are Gaussian, with $\mu = \vec{0}, \vec{1}$ and $\sigma = \vec{1}$ and a low contamination of $\nu = 1\%$ is given. More details about the experimental setup can be found

in Appendix A.1.1.3.

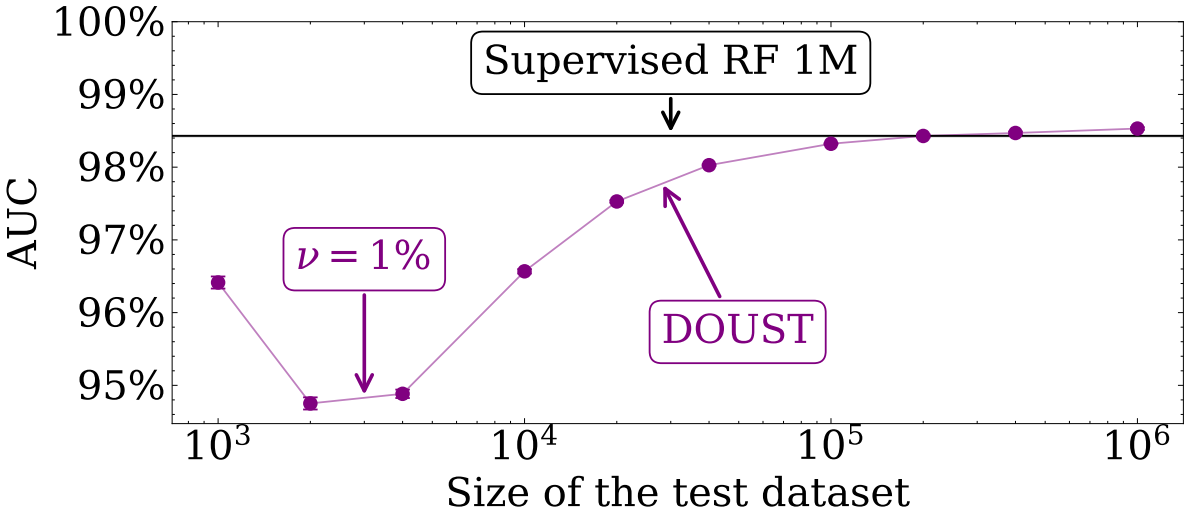


Figure 8.6: ROC-AUC score of DOUST on simulated data with a low contamination of 1%. While for a low number of measurements, the AUC is worse, it increases to the supervised limit when increasing the dataset size (and even surpasses it slightly). This limit here is again calculated through a supervised random forest with $1M$ training and test samples.

Interestingly, the separation quality is not monotonous but contains a surprisingly high value at 1000 samples. Our explanation for this is that these are so few samples that our second training step has not yet converged, and we are not finding the worse solution.

However, afterward, we see that our separation quality continuously grows as we have more measurements. The performance here does not reach perfect separation, as both distributions somewhat overlap, but almost reaches the supervised limit with 200000 samples and even surpasses it slightly for higher values.

This shows that the requirement for DOUST to reach supervised performance in the zero-supervised case is to only have enough (unlabeled) measurements, making our method highly useful for many applications where unlabeled data can cheaply be recorded. Still, it might be possible to modify our approach to work when anomalies are rare and few measurements are made. Intuitively, following our thought experiment, it might be possible to choose to "select both borders" and follow an unsupervised paradigm, at least as long as there is confusion. Thus, this would use DOUST only when it provides a benefit.To help with further development, we describe other approaches we have taken in the appendix.

This includes weighting the terms of our loss function to move the equilibrium described in Equation 8.4: Appendix A.2.4 shows that such weights do help to improve the

performance in every case, but especially at low ν. Still, it is also not enough to cancel out the performance decrease.

Instead, in Appendix A.4, we describe our experiments with alternative loss functions. Notably is the function described in Appendix A.4.1, which is independent of the fraction of anomalies in the test set ν. Sadly, this independence comes at the cost of lower overall performance and a more complicated training procedure.

8.6 Conclusion

In this chapter, we are the first to explicitly maximize the difference between training and test distribution for anomaly detection. As long as two conditions are fulfilled, this allows our algorithm DOUST to outperform competitive algorithms drastically. These conditions are

1. A clean training set, not containing anomalies; thus weakly supervised classification
2. Enough unlabeled test samples ($N \gg \frac{1}{\nu^2}$)

If these conditions are fulfilled, DOUST is able to achieve supervised performance while using zero labeled anomalies. We believe the possible use cases of this to be immense. While many supervised fault detection algorithms work well on specific cases, complicated unsupervised/weakly supervised algorithms are needed to catch unexpected errors with a lower success rate. Using our algorithm, we can also catch these unexpected errors with a performance that is competitive to the expected labeled errors.

However, we also do not believe our algorithm is perfect yet. The space of possible loss functions is large, and it is likely that there are those loss functions that require significantly fewer samples to work (second condition), extending our possible applications even further. Similarly, we think it should also be possible to soften our first condition further. While we assume this here, we likely are never able to use perfectly clean training data. However, a difference in anomaly fraction between training and test data might also be more than enough (Appendix A.2.4).

To extend this, further studies proposing shallow alternatives to DOUST (Ok, 2024) and using test-time data for model selection (Hariharan, 2024) have already been conducted by thesis students.

CHAPTER 9

UNDERSTANDING THE LIMITS OF TEST TIME TRAINING IN ANOMALY DETECTION

This chapter follows: Klüttermann, S., & Müller, E. (2025). Rare anomalies require large datasets: About proving the existence of anomalies [Under review]. *International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*

9.1 Introduction

Despite the diversity of anomaly detection algorithms and their applications, a common characteristic among them is their output of anomaly scores for individual samples. In general, a higher anomaly score indicates a greater likelihood that a sample is anomalous, allowing for a ranking of dataset samples based on their anomaly levels. However, in practical applications, the ranking of anomalies may not be the primary concern. While identifying where an error has occurred is often important, determining **that** an error has occurred is a more fundamental question. In this chapter, we demonstrate that addressing this question is far from trivial and, in many cases, more complex than simply ranking anomalous samples. Furthermore, studying this question can help explain the lower limit ($N \gg \frac{1}{\nu^2}$) observed in the previous chapter.

Our approach leverages the presence of a labeled set of normal samples alongside an unlabeled dataset (weakly supervised setting), allowing us to examine whether anomalies can be detected by assessing differences between these datasets. To explore this question,

we construct a comprehensive set of anomaly detection problems, covering four orders of magnitude in dataset sizes and five orders of magnitude in contamination rates within the unlabeled test set. Our analysis employs four different statistical tests and five distinct anomaly detection algorithms.

Through this analysis, we identify a decision boundary that differentiates between scenarios in which anomalies can be reliably detected and those where detection is unfeasible. Our findings indicate that this decision boundary is influenced by both the contamination rate and the dataset size and remains similar to the limitation studied in the previous chapter across all tested statistical methods and anomaly detection algorithms. Finally, we introduce a further thought experiment to show that we see a similar behavior even when drastically simplifying our setup and removing both the anomaly detection algorithms as well as the statistical tests.

9.2 Testing for the existence of anomalies

Most anomaly detection algorithms compute an anomaly score, ranking samples by their level of anomalousness. However, an anomaly score of, for instance, 3.2 does not directly indicate whether a sample is genuinely anomalous or not, as it lacks an absolute meaning. Typically, a threshold is applied, such as classifying samples with scores higher than 95% of normal training samples as anomalous. Yet, in practice, the primary question may not be where an anomaly has occurred but whether any anomaly is present at all. For instance, in aviation, the occurrence time of an electronic fault in the last flight may be less crucial than knowing if a fault occurred at all. This binary determination cannot be achieved solely through a threshold-based approach, which usually designates a fraction of samples as anomalous regardless of the actual presence of anomalies.

To address this, research has explored methods for translating anomaly scores into probabilities (Gao & Tan, 2006). However, this remains a challenging problem (Röchner et al., 2024) as accurate probability estimation requires careful calibration, and to determine if anomalies are present, one would need to aggregate potentially thousands of probability values, thus compounding any errors. In contrast, we propose a different approach by directly testing for statistical differences between normal and potentially contaminated samples.

Instead, to determine whether a potentially contaminated dataset contains anomalies, we employ statistical tests. Specifically, we focus on tests that assess differences between a known uncontaminated reference dataset and a potentially contaminated, unlabeled test

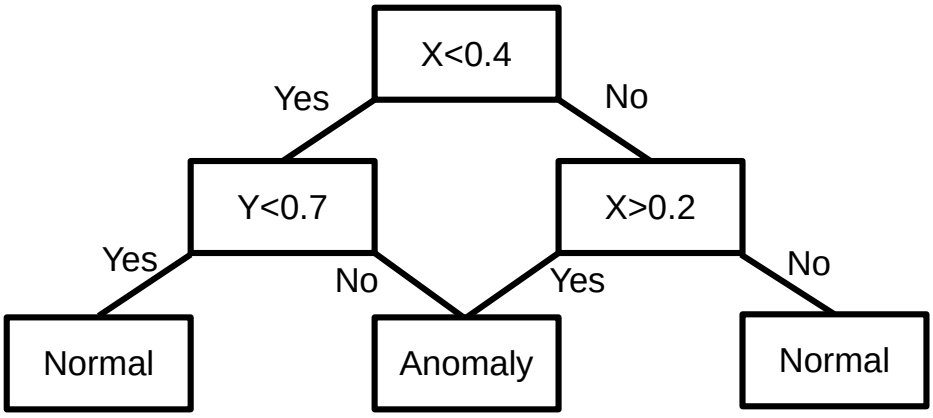


Figure 9.1: Example decision tree used to generate our data. This tree is two dimensional and has a depth of 3.

set. The null hypothesis assumes that the distributions of the two datasets are identical. Our objective is to identify cases where a statistically significant difference ($p < 0.05$) indicates the presence of anomalies. Several statistical tests have been developed for this purpose, each with distinct advantages and limitations. In this work, we employ four such tests. First, we use the Kolmogorov-Smirnov test (Massey, 1951), which compares empirical distribution functions. Next, we apply the Anderson-Darling test (Anderson & Darling, 1954), which is more sensitive to the tails of distributions and may be particularly effective for detecting anomalies. Similarly, we consider the Mann-Whitney U test (McKnight & Najab, 2010), known for its robustness to outliers, making it less prone to being misled by extreme values in the training set. Finally, we also include the simpler Student's t-test (Student, 1908).

9.3 Methodology

To analyze the effect of dataset size across multiple orders of magnitude, we rely on simulated data, as most anomaly detection benchmark datasets contain only a few thousand samples, with even fewer containing a substantial number of anomalies. For this purpose, we generate 100 synthetic datasets, each consisting of ten-dimensional data where each feature is normalized to range from 0 to 1.

Data generation proceeds by constructing a complete binary decision tree of depth 5. In each branch of the tree, a feature and a split value are randomly selected, and each leaf node is labeled as either True or False. A sample is labeled as "normal" if it reaches a leaf marked True and "anomalous" otherwise, resulting in a complex yet randomized distribution of normal samples within a defined region. An example of such a decision tree is shown in

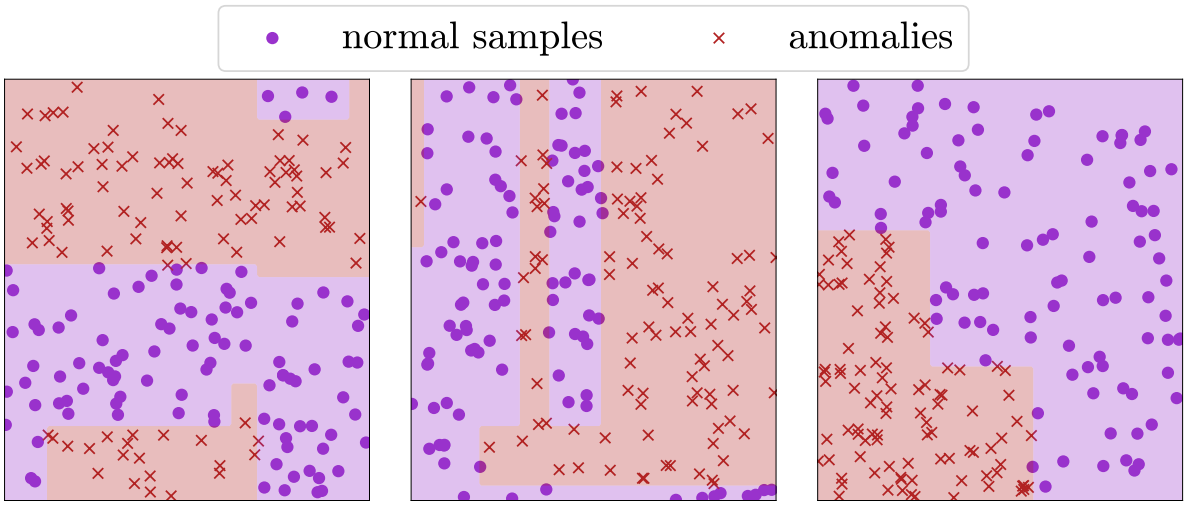


Figure 9.2: Three examples of datasets generated using our data generation algorithm, shown in two dimensions. This setup enables us to create arbitrarily large datasets for our experiments. We show here an uncharacteristically high number of anomalies to better visualize them.

Figure 9.1. Also Figure 9.2 illustrates three examples of these generated datasets (shown in two dimensions for visualization), demonstrating the variability and complexity of our data generation process.

For each dataset, we generate up to one million samples for the training set, as well as for both the normal and anomalous portions of the test set.

Using these synthetic datasets, we train various anomaly detection algorithms to detect significant differences between clean training data and contaminated test data. Each model is trained on N normal samples and subsequently evaluated on a test set of N samples, of which $\nu \cdot N$ samples are anomalous (with ν as the contamination rate).

After training, we measure the effectiveness of anomaly detection by evaluating the fraction of datasets where a significant difference between normal and contaminated data is detected by a statistical test (with $p < 0.05$). We repeat this process across different statistical tests, anomaly detection algorithms, dataset sizes N, and contamination rates ν to comprehensively assess performance.

9.4 Experiments

We begin by analyzing the fraction of datasets with significant differences on the maximum dataset size $N = 1,000,000$ using a Gaussian Mixture Model (gmm) (Oluwasegun & Jung, 2023) in Figure 9.3.

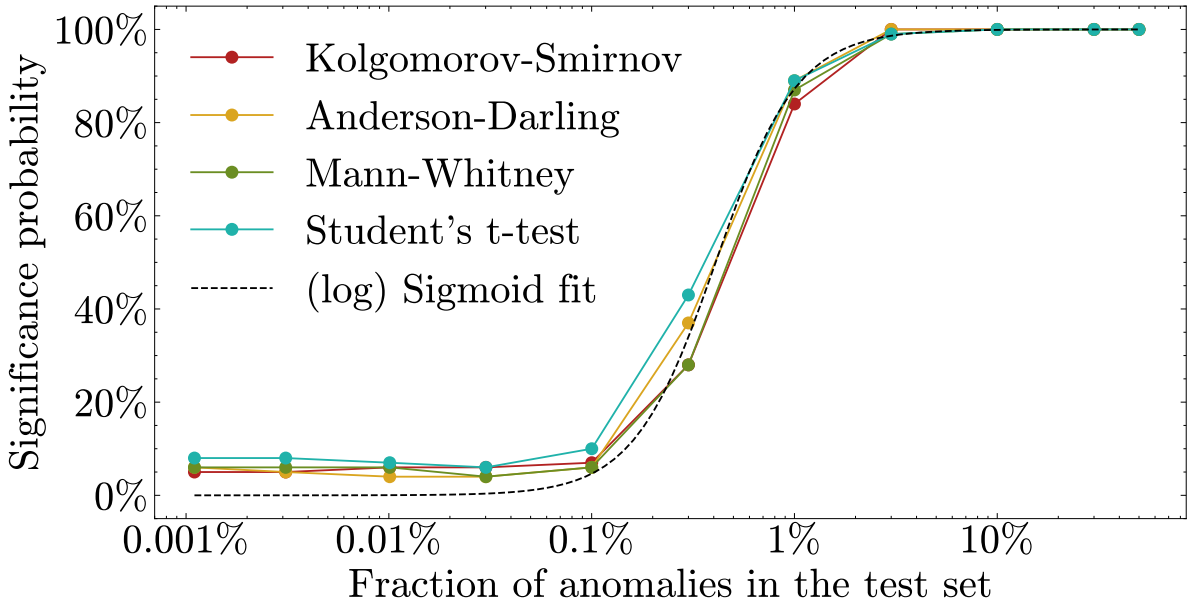


Figure 9.3: Fraction of datasets that contain a significant difference between uncontaminated training data and contaminated test data for various statistical tests and contamination rates using a Gaussian Mixture Model with one million total points in the training and test sets.

Two key observations arise here. First, the difference between the applied statistical tests is minimal, with the largest difference observed between the Student's t-test and the Mann-Whitney test, approximately 15% at $\nu = 3 \cdot 10^{-3}$. This holds across other values of N and different anomaly detection algorithms, with a maximum observed deviation of 22% between two statistical tests in all experiments studied in this chapter. Therefore, for simplicity, we use the average significance probability over all tests in subsequent analyses. Second, the general curve appears to follow a sigmoid function in logarithmic space. Specifically, we fit a logarithmic sigmoid function to the average probability:

$$P = \text{sigmoid}\big(\text{slope} \cdot (\log(x) - \log(\text{center}))\big) \tag{9.1}$$

Using gradient descent, we estimate slope ≈ 2.15 and center $\approx 4.1 \cdot 10^{-3}$ in this example relation. This sigmoidal function seems to capture the relationship well, suggesting that only two cases exist: at high contamination rates ν, the difference between training and test data becomes statistically noticeable, while at low ν, it is statistically negligible. The transition between these states is relatively sudden, spanning roughly a factor of 10 in ν. This pattern is consistent across different algorithms and values of N. Notably, however,

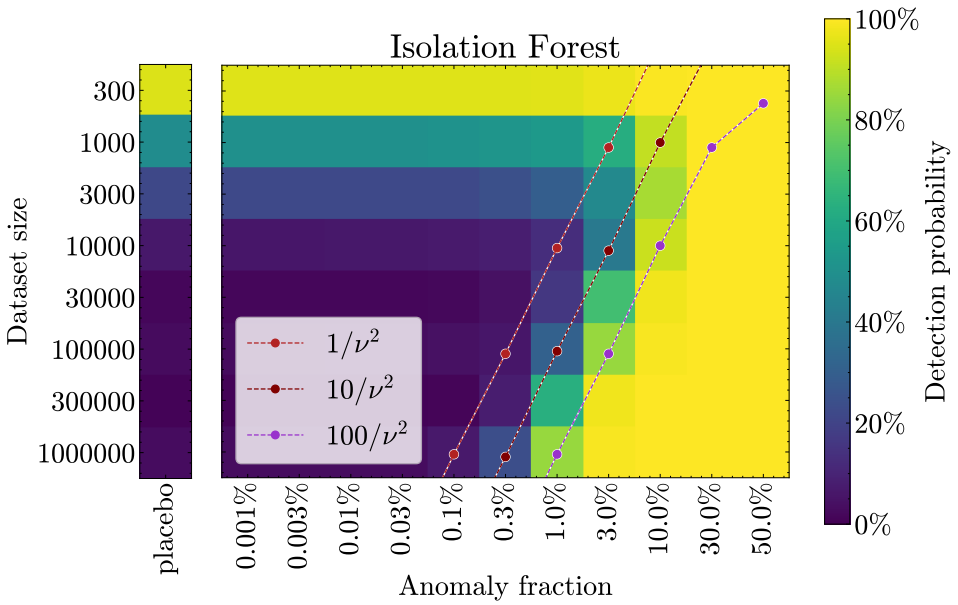


Figure 9.4: Average detection probability using an Isolation Forest as a function of contamination fraction ν and dataset size N. "Placebo" represents the limit of no contamination ($\nu = 0$).

we will show in the remainder of this chapter that the threshold for transition ($center$) depends on N and the algorithm used.

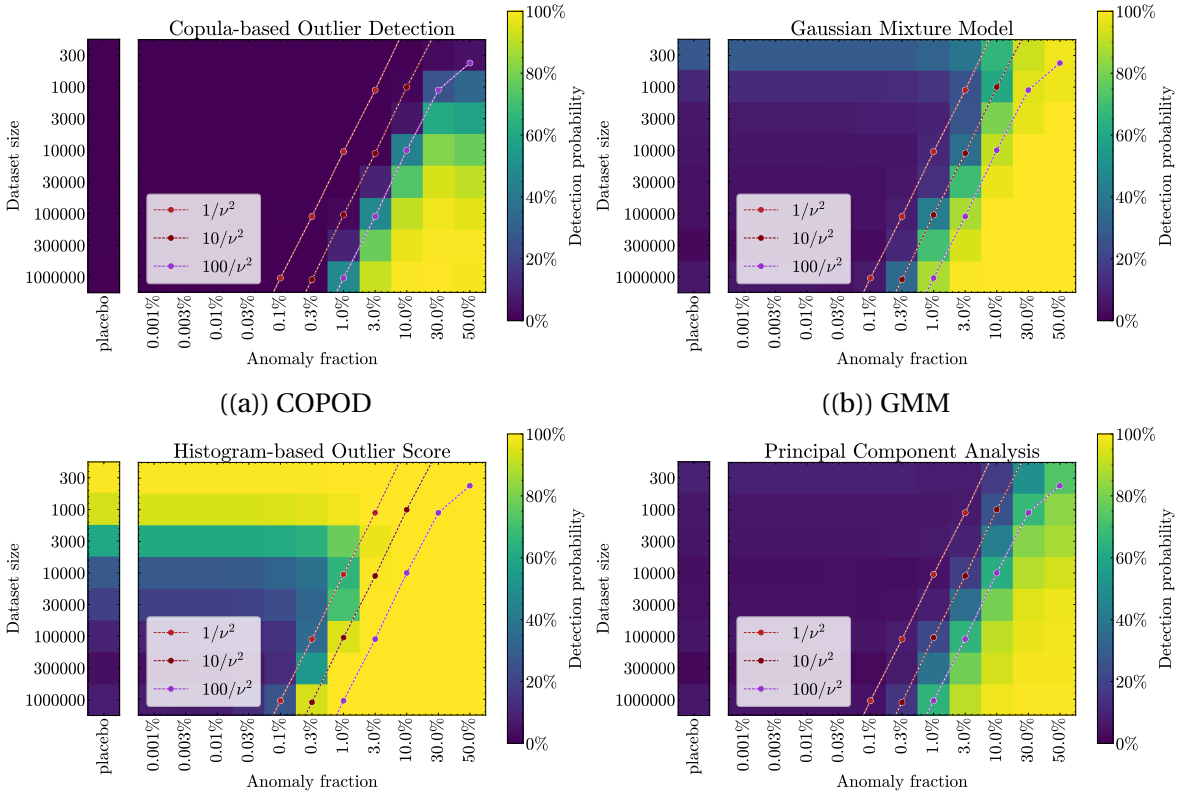
To explore this, we plot the average detection probability as a function of both N and ν for the Isolation Forest (IFor (F. T. Liu et al., 2008)) algorithm in Figure 9.4. We include a "placebo" column, which indicates the likelihood of a false positive at $\nu = 0$. Similar plots for other algorithms used in this study are shown in Figure 9.5.

These plots reveal two main insights. First, for small dataset sizes, the likelihood of false positives (positive statistical tests in the placebo column) is substantial. This implies that small datasets cannot reliably detect distributional differences, as inherent variations exist even between samples from the same distribution. To confirm this, we conducted a complementary experiment with a fixed training set size of one million samples but varying test set sizes (Figure 9.6). As expected, false positives disappear in the larger test sets.

The second insight is that the minimum anomaly fraction required to detect a training-test difference (i.e., $center$ in Equation 9.1) depends on dataset size. Specifically, the 50% decision threshold follows a similar quadratic relation to the one observed in Chapter 8:

$$N \geq \frac{\alpha_{\text{algo}}}{\nu^2} \tag{9.2}$$

where α_{algo} is an algorithm-specific constant. To further explore this dependency, we



((a)) COPOD ((b)) GMM

((c)) HBOS ((d)) PCA

Figure 9.5: Average detection probability as a function of contamination fraction ν and dataset size N for various algorithms used in this study.

examine the average detection probability along these proposed decision boundaries for different algorithms and values of α_{algo} in Figure 9.7. Here, we use a constant training size to remove the effects of false positives.

Similar to Figure 9.3, the detection probability curves generally also fit the logarithmic sigmoid form (Equation 9.1) with low uncertainties along the hypotheses lines, validating Equation 9.2. Some algorithms exhibit higher detection probabilities (lower α_{algo}), with the best (HBOS, $\alpha_{HBOS} \approx 2$) aligning well with our data's linear decision boundaries (Figure 9.2), while the worst (PCA, $\alpha_{PCA} \approx 30$) struggles to find a (usually non-existent) lower dimensional representation. That different algorithms result in different α_{algo}, encoding their ability to detect anomalies well constants suggests that on large, realistic datasets, such significance studies could serve as benchmarks. Notably, compared to the usual anomaly detection metrics, this approach requires no anomaly labels, enabling the benchmarking of algorithms on unlabeled (often significantly larger) datasets. Still, such an analysis might be quite expensive, as we also had to restrict our analysis to only very fast algorithms here.

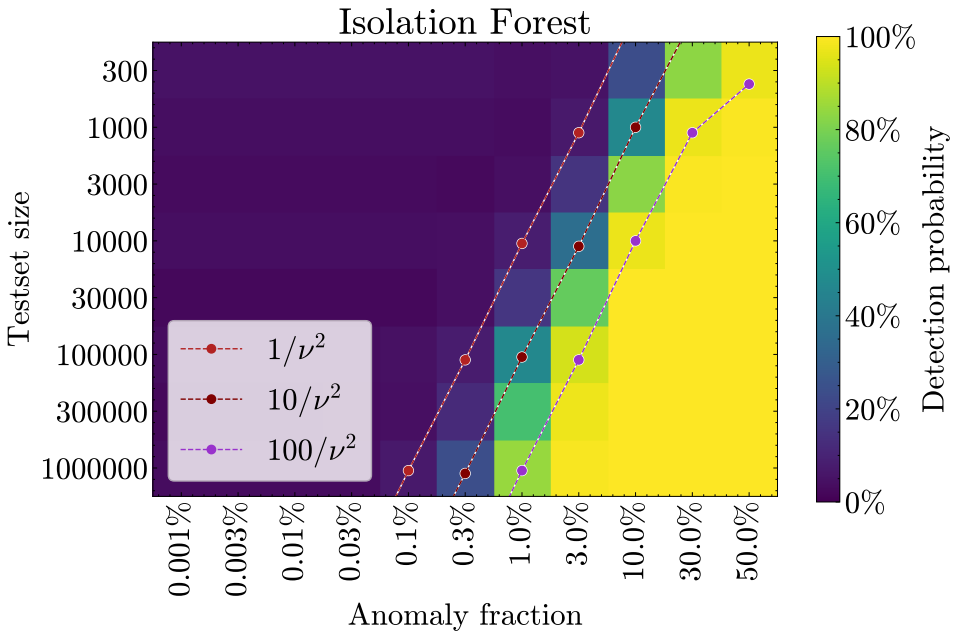


Figure 9.6: Analogous to Figure 9.4 with maximum training set size but varying test set sizes.

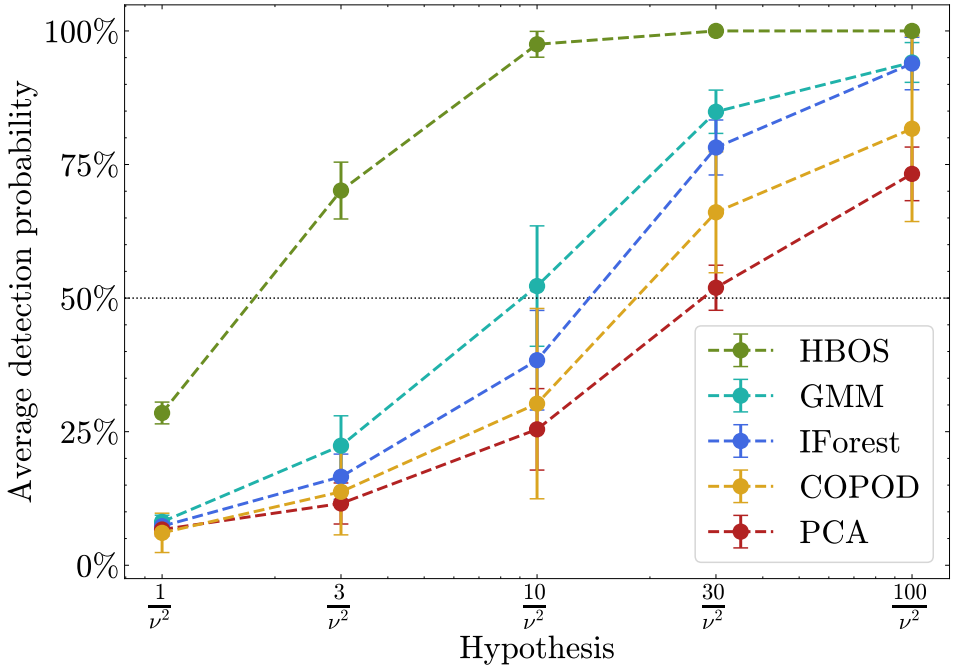


Figure 9.7: Average detection probability for different algorithms and decision boundary hypotheses. Each value represents the average z/color value along the lines shown in Figure 9.6.

Lastly, α_{algo} varies by an order of magnitude, which is relatively stable compared to the five orders of magnitude covered by ν values. Using that α has a limited range, we can estimate the minimum samples needed for 50% detection probability, as shown in Table 9.1.

ν	50%	10%	5%	1%	0.5%	0.1%
$N(\alpha = 1)$	4	100	400	10,000	40,000	1,000,000
$N(\alpha = 10)$	40	1,000	4,000	100,000	400,000	10,000,000
$N(\alpha = 100)$	400	10,000	40,000	1,000,000	4,000,000	100,000,000

Table 9.1: Minimum test samples required for 50% anomaly detection probability across various contamination rates ν and algorithmic quality measures α.

These sample requirements are high, especially for low values of ν. This is a fundamental challenge, as anomalies are typically rare. For example, with $\nu = 1\%$ and an optimistic guess of $\alpha = 1$, at least $N \geq 10{,}000$ samples are needed for 50% detection probability. This limits benchmarking utility in anomaly detection. The benchmark study (Han et al., 2022) used to evaluate our algorithms in previous chapters collected a set of 47 classical benchmark datasets, of which only 14 (30%) contain over 10,000 samples, required for successful detection with $\nu = 1\%$ and $\alpha = 1$. Most of these datasets contain a much higher fraction of anomalies ($\nu \gg 1\%$), as this is beneficial for benchmarking and usually does not affect the ROC-AUC anomaly ranking. However, a contamination rate of $\nu > 10\%$ is not realistic in anomaly detection applications. And even with this unrealistically high fraction of anomalies, only roughly half of datasets fulfill Equation 9.2 for $\alpha = 10$.

9.4.1 Thought Experiment

We further propose a thought experiment to provide a deeper intuition on the relationship between contamination rate and the minimum number of samples needed. For this, we consider a simplified version of our experiment, where anomalies are identifiable if their maximum anomaly score exceeds that of the normal samples. We assume that anomaly scores are generated from two Gaussian distributions, G_N and G_A, with parameters $\mu_A - \mu_N = 2$ and $\sigma_A = \sigma_N = 1$. While scores from the abnormal distribution G_A are generally higher, our test may still fail if the number of normal samples (N_{normal}) significantly exceeds the number of anomalies ($N_{abnormal}$). Thus, a relationship exists between the count of anomalies and the maximum allowable normal samples. Here, we search for this relationship, denoted as $N_{normal} \leq f(N_{abnormal})$.

To achieve this, we draw multiple sets of normal and abnormal samples for given values of $N_{abnormal}$ and N_{normal}, identifying the value of $N_{normal}(N_{abnormal})$ where exactly

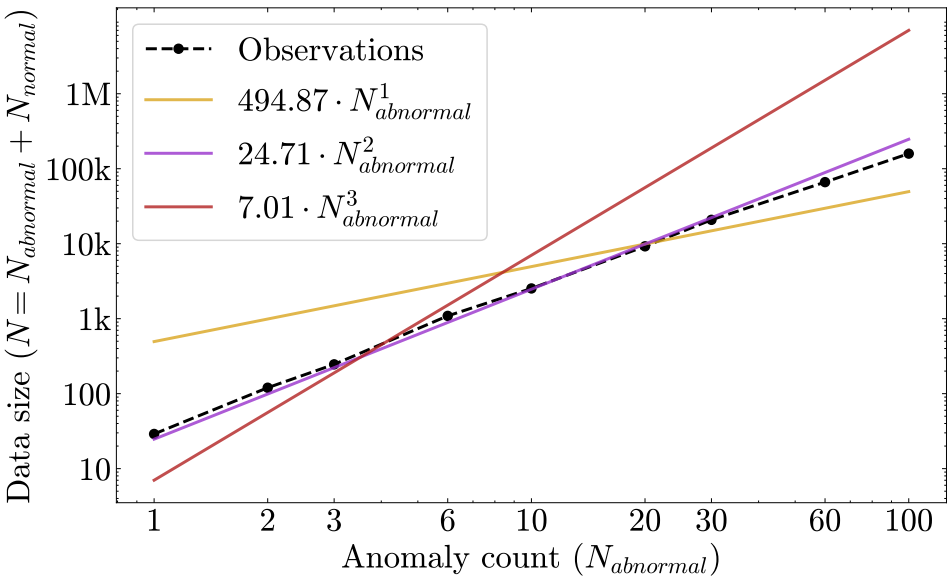


Figure 9.8: Size of the dataset required to solve our thought experiment with a 50% probability. Solving this involves a probabilistic optimization problem, leading to minor inaccuracies. Nonetheless, our measurement appears to closely follow a quadratic relationship.

half of the sets ($\frac{3000}{2} = 1500$) succeed. This is accomplished using a logarithmic binary search. Given the random nature of these variables, this optimization is somewhat imprecise; therefore, we employ a large sample size (3000 distributions) at each optimization step and repeat the optimization to confirm results, finding only negligible differences. We further repeat this search for various values of $N_{abnormal}$, and illustrate the results in Figure 9.8.

From this experiment, the quadratic monomial relationship appears to fit best:

$$N_{\text{abnormal}} + N_{\text{normal}} \leq \frac{1}{\alpha} \cdot N^2_{\text{abnormal}} \tag{9.3}$$

This relationship aligns with our proposed Equation 9.2, suggesting that our found relation is broadly applicable and extends beyond the use of statistical tests.

Proof. Using $\nu = \frac{N_{\text{abnormal}}}{N_{\text{normal}}+N_{\text{abnormal}}}$, we can rewrite Equation 9.2 as follows:

$$N = N_{\text{abnormal}} + N_{\text{normal}} \geq \alpha \cdot \frac{(N_{\text{abnormal}} + N_{\text{normal}})^2}{N^2_{\text{abnormal}}}$$

$$\Rightarrow N^2_{\text{abnormal}} \geq \alpha \cdot (N_{\text{abnormal}} + N_{\text{normal}}) \Leftrightarrow N_{\text{abnormal}} + N_{\text{normal}} \leq \frac{1}{\alpha} \cdot N^2_{\text{abnormal}}$$

□ ■

9.5 Conclusion

In this chapter, we present the first explicit study addressing the fundamental question of whether any anomalies exist within a dataset. Through over three million statistical tests across various anomaly detection tasks, algorithms, and statistical methods, we establish a clear requirement for reliably detecting the existence of anomalies. Given the size of an unlabeled dataset N, the contamination rate ν, and an algorithm-dependent constant α_{algo}, we find that

$$N = N_{normal} + N_{abnormal} \geq \frac{\alpha_{\text{algo}}}{\nu^2} \qquad \Leftrightarrow \qquad N_{\text{normal}} \leq \frac{1}{\alpha_{\text{algo}}} \cdot N_{\text{abnormal}}^2$$

This condition serves as an upper bound on anomaly rarity, beyond which it becomes infeasible to prove the existence of anomalies using the methods considered here.

This also suggests that creating test-time training algorithms like the one proposed in Chapter 8 for low-data scenarios might be impossible, as when we can not even detect the existence of anomalies, it is unlikely that we can optimize their scores. Still, this might be solved by the development of better statistical tests.

For future work, we plan to explore the generality of this relationship further. For instance, we like to conduct user studies to examine whether human cognition aligns with this mathematical law. Depending on the findings, this limitation may be alleviated by advancing weakly supervised anomaly detection, potentially utilizing large, unrelated datasets.

CHAPTER 10

HYPERPARAMETER-AWARE BENCHMARKING

This chapter follows: Klüttermann, S., Gupta, S., & Müller, E. (2025). Evaluating anomaly detection algorithms: The role of hyperparameters and standardized benchmarks [Under review]. *International Joint Conference on Artificial Intelligence (IJCAI)* and is partially based on the work of a thesis student: Joby, J. E. (2024). *Understanding the impact of automated hyperparameter tuning on anomaly detection methods* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/jessia.pdf

10.1 Introduction

While we have proposed, studied, and benchmarked multiple anomaly detection algorithms in the previous chapters, a comparison between them is still missing.

Additionally, while we already used a close-to-standard set of benchmark datasets in the previous chapters, as well as a large number of competitor algorithms, hyperparameter selection remains a critical and underexplored source of variability in anomaly detection research.

So, to conclude our work on anomaly detection, we want to remedy both problems by creating a hyperparameter-aware way of comparing anomaly detection algorithms and using this to also compare our algorithms against stronger competitors.

This is important because, as our results show, hyperparameters can be deliberately optimized to favor one algorithm over others, even when their average performance is comparable. This practice introduces significant bias, potentially distorting conclusions drawn

from benchmarks. And while we have not done this in the previous sections, some of our competitors are still weaker simply because of a bad choice of hyperparameters.

Such a comparison of algorithms in a weakly supervised setting presents interesting challenges. Although hyperparameters can be optimized on specific benchmarking datasets, their performance often fails to generalize to other datasets, where thanks to a lack of labeled data, unsupervised optimization is inherently infeasible. Consequently, many studies (Olteanu et al., 2023) and also our work in previous chapters default to using predefined hyperparameters from the original papers or popular libraries such as PyOD (Y. Zhao et al., 2019). However, as we demonstrate, the impact of suboptimal hyperparameters can rival or exceed the differences between algorithms, thus constraining the field's progress.

In this chapter, we propose a novel pipeline for selecting anomaly detection hyperparameters and use this approach to compare the algorithms proposed in this thesis against stronger competitors. Our approach optimizes hyperparameters using one set of anomaly detection datasets and evaluates performance on a disjoint set of datasets to ensure generalizability. To support this, we curate the most extensive anomaly detection dataset collection to date, comprising thousands of datasets, of which 498 are used in this chapter because of computational limitations.

10.2 Hyperparameters for Unsupervised and Weakly supervised Anomaly detection

Like for most machine learning approaches, hyperparameters also significantly influence the performance of anomaly detection methods. Still, especially for unsupervised and weakly supervised applications, directly optimizing hyperparameters is close to impossible. This is because feedback for fine-tuning a given setup is unavailable without labeled anomalies.

Consequently, while numerous anomaly detection algorithms exist, along with studies that compare their effectiveness (S. S. Khan & Madden, 2010; Olteanu et al., 2023; Ruff et al., 2021), these comparisons are inherently limited. They must often assume the use of suboptimal hyperparameters, which can skew performance evaluations.

The most practical approach to date has been to rely on expert-selected hyperparameters during algorithm implementation, often guided by recommendations in the original research papers. For example, the popular anomaly detection library PyOD (Y. Zhao et al., 2019) adopts this approach by using default hyperparameters based on the original papers. However, as we will demonstrate, these default settings are far from optimal and an optimized set of hyperparameters would be preferable.

Although extensively studied in the supervised setting, leading to advancements such as AutoML frameworks (Karmaker et al., 2020) like AutoSklearn (Feurer et al., 2015), hyperparameter optimization still remains an underexplored area in unsupervised and weakly supervised tasks like anomaly detection. AutoML methods typically optimize a well-defined metric, such as accuracy, that quantifies model performance. However, in the absence of labeled data and a quality metric for anomaly detection that does not require labels, most existing methods are inapplicable.

For unsupervised and weakly supervised tasks, one viable strategy is one-shot optimization, where an optimal set of hyperparameters is suggested outright rather than iteratively optimized. MetaOD (Y. Zhao, Rossi, & Akoglu, 2021) represents an approach in this direction, using neural networks to recommend the best combination of hyperparameters and algorithms from a limited predefined set for a specific task. While promising, this method has notable limitations. Its complexity, restricted hyperparameter search space, and reliance on retraining for new algorithms make it less suitable for comprehensive benchmarking studies. Furthermore, its adaptability to new algorithms is constrained by the need for extensive retraining.

In contrast, we advocate for a simpler and more versatile approach: identifying a single robust set of hyperparameters for each algorithm. This approach offers several advantages: it simplifies the benchmarking process, minimizes the risk of overfitting hyperparameters to specific datasets, and allows direct optimization for individual algorithms. Although a different set of hyperparameters might perform better on specific datasets, a singular configuration provides a reproducible and broadly applicable standard for evaluation and is significantly easier to use.

Previous studies have explored hyperparameter optimization for specific algorithms. For example, there is work on optimizing the KNN-based anomaly detection algorithm (Ramaswamy et al., 2000) and selecting the kernel function for OCSVM (Budynkov & Masolkin, 2017). However, these studies focus on a small number of hyperparameters or specific algorithms. In contrast, this chapter systematically explores over 80 hyperparameters spanning 13 widely used algorithms, providing a more comprehensive perspective on hyperparameter optimization for anomaly detection.

10.3 Methodology

Our methodology is summarized in Figure 10.1. We employ two sets of datasets: one set (**A**) is used to optimize hyperparameters, and the other set (**B**) is used to evaluate the per-

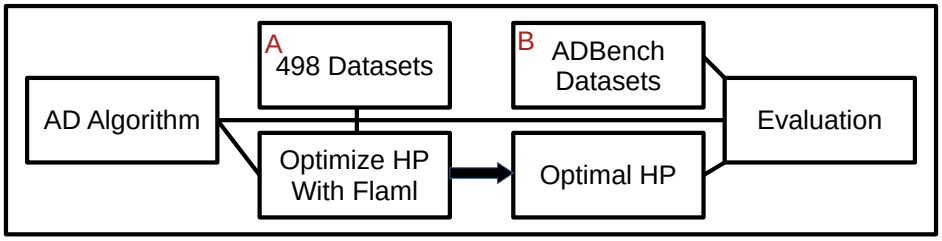


Figure 10.1: Overview of our methodology. We optimize the hyperparameters of an algorithm on a large set of anomaly detection datasets and evaluate them on the unrelated ADBench datasets.

formance of these hyperparameters. To optimize hyperparameters on set **A**, we employ `flaml` (C. Wang et al., 2021), a lightweight framework that supports custom evaluation functions for hyperparameter optimization. The optimization objective is to maximize the average ROC-AUC across all datasets in set **A**.

Some hyperparameter configurations may be erroneous on certain datasets (e.g., a learning rate that is too high may cause a neural network to produce NaN weights). To address this, we assign a score of -1 to the ROC-AUC of any dataset where an algorithm fails with a given hyperparameter configuration. Since `flaml` uses an evolutionary approach to optimization, this feedback helps it avoid non-functional configurations in subsequent iterations.

We allocate ten CPU cores for each algorithm's hyperparameter optimization, running for approximately one week of CPU time per algorithm to ensure a fair comparison. The hyperparameter search space is defined in Section 10.4, with a focus on incorporating as many relevant hyperparameters as possible, alongside a wide but reasonable range for each. Categorical variables (e.g., activation functions in neural networks) are handled separately, while numeric values are sampled uniformly or logarithmically, depending on the parameter. Logarithmic sampling is particularly beneficial for parameters like the number of training epochs, where differences are more pronounced at smaller scales (e.g., 1 vs. 2 epochs) than at larger scales (e.g., 100 vs. 101 epochs).

`flaml` prioritizes evaluating faster configurations early in the optimization process, which introduces a slight bias toward speedier hyperparameter settings. We consider this a feature rather than a limitation, as it aligns with our goal of identifying efficient and effective configurations.

After completing the optimization, we evaluate the optimized hyperparameters on the 121 datasets proposed by ADBench (Han et al., 2022) as set B. This evaluation compares the optimized performance against the performance achieved using default parameters. For

default settings, we rely on the parameters provided by the respective implementations.

10.3.1 Datasets

For our analysis, we require a large and diverse set of anomaly detection datasets to ensure that our results effectively generalize to new datasets. Moreover, these datasets must be independent of those used for evaluation (ADBench), as reusing evaluation datasets during optimization could unfairly bias the results. We prioritize real-world datasets over synthetic ones, as the latter often introduce human biases in their creation. However, constructing a substantial collection of real-world datasets is challenging and costly, given the rarity of anomalies and the fact that most widely used datasets are already part of ADBench.

To address this, we generate new anomaly detection datasets from classification datasets by treating one class as normal and another class as anomalous. This approach is commonly used in the literature (Ruff et al., 2018), for instance, in image-based anomaly detection using datasets such as CIFAR-10 (Krizhevsky, 2009), and is also part of the ADBench evaluation set. It can be argued that classification datasets exhibit different distributions compared to true anomaly detection datasets since anomalies may not naturally form clusters. We somewhat mitigate this by using a large and diverse set of datasets, but it likely would be possible to construct a better set of datasets given unlimited resources.

We retrieve datasets using the Kaggle API ("Kaggle," n.d.), resulting in an initial collection of approximately 10,000 datasets. These are filtered to retain only classification datasets with tabular data. To transform these into anomaly detection datasets, we preprocess them as follows:

- Remove string or categorical features with more than 10 unique values, and one-hot encode the remaining categorical features.
- Handle missing values by removing samples with rare missing values ($< 1\%$), and otherwise removing the affected features.
- Split datasets based on their last categorical feature, treating one group as the normal class and each remaining group as anomalous.

After preprocessing, we validate each generated dataset by ensuring it achieves an ROC-AUC score of at least 0.55 using a simple isolation forest (F. T. Liu et al., 2008), guaranteeing that each dataset contains meaningful information. This process results in 3,174 datasets of various sizes.

For hyperparameter optimization, we further filter the datasets:

- Restrict to datasets with no more than 100 features and 1 million samples.
- To avoid redundancy, select only one random subset per original dataset when multiple subsets are generated.

This yields a final pool of 498 datasets for hyperparameter optimization. While this large number of datasets is useful to generate very general results, using more datasets also limits how many different hyperparameter configurations can be evaluated.

To investigate this trade-off, we conduct our optimizations twice: once using all 498 datasets (**L**arge) and once using only the 170 datasets with a size smaller than 100KB (**S**mall).

10.4 Results

This section presents the outcomes of our hyperparameter optimization strategy. We detail the optimized hyperparameter configurations for the 13 analyzed algorithms and provide a comprehensive statistical comparison of their performance both before and after optimization to the algorithms proposed in this thesis. By systematically evaluating the impact of hyperparameter tuning, we highlight its critical role in improving algorithmic efficacy and ensuring better benchmarking practices.

First, we summarize the performance of both the baseline hyperparameter set and the optimized sets for the small and large datasets across all algorithms in Figure 10.2. For clarity, we highlight hyperparameter sets that outperform the baseline in green and those that perform worse in red. Additionally, columns that do not show statistically significant differences, as determined by a Wilcoxon-Friedman test (Wilcoxon, 1945), are struck through.

Additionally, we show the performance of the three general anomaly detection algorithms proposed in this thesis, DEAN (Chapter 4), SEAN (Chapter 5), and DOUST (Chapter 8) as horizontal lines. While it might technically also be possible to optimize the hyperparameters of our algorithms, various effects make this infeasible. First, considering SEAN, we are able to easily find slightly better hyperparameters (resulting in a significant, roughly 1% higher performance). However, this performance is mainly the result of a higher number of submodels and thus comes at the cost of a significantly higher runtime and would contradict the design philosophy of SEAN. We have also already studied hyperparameters like this in Section 5.4.4. A similar effect happens for DEAN, resulting in very large ensembles and higher computational costs. But here, this also results in very few optimizations finishing in the previously described uniform allowed computational resources (less than 100 for both the Small and the Large case), and with the high number of possible hyperpa-

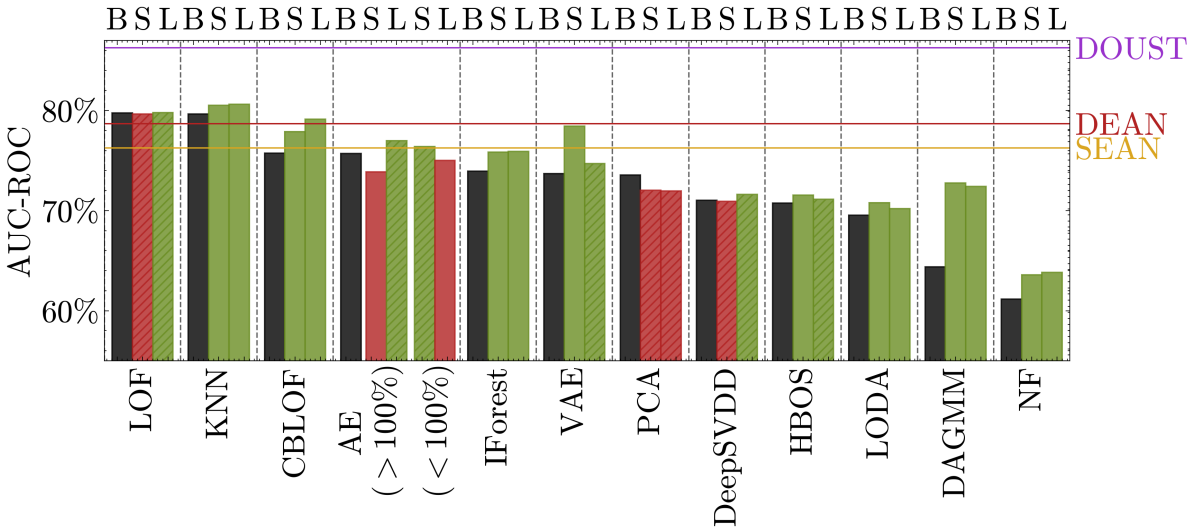


Figure 10.2: Comparison of average performance before and after optimization. Each algorithm is represented by three columns corresponding to the **B**ase performance, and the performance after the **S**mall and **L**arge optimization loops. Hyperparameter configurations that improve upon the base performance are highlighted in green, while those that do not are marked in red. Additionally, a Wilcoxon-Friedman test is conducted, and columns that do not exhibit statistically significant differences to the baseline performance at $p = 0.05$ are struck through. We show the performance of the algorithms proposed in this thesis as horizontal lines.

rameters, no useful solution is found. Finally, running a DOUST model is error-prone, and thus, it is likely that DOUST fails on certain datasets. This is true for multiple of the deep learning algorithms considered here, and our approach is simply to punish such datasets with an AUC score of -1, hopefully resulting in flaml's optimization reducing the number of faulty hyperparameters. But as such failures are relatively common for the DOUST case, the optimization simply finds such hyperparameters that minimize their occurrence (e.g. training for a single epoch in each phase), and this does not result in an effective anomaly detector.

Comparing the performance of our algorithms to their competitors, we first find that DOUST still performs drastically better than each competitor, even with optimized competitors. It should, however, be noted that we evaluate here each algorithm on test sets that contain 50% anomalies, and thus, the necessary condition for DOUST to work (as studied in Chapter 8 and Chapter 9) is likely always fulfilled. Comparing both DEAN and SEAN, we see that there are some competitors that sometimes outperform us only after hyperparameter optimization. CBLOF improves for such an amount that it has a higher average AUC than SEAN, and after the more effective optimization, even DEAN. Additionally, there is a

hyperparameter configuration for each autoencoder variation that can outperform SEAN. It should be noted though, that this is mostly just a result of the low number of submodels used, especially for SEAN and that a larger submodel count, as studied in Section 5.4.4 still outperforms each of these improved hyperparameters.

Considering our optimization further, 14 out of 26 hyperparameter sets significantly outperform the baseline, while an additional 6 sets show improvement, albeit without statistical significance. Furthermore, on 12 out of the 13 algorithms, at least one optimized hyperparameter set outperforms the baseline, and on 8, both optimized sets achieve this. Unfortunately, the PCA algorithm does not benefit from hyperparameter optimization, a topic we will discuss further below as the algorithm also behaves quite strangely overall. While optimization is not perfect for every algorithm, our optimization generally improves an algorithm's performance. The three highest observed performances of competitor algorithms were achieved after optimization, and the differences between hyperparameter sets often exceed those between algorithms, even when comparing fundamentally different methods. For example, the improvement of IForest and VAE is drastically larger than the difference between both, underscoring the importance of hyperparameter optimization over algorithm development in weakly supervised learning tasks.

Consistent with recent benchmark studies (Han et al., 2022; Livernoche et al., 2024) and our experiments in previous chapters, we find that classical anomaly detection algorithms, such as LOF, KNN, and CBLOF, outperform more recent deep learning approaches when evaluated with baseline parameters. Interestingly, this gap diminishes when considering optimized hyperparameters, with three variants of Autoencoders outperforming CBLOF. This trend is expected, as deep learning algorithms typically involve many more hyperparameters that require tuning and have less theoretical research to guide the choice of optimal values (Ramaswamy et al., 2000). However, the larger number of hyperparameters also results in less efficient optimization. In this study, we allocated an equal amount of optimization time for each algorithm to ensure fairness and to limit computational demands. However, since neural network-based algorithms are often slower, this results in fewer hyperparameter combinations being considered here.

This number of hyperparameter combinations evaluated by each algorithm is visualized in Figure 10.3. While the hyperparameter space of some algorithms, such as PCA and IForest, is well explored, others, like NF and SOD, are approximately 100 times slower, leading to less thorough exploration of their respective hyperparameter spaces. Allowing for a maximum number of optimization steps could be a more practical approach, but computational constraints make this difficult.

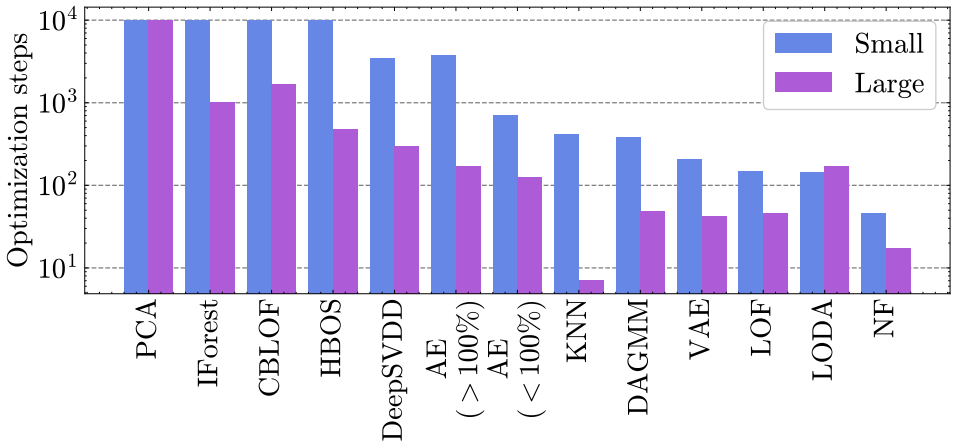


Figure 10.3: Number of hyperparameter combinations explored during optimization. Due to the imposed maximum time cost of one week, a smaller number of optimization steps corresponds to a slower algorithm. Additionally, a maximum of 10,000 optimization steps was set, as visible for the first four algorithms.

Table 10.1: Hyperparameters LOF algorithm

Hyperparam	Options	Base	Small	Large
n_neighbors	1 - 1000	20	5	8
p	1 - 6	2	1	1
algorithm	auto / kd_tree / brute / ball_tree	auto	ball-tree	ball-tree
leaf_size	10 - 50	30	13	10
ROC-AUC		0.7976	0.7967	0.798

Table 10.2: Hyperparameters KNN algorithm

Hyperparam	Options	Base	Small	Large
n_neighbors	1 - 1000	5	1	1
method	mean / max / median	max	max	max
p	1 - 6	2	5	1
ROC-AUC		0.7967	0.8052	0.8061

10.4.1 Individual algorithm performances

Next, we discuss the optimized hyperparameters for each algorithm, beginning with LOF (Breunig et al., 2000), as shown in Table 10.1. The performance of both optimization strategies is comparable to the base performance. However, the optimized hyperparameters exhibit notable similarity, with a slightly reduced number of neighbors, a Manhattan distance metric, the Ball Tree algorithm, and a lower value for the leaf_size parameter.

Table 10.3: Hyperparameters CBLOF algorithm

Hyperparam	Options	Base	Small	Large
n_clusters	1 - 100	8	87	99
α	0.1 - 0.9	0.9	0.709	0.9
β	1.0 - 5.0	5	1.4782	1.4395
use_weights	✓ / ✗	✗	✗	✗
ROC-AUC		0.7576	0.779	0.7914

For KNN (Gu et al., 2019), as shown in Table 10.2, the optimized hyperparameters are quite similar across both optimization strategies, particularly for the number of neighbors and the method used. Notably, the number of neighbors aligns with theoretical research (Ramaswamy et al., 2000) for uncontaminated training sets (it is important to note that we use ADBench default parameters as baselines). However, there is some discrepancy between the optimizations regarding the choice of distance metric (p). Despite this, the performance remains largely unaffected, with both optimizations outperforming the baseline. The Large optimization variant overall achieves the highest average ADBench anomaly score observed in this study.

Continuing with CBLOF (Z. He et al., 2003), as shown in Table 10.3, most of the optimized parameters also show consistent results across both optimization strategies. A smaller β value than the baseline is preferable, and increasing the number of clusters also improves performance. This second observation is perhaps not surprising, as with a high number of clusters, possibly approaching the number of samples, the cluster distances calculated by CBLOF become very similar to those used in LOF and KNN, which, as shown in Figure 10.2, are more effective algorithms for the tasks studied here. Using this approach, CBLOF is also able to outperform both the configurations of DEAN and SEAN shown here.

Our first deep learning algorithm, the Autoencoder (Sakurada & Yairi, 2014), presented in Table 10.4, involves significantly more hyperparameters than the previously studied algorithms. This likely accounts for the considerable variation in optimized hyperparameters. Additionally, inspired by a paper (Yong & Brintrup, 2022), we consider two separate cases: an undercomplete autoencoder, where we restrict the latent size to be smaller than the number of features, and an overcomplete autoencoder, where the latent size needs to exceed the number of features. In both cases, the latent size is represented as a factor relative to the number of features. Since the ADBench implementation does not support the overcomplete case, we implemented our own version. We use here the abbreviation "(act)" to refer to a set of common activation functions (ReLU, tanh, sigmoid, softmax, linear).

Although not every hyperparameter set outperforms the baseline, and significant dis-

Table 10.4: Hyperparameters AE algorithm

Hyperparam latent_size	Options	Base	Small < 100%	Large < 100%	Small > 100%	Large < 100%
latent_size_factor	0.1 - 1.0 - 10.0	(32)	0.2041	0.463	3.1757	1.5514
layers	1 - 7	3	1	2	2	1
batch_size	16 - 128	32	23	29	123	103
learning_rate	1e-5 - 0.01	1e-3	3.92e-3	2.16e-3	7.55e-3	2.07e-3
epochs	100 - 500	10	496	141	191	157
encoder_activation	(act)	relu	tanh	tanh	relu	relu
decoder_activation	(act)	relu	softmax	relu	sigmoid	relu
output_activation	(act)	linear	linear	linear	linear	linear
encoder_dropout	0.05 - 0.99	None	0.05	0.11	0.05	0.05
decoder_dropout	0.05 - 0.99	None	0.99	0.2155	0.05	0.4799
evaluation_metric	1 - 6	2	2	4	1	4
loss	mse / bce	mse	mse	mse	mse	mse
regularisation	l1/l2	l1	l2	l2	l2	l2
regularizer	0.0001 - 0.5	0	6.29e-4	1e-4	1e-4	2.08e-4
ROC-AUC		0.7572	0.7387	0.7700	0.7640	0.7500

Table 10.5: Hyperparameters IFOR algorithm

Hyperparam	Options	Base	Small	Large
n_estimators	1 - 1000	100	356	230
bootstrap	✓ / ✗	✗	✗	✗
max_features	0.1 - 1.0	1.0	0.9699	0.831
max_samples	0.1 - 1.0	auto	1.0	1.0
ROC-AUC		0.7393	0.7586	0.7591
AUC ($n = 1k$)		0.7412	0.7598	0.7592

agreement exists between the optimized parameters, some conclusions can still be drawn from the optimization process.

First, it remains inconclusive whether an overcomplete or undercomplete autoencoder is preferable, supporting the claim made in Reference (Yong & Brintrup, 2022). The best and worst performance are both observed with an undercomplete autoencoder, while the overcomplete autoencoder tends to be slightly faster to evaluate (see Figure 10.3). In both cases, two dense layers for both the encoder and decoder appear optimal, and the training should be extended well beyond the ten epochs suggested by the ADBench parameters, although the improvement is marginal. Additionally, regularization does not seem beneficial.

The Isolation Forest (F. T. Liu et al., 2008), as shown in Table 10.5, appears to have a high degree of agreement between the optimized hyperparameters, with the number of

Table 10.6: Hyperparameters PCA algorithm

Hyperparam	Options	Base	Small	Large
n_components	0.1 - 1.0	1.0	0.1523	0.1661
n_selected	0.1 - 1.0	1.0	0.3558	0.3877
ROC-AUC		0.7356	0.7205	0.7198

estimators being roughly three times higher as the only notable change. This raises the question: why are more estimators not considered? It is well-established (See C. Aggarwal (2012) and Chapter 3) that for unsupervised or weakly supervised ensembles like Isolation Forest, more submodels generally lead to better performance. During optimization, similar hyperparameter configurations with more submodels have been considered, and they would slightly increase the ADBench performance, as shown in the last row of Table 10.5.

We believe the observed results are due to overfitting of our optimization algorithm. Since the Isolation Forest is non-deterministic, performance can vary slightly depending on random choices. Although there are few parameters to optimize, each leads to a new evaluation with a different random seed, allowing for slightly better or worse-performing runs. Thus, increasing the number of submodels improves performance, albeit with diminishing returns, while also reducing uncertainty and thus allowing the optimizer to extract performance from random changes less efficiently. A smaller number of estimators represents the optimal trade-off between the benefits of larger ensembles and the potential overfitting caused by the random seed. This suggests that maybe a smaller number of optimization steps might have been beneficial. Nevertheless, our results also indicate that the overfitting effects are minimal.

The results for PCA (Callegari et al., 2011), presented in Table 10.6, are intriguing. The optimized performance is worse than the baseline configuration. However, the baseline parameters themselves seem questionable: a PCA model, which has access to all features and utilizes the full reconstruction, should theoretically achieve a perfect reconstruction, leading to a reconstruction error of 0 when used as an anomaly score. We do not have a satisfactory explanation for this unexpected performance, but it may be that similar to CBLOF, this limitation becomes comparable to the performance of another, more effective algorithm.

DeepSVDD (Ruff et al., 2018), as presented in Table 10.7, shares many hyperparameters with the previously discussed autoencoder. In addition to the same shortcut "act" for activation functions, we also introduce a shortcut for the number of hidden neurons, denoted as "neurons," which we optimize. To ensure reasonable optimization time, we restrict the possible neural network architectures to a fixed set of configurations: $20-20$, $10-10$, $64-32$,

Table 10.7: Hyperparameters DeepSVDD algorithm

Hyperparam	Options	Base	Small	Large
batch_size	16 - 128	32	25	91
epochs	50 - 200	100	51	109
dropout_rate	0.05 - 1	0.2	0.05	0.05
h._neurons	(neurons)	64-32	64-32	20-10-3
h._act	(act)	relu	relu	relu
output_act	(act)	sigmoid	linear	sigmoid
l2_regularizer	1e-4 - 0.5	0.1	8.64e-4	7.44e-3
optimizer	sgd / adam	adam	adam	adam
preprocessing	✓ / ✗	✓	✗	✓
use_ae	✓ / ✗	✗	✗	✓
ROC-AUC		0.7103	0.7161	0.7095

Table 10.8: Hyperparameters LODA algorithm

Hyperparam	Options	Base	Small	Large
n_bins	auto / 5 / 10 / 20	10	10	10
n_random_cuts	10 - 1000	100	541	726
ROC-AUC		0.6954	0.708	0.7019

Table 10.9: Hyperparameters HBOS algorithm

Hyperparam	Options	Base	Small	Large
alpha	0.01 - 1.0	0.1	0.01	0.01
n_bins	0 / 5 / 10 / 20	10	5	20
tol	0.1 - 1.0	0.5	0.1	0.1016
ROC-AUC		0.7076	0.7154	0.7116

$20-10-5$, $20-10-3$, $30-20-10-3$, $128-64-32$, and $20-10-3-10-20$. These relatively small configurations were chosen to maintain optimization efficiency and to align with the typically low-dimensional datasets used in this study. Upon reviewing the results, it appears that the baseline parameters are already quite well-chosen, with the only notable differences being a lower dropout rate and regularization rate. However, these adjustments do not significantly affect the overall performance.

The LODA (Pevný, 2016) algorithm, as shown in Table 10.8, has two primary parameters. While the number of bins appears to be well-chosen, increasing the number of random cuts slightly enhances performance.

The HBOS (Goldstein & Dengel, 2012) algorithm in Table 10.9 behaves similarly, with a smaller alpha and tol(erance) value leading to a slight performance improvement.

Table 10.10: Hyperparameters NF algorithm

Hyperparam	Options	Base	Small	Large
K	2 - 100	10	15	15
batch_size	16 - 128	64	16	16
epochs	10 - 500	200	499	165
lr	1e-05 - 0.01	2e-3	4.05e-4	2.29e-4
ROC-AUC		0.6116	0.6359	0.6382

Table 10.11: Hyperparameters VAE algorithm

Hyperparam	Options	Base	Small	Large
batch_size	16 - 128	32	18	26
dropout_rate	0.05 - 0.99	0.2	0.0537	0.0593
epochs	100 - 500	100	266	142
hidden_act	(act)	relu	relu	tanh
l2_regularizer	0.0001 - 0.5	0.1	0.5	0.1012
latent_dim	0.1 - 10.0	(2)	1.2028	6.6883
num_features	1 - 100	4	49	2
output_act	(act)	sigmoid	tanh	tanh
ROC-AUC		0.7371	0.7846	0.7471

The Normalizing Flow (Rezende & Mohamed, 2015) algorithm, presented in Table 10.10, is less straightforward. Despite being a neural network-based method, the constraints of normalizing flows limit the number of tunable parameters. In our experiments, adjustments such as a lower learning rate and smaller batch size result in more careful training, which, when combined with a more complex distribution (higher K), leads to a modest performance improvement. However, the performance is quite low when compared to other algorithms, aligning with some theoretical findings limiting the value of normalizing flows for such tasks (Kirichenko et al., 2020).

The Variational Autoencoder(Kingma & Welling, 2014), shown in Table 10.11, exhibits one of the largest performance improvements after optimization. The difference of 4.75% between the Small and Base versions is notably larger than the difference between LOF and AE, our best and fourth-best algorithms. This improvement could be attributed to the fact that the base version uses a fixed latent dimension of 2, whereas our approach employs a latent dimension relative to the number of features, similar to the autoencoder optimization. While we no longer distinguish between overcomplete and undercomplete versions, the results clearly indicate that an overcomplete version is preferable. Moreover, our optimization suggests a lower dropout rate, a significantly increased number of features, and strong regularization. We believe that further investigation is warranted, as this relatively

Table 10.12: Hyperparameters DAGMM algorithm

Hyperparam	Options	Base	Small	Large
batch_size	16 - 128	256	35	53
lambda_cov	0.01 - 1.0	0.005	0.2004	0.1617
lambda_energy	0.01 - 1.0	0.1	0.0662	0.01
latent_dim	1 - 25	1	7	8
lr	1e-5 - 0.01	1e-4	2.69e-3	1.42e-4
lr_milestones	0 - 1	(50)	61.93%	92.66%
n_gmm	1 - 25	4	6	24
num_epochs	100 - 500	200	107	142
patience	5 - 200	50	35	35
ROC-AUC		0.6438	0.7276	0.7243

strange set of hyperparameters leads to the highest performance among all deep learning algorithms.

An even higher improvement of 8.38% is observed for the DAGMM (Zong et al., 2018) algorithm, as shown in Table 10.12. However, the performance of DAGMM is less remarkable overall, as the baseline performance is not particularly impressive. Similar to the VAE, this may be due to the suboptimal choice of baseline parameters, with nearly every hyperparameter undergoing drastic changes after optimization.

10.4.2 Statistical comparison

With these optimized parameters, we can now compare our proposed algorithms against more effective competitors. To do this, we use critical difference plots to visualize the results of a Wilcoxon test (Wilcoxon, 1945). We consider p-values below $p \leq 5\%$, after applying the Bonferroni-Holm correction (Holm, 1979), as significant.

Figure 10.4 shows three different variations. The top plot compares the performance of the algorithms with baseline parameters, the middle plot contains the performance after optimization (average of the Small and Large optimization), while the bottom plot compares performance using the best set of hyperparameters identified in this study. With these optimized parameters, we observe a notable change in the ordering of the algorithms, with some, such as DAGMM and VAE, experiencing significant shifts in their rankings.

In each set, our DOUST algorithm is the best-performing one, with an average rank below 2 before optimization and between 2 and 3 after optimizing the competing algorithms. While the difference to the next best algorithm, KNN, with an average rank between 4 and 5 in each comparison, is large, it is still not significant. This is different from our experiments

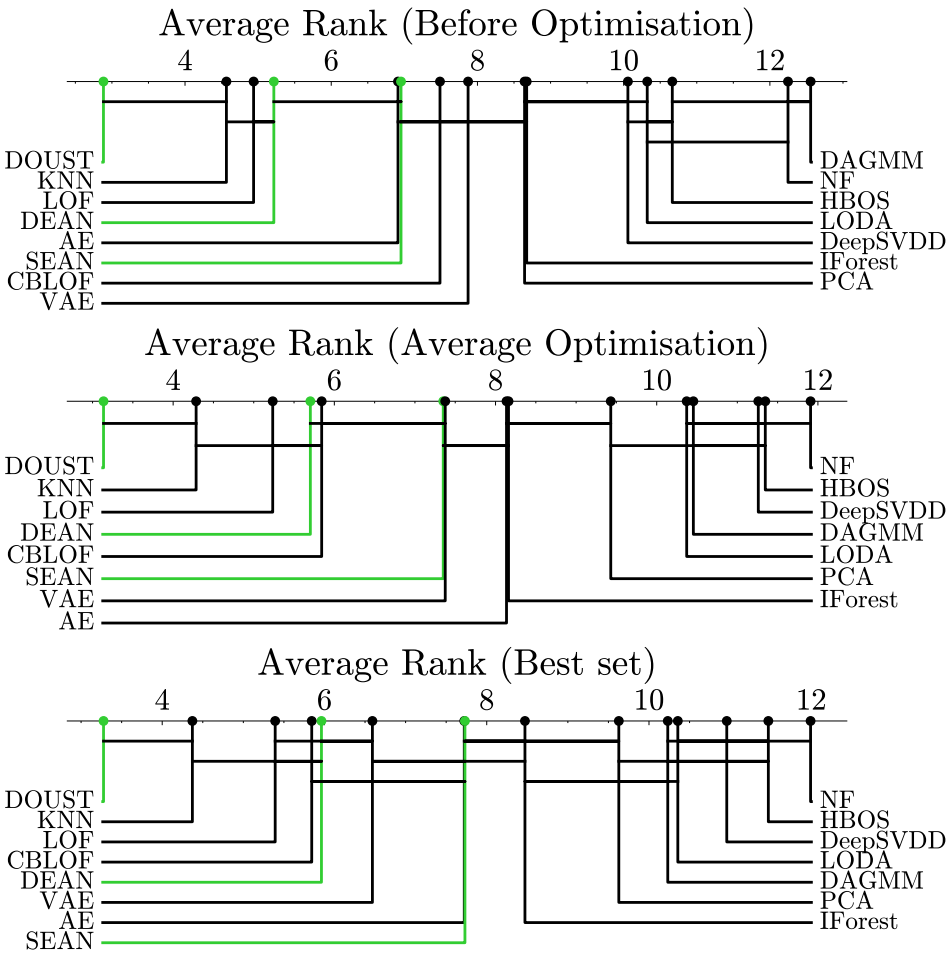


Figure 10.4: Critical difference plot on ADBench data comparing the algorithms used in this study. The top plot shows performance with baseline parameters, the middle one shows the average performance after optimization and the bottom plot shows performance with the best set of hyperparameters found in this chapter. We include the algorithms proposed in this thesis, without optimized hyperparameters in green.

in Chapter 8, likely because of the larger number of competitor algorithms.

When optimizing the hyperparameters of CBLOF, this algorithm is able to slightly outperform DEAN, and the same is true for autoencoder variations and SEAN. Still, these differences are minor, and a larger number of submodels in our ensemble approaches can reverse this trend.

Additionally, even the worst-performing algorithm proposed in this thesis, SEAN, which is more optimized for speed than performance, is still able to outperform at least 7 out of 12 competitors in each comparison while being the fastest algorithm considered here. Alternatively, when intentionally optimized for performance over runtime (Section 5.4.4), it only performs worse than two competitors and DOUST. This proves the potential of ab-

stract ensemble methods for anomaly detection.

10.5 Conclusion

In this chapter, we introduce a novel and effective methodology for optimizing hyperparameters in weakly supervised anomaly detection. We apply this methodology to 13 commonly used anomaly detection algorithms and observe performance improvements in the majority of cases. We also use it to compare the algorithms previously proposed in this thesis and show that even with optimal competitor configurations, our algorithms are still highly competitive.

Particularly for deep learning algorithms, we see notable gains when using the optimal hyperparameters, underscoring the importance of careful hyperparameter tuning. Furthermore, our work provides valuable insights into how these algorithms can be better utilized and, in some cases, might serve as a foundation for the development of new algorithms.

Part IV

Abstraction beyond Anomaly Detection

After studying the benefits of abstract ensemble techniques for anomaly detection in the previous sections, we want to now study if it is possible to extend this approach to other machine learning tasks. For this, we choose to study re-identification, a subset of contrastive learning, where we task a machine learning model to recognize which images are taken of the same object. It is widely used, for example, in person re-identification to track people through surveillance cameras (M. Ye et al., 2022). Further applications include animal tracking (Cermak et al., 2024) or signature verification (Z. Wang et al., 2023).

Abstraction-based methods for re-identification tasks are especially interesting, as most re-identification approaches rely on very complicated neural networks (Fu et al., 2021) and simple more abstract methods are rarely studied (M. Ye et al., 2022). At the same time, re-identification is often used on very large scales (like the video surveillance of an entire district), and thus, even slightly more efficient methods would be very useful. Additionally, re-identification requires generalization; as, for example, a neural network identifying only a large number of known people is not helpful since the set of possible people grows and changes frequently. Instead, re-identification uses contrastive learning to find a representation in which images of the same subject are close to each other, hoping that a similar representation learning methodology also works for a different set of objects/persons. And especially for large changes (like for example when everybody starts wearing face-masks), this is far from a solved problem (Kantarcı et al., 2024). Here abstract methods that theoretically should generalize more easily could be extremely useful.

Our work towards this is done in tandem with Jérôme Rutinowski, who works for the logistics department at TU Dortmund. He is working on re-identifying pallet blocks using their unique press wood pattern to hopefully track the objects that are carried on them. Working with such novel patterns is perfect for studying abstraction in re-identification since this task is much less well developed than, for example, person re-identification, and many methods developed for such tasks are simply impossible through the lack of large datasets and pre-trained models (Fu et al., 2021). Additionally, pallet blocks contain very few obvious features compared to other tasks, where, for example, the way a person is moving can be used to identify them (Nambiar et al., 2019). Overall, this makes his tasks quite interesting for studying a competing approach to simply large neural networks.

CHAPTER 11

ABSTRACT RE-IDENTIFICATON

This chapter follows the paper: Klüttermann, S., Rutinowski, J., Reining, C., Roidl, M., & Müller, E. (2022). Towards graph representation based re-identification of chipwood pallet blocks. *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1543–1550. https://doi.org/10.1109/ICMLA55696.2022.00279
aswell as the paper: Klüttermann, S., Rutinowski, J., Polachowski, F., Nguyen, A., Grimme, B., Roidl, M., & Müller, E. (2024). On the effectiveness of heterogeneous ensemble methods for re-identification [Accepted and presented, to appear]. *Proceedings of the 23rd IEEE International Conference on Machine Learning and Applications (ICMLA), Special Session 3: Machine Learning for Predictive Models in Engineering Applications (MLPMEA)*
and was partially inspired by the work of a thesis student: Grimme, B. (2023). *Entwicklung eines algorithmus für die wiedererkennung von palettenklötzen basierend auf maschinellem lernen* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/britta.pdf

11.1 Introduction and Motivation

Euro-pallets are a widely used type of load carrier, with hundreds of millions of them being in constant circulation (Deviatkin & Horttanainen, 2020). Since they are primarily moved and shipped while carrying goods, their location and identity represent domain knowledge that can be beneficial to operators that deal with the pallet in question. However, Euro-pallets, like other standardized load carriers, do not hold any inherent identification meth-

ods and are instead only identifiable by batches, defined by certain criteria, e.g., their place of assembly (Packaging, 2004). In this context, Euro-pallets can thus not be identified as individual pallets but rather classified by a given set of criteria. Therefore, the identification of load units commonly takes place through documentation (e.g., waybills) or artificial markers (e.g., QR-codes) that are attached and linked to the good itself. The motivation for this approach lies in the relative (in comparison to the goods) inexpensiveness of load carriers and the frequent change of the goods they carry. If however, it were technically and economically feasible to identify load carriers based on their inherent visual characteristics, while linking them to the good they are currently carrying, further knowledge could be gathered about the load unit and it would permit related processes to first be analyzed and subsequently to be improved upon. A first attempt to solve this problem was presented in (Rutinowski et al., 2021b), in which the authors demonstrated the feasibility of identifying the chipwood pallet blocks used in Euro-pallets based on their unique wooden patterns. However, this work relies on large-scale networks initially designed for person re-identification.

Therefore, in this chapter, we explore a novel, alternative approach using more abstract data representations. Such an abstract representation reduces the data dimensionality while preserving the information necessary for successful re-identification and increasing the model's reliability at the same time. First, we present an approach based on the assumption that the parts of an image that are unexpected provide the most information towards re-identification. To understand this assumption, consider, for example, a blue t-shirt or heart-shaped sunglasses: The more anomalous feature allows for easier recognition of a known person. For this, we are interested in characteristic features of the respective pallet block surface, such as prominent chips (i.e., visually distinguishable by size, color, etc.). These anomalies are represented as the nodes of a graph, which are connected by edges that represent the distances between the nodes. By representing this set of information as an abstract graph, the dimensionality of the data is reduced while maintaining the key features of the images.

However, because we will see that the performance of this method is not high enough to be useful in practice, we will then suggest a heterogeneous ensemble method to increase the performance significantly. This follows our experience from the chapters on anomaly detection as we create an abstract ensemble method, with every submodel considering a different view of the data.

11.2 Related Work

11.2.1 Re-Identification

We define re-identification as the attempt to retrieve a previously recorded subject of interest over a network of cameras (M. Ye et al., 2022). Thus, the predictions of a re-identification model are meant to match input data from a query dataset $\mathcal{Q}$ with a given gallery dataset $\mathcal{G}$. Prior to the matching task, training occurs using the training dataset ($\mathcal{T}$). (These datasets are distinct ($\mathcal{Q} \cap \mathcal{G} = \mathcal{Q} \cap \mathcal{T} = \mathcal{G} \cap \mathcal{T} = \emptyset$), in order to prevent information leakage and to limit overfitting.) After training, the task of re-identification can then formally be described as matching an image $x_i \in \mathcal{Q}$ of a subject i to an image $y_j \in \mathcal{G}$ of a subject j where $i = j$ (Rutinowski, Vankayalapati, et al., 2022).

Re-identification is frequently used for pedestrian surveillance (Leng et al., 2019; Ming et al., 2022; Nambiar et al., 2019; D. Wu et al., 2019; M. Ye et al., 2022), while other applications include vehicle surveillance (S. D. Khan & Ullah, 2019; X. Liu et al., 2016; Wei et al., 2018), material identification (Hermanson & Wiedenhoeft, 2011; Klar, 1995; Rutinowski et al., 2021b; Takahashi & Ishiyama, 2014; Takahashi et al., 2017) as well as animal identification (Jewell et al., 2016; B. V. Li et al., 2018; Lokare et al., 2014). Since the methods of re-identification are most commonly applied to pedestrians, different kinds of visual identification criteria have been the subject of research and can be used to distinguish between them. As such, full body re-identification based on the overall appearance of a person is a prominent one, while facial re-identification (or recognition) (Kortli et al., 2020) and gait-based pedestrian re-identification (Nambiar et al., 2019) are of interest too. For non-human, industrial entities however, only visual characteristics and domain knowledge can reasonably be taken into account.

A modern approach to re-identification, which is also considered in this chapter, is taken by so-called siamese neural networks (Koch, 2015), which learn a transformation in which similar samples are close to each other, and dissimilar ones are not. Such neural networks are often trained using a triplet loss, which minimizes intra-class distances while maximizing inter-class distances (Ming et al., 2022; L. Zheng et al., 2016).

Several evaluation metrics are commonly used to compare the resulting re-identification models. One such metric is ranked accuracy, also known as top-n accuracy, which describes whether the correct query identity is retrieved as part of the first n matches in the gallery data (Ming et al., 2022; M. Ye et al., 2022). The motivation behind the use of ranked accuracy is that it provides insight into the severity of a wrong retrieval, as there is a practical difference, for many use cases, between the second or the ninetieth match being the cor-

rect identity if the first one is not. From the ranked accuracy results, the CMC (cumulative matching characteristics) curve can be generated by summing up the accuracy of each query identity divided by the overall amount of query identities, giving a comprehensive understanding of the model's performance (Ming et al., 2022; M. Ye et al., 2022).

For one task considered in this chapter, Rutinowski et al. (2021b) proposed a Euro-pallet re-identification pipeline, exploiting the unique chipwood pattern of pallet blocks. In this contribution, pallet blocks are detected through YOLO-based object detection. The resulting images are passed through a ResNet50 implementation of PCB_P4 (Part-based Convolutional Baseline) (Y. Sun et al., 2018), using cross-entropy loss, to be stored as a feature vector for the subsequent task of re-identification under a closed-set assumption. The results of the contribution suggest that the surface structure of chipwood can be reliably exploited for the purpose of re-identification, using the dataset *pallet-block-502*, that was created for this very purpose.

Still, their approach is based on the usage of very large neural network approaches, and thus, it is slow when training or predicting, limiting its usefulness. Instead, we aim to create a likely faster approach, using our focus on abstraction methodologies.

11.2.2 Graph Representation Learning

As graphs are designed to store relations between subjects, nodes and edges are often used to represent and analyze non-trivial subjects. Examples range from medicine over particle physics to linguistics (Agosta et al., 2013; Feynman, 1949; Piperski, 2014). Because of this diverse range of uses, the method of graph machine learning has gained popularity in recent years (J. Zhou et al., 2018).

Even for images, a graph representation can outperform ordinary convolutions (Avelar et al., 2020), as graphs allow for the modeling of structures instead of pixels and store these structures in only a few features. This can, for instance, be realized through superpixels (Achanta et al., 2012), which are regions consisting of multiple similar pixels that are combined into one node. In contrast to the superpixel approach, this work does not focus on representing all individual pixels and instead focuses on determining regions containing relevant information. In person re-identification, the use of graph neural networks is novel but gaining popularity. D. Chen et al. (2018) and Shen et al. (2018) employ graphs to improve the triplet loss used in a siamese network. Displaying a higher degree of similarity to the herein presented approach, Khaldi et al. (2022) use graphs to represent a person by their joints. However, while doing so, the authors focus on extracting information from

related structures in time. Given these circumstances and the specific use case, their representation graph construction is heavily optimized for the task of person re-identification.

11.2.3 Ensembles for Re-identification

Using ensemble models is a common practice in machine learning, allowing the combination of different predictors to achieve a better performance (Zimek et al., 2014). There are many methods, ranging from the averaging of multiple models (bagging) (Breiman, 1996) to training new models on the prediction of the ensemble submodels (stacking) (Sandim, 2017). Additionally, ensembles can have further uses, like explainability (See Chapter 7). However, there are surprisingly few applications to re-identification, which requires us to develop ensemble methods further. We want to summarize the existing approaches in this section.

A common approach to re-identification focuses on extracting general features representing aspects of a sample and using their resemblance as an indicator of similarity. This invites the use of ensembles to combine different features. A previous publication (X. Liu et al., 2015) employs ensembles by combining different hand-crafted color descriptors, which allows the extraction of information from more than one feature. Recent approaches (Serbetci & Akgul, 2020; J. Wang et al., 2019; Y. Yang et al., 2018) apply more advanced neural networks to find features using pre-trained models or to extract them directly.

The drawback of feature extraction is that many datasets contain a large amount of superfluous features, which tend to be included in well-studied, commonly extracted features. Additionally, there has not been enough research conducted yet to find specialized features for pallet re-identification and a solution emerging from such research would only be applicable to one use case or dataset, thus not leading to a general solution. Siamese neural networks(Koch, 2015), however, are able to effectively extract useful features automatically, providing a more generalizable and adaptable solution. Other approaches use ensembles for metric learning (Paisitkriangkrai et al., 2015) or to better handle multiple data modalities (M. Ye et al., 2020).

However, to the best of our knowledge, ensembles have hardly been studied for siamese neural networks. Worth mentioning is (S. Xu et al., 2018), in which ensembles are used to enhance contrastive learning in the context of natural language processing. While this ensemble method shares similarities with our approach, the application is fundamentally different, which limits the transferability of results.

11.3 Methodology

The task of re-identification is often solved by learning a transformation f of a dataset such that the distances between the representations of the same entity are minimized and the distances between different ones are maximized. This goal can be formalized as follows (with x^i being a sample following concept i):

$$\|f(x_a^i) - f(x_b^j)\|_2 = \begin{cases} \text{small if } i = j \\ \text{large if } i \neq j \end{cases} \quad \forall x, y \tag{11.1}$$

This task can be achieved by minimizing a suitable loss function, e.g., triplet loss (Hoffer & Ailon, 2015):

$$L_{triplet} = \max(0, \|f(x_a^i) - f(x_b^i)\|_2 - \|f(x_a^i) - f(x_c^j)\|_2 + \alpha) \;,\; i \neq j \tag{11.2}$$

where x_a^i, x_b^i and x_c^j represent three samples of two different subjects i and j and α is a hyperparameter. This loss value is averaged over a large number of different triplets.

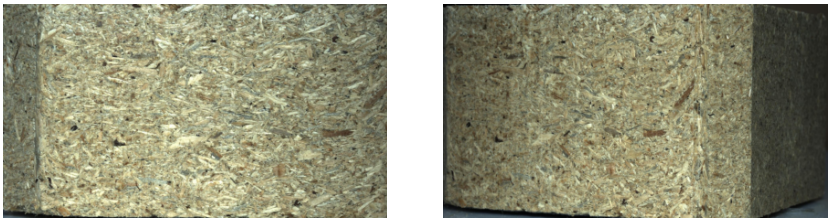
11.3.1 Data Preparation

In this chapter, we use two datasets inspired by the logistics application, examples of which are shown in Figure 11.1.

The first one (Rutinowski et al., 2021a) contains images of pallet blocks and 5,020 RGB images of pristine, unbranded EPAL pallet blocks of which only half 2,510 are used for the herein presented experiments, since the remaining pallet blocks are taken under less visible lighting conditions. This dataset contains images taken from five different perspectives: central (C), left-hand and right-hand side rotation (RL, RR), and left-hand and right-hand side shift (L, R). We show an example of these rotations in Figure 11.2. The second dataset (Rutinowski, Endendyk, et al., 2022) contains 5,088 images of galvanized metal plates. We again use half of the dataset (2,544), which was taken with photographic lighting conditions.

11.3.2 Cross-validation

To provide more meaningful results and to estimate prediction uncertainties, we employ cross-validation. Cross-validation is less common in re-identification research, likely because training can be expensive.


(a)

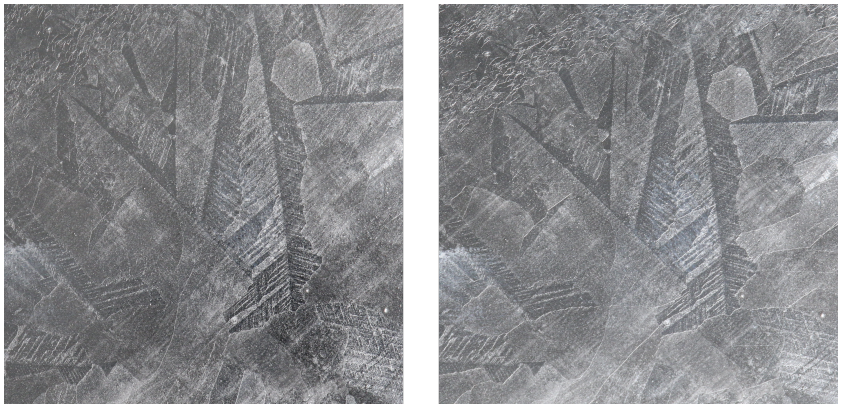

(b)

Figure 11.1: Excerpts of the used datasets. a) pallet block dataset (Rutinowski et al., 2021a), b) galvanized metal dataset (Rutinowski, Endendyk, et al., 2022)

We split our dataset into five different groups on the pallet block dataset and six different groups on the metal dataset. One of these folds is chosen as a query ($\mathscr{Q}$) and gallery ($\mathscr{G}$) set, in which for each pallet block, one image is selected for the query ($\mathscr{Q}$) set, while the rest become part of the gallery ($\mathscr{G}$) set. The remaining folds are used as a training ($\mathscr{T}$) set.

11.3.3 Graph-based Approach

We first study an approach representing a pallet block through a graph. Our methodology, visualized in Figure 11.3, can be subdivided into three key components:

1. The information stored in the dataset images is transferred into a graph. This allows for the removal of a large portion of features from the dataset while allowing us to focus on a more interesting high-level representation. Given this background, we assume that the parts of the pallet blocks that contain the largest amount of relevant information are the most visually striking pieces of chipwood. The anomalies in these image parts likely contain most of the information needed for re-identification.

2. We use highly reduced (in terms of dimensionality) graph representations to train a siamese neural network to further reduce the data dimensionality. During this pro-

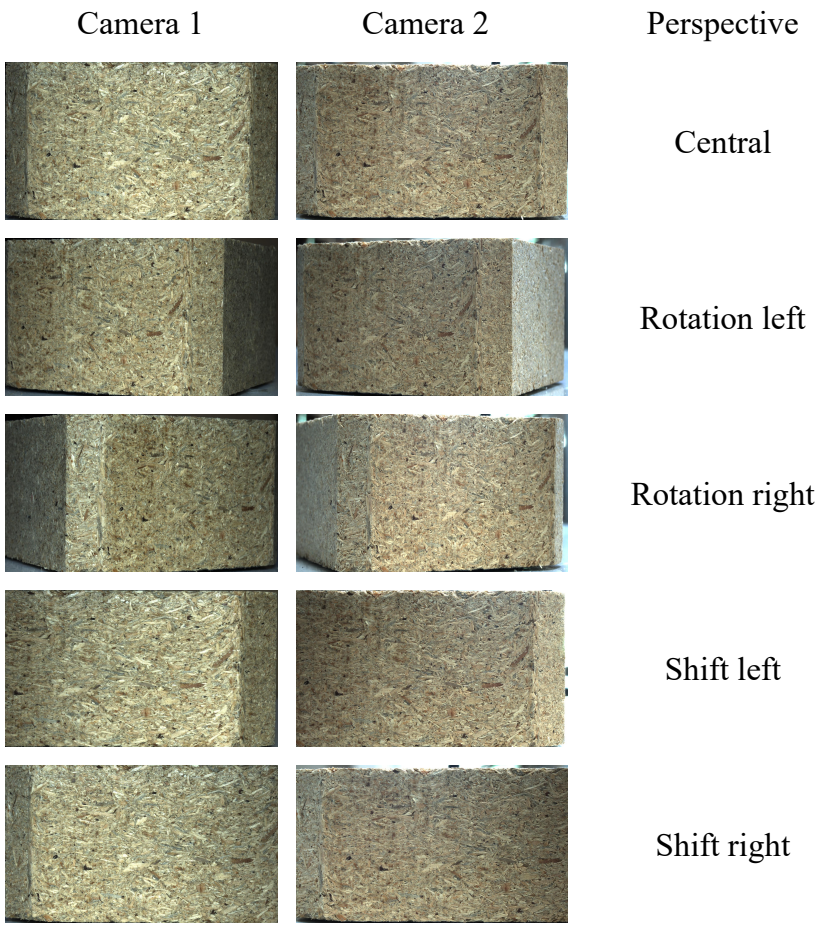


Figure 11.2: Example of the ten images given for one pallet block (ID 301).

cess, we map images of the same pallet block to similar positions in the output space, while mapping different blocks to distinct embedding space positions.

3. We use an algorithm to match new images in their graph representation, i.e., the siamese network's output, to the a priori known image that is the closest in the embedding space (i.e., the most similar one). For this purpose, a k-nearest neighbor algorithm with $k = 1$ is used.

One further challenge that arises with the first dataset is that the chamfered edges of

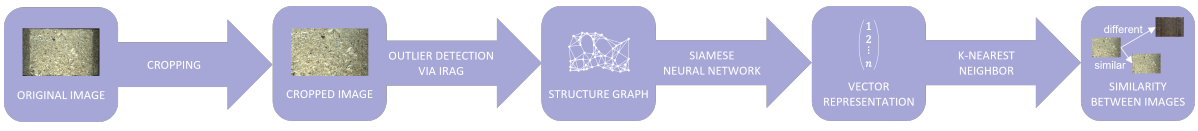


Figure 11.3: Workflow of the graph based approach.

the pallet block (see Figure 11.2) are distinct from the rest of the image, are not always in the camera frame and should thus generally not be considered to contain anomaly nodes. In order to remedy this wrongful consideration as anomalous, the images are cropped so that the chamfered edges of the pallet blocks are not visible. We use a simple heuristic, preferably removing too much of the image rather than accidentally leaving a chamfered edge in the resulting dataset. This means that information is lost during pre-processing, which might further complicate the task at hand. At the same time, it might also make the resulting algorithm more robust, as pixels on the side of the center area are not always clearly visible (consider the right-hand side shift in Figure 11.2). We subsequently rescale our images to a size of 500 × 300 px, roughly preserving the original average image aspect ratio of 1.7.

11.3.4 Graph Representation Procedure

Since we consider anomalies in the dataset images to be points of interest, it is sensible to define these anomalies as the nodes of the representation graph. This requires an efficiently working algorithm to find anomalies in a given image. While we likely could use one of the anomaly detection methods proposed in the earlier chapters of this thesis, the scale of the anomaly detection task makes this non-trivial. In addition to thousands of images, we have to check every small subpart of an image for its anomaly score. This results in very many prediction tasks, eroding every speed benefit we can gain from the reduced description size. Alternatively, we could use the Shapley scores from Chapter 7 to understand which part of an image is the most anomalous. But this would still incur very large computations, as our images are even larger than the version of celebA used there (which required $27h$ when highly parallelized). So instead, we use a simple heuristic, exploiting the simple structure of our dataset. Still, it would be interesting to see if we could generalize our findings here to less homogenous datasets when using such an anomaly detection task. This heuristic considers a subregion (16 × 16 px areas) S to be more anomalous the higher $A(S)$ is (with the average $mean(S)$ and the standard deviation $std(S)$ aswell as a small $\epsilon = 10^{-6}$). $A(S)$ is defined as:

$$A(S) = \frac{std(S)}{mean(S) + \epsilon} \tag{11.3}$$

We iterate over each 16 × 16 px area (with a 50% overlap between successive areas in each dimension) of the image and calculate such an anomaly score $A(S)$ for it.

To build a graph from a set of anomalous nodes, a challenge to be taken into consideration is the wrongful adjacency of nodes that display such extreme proximity that they ought to be merged into a single node. In order to minimize this occurrence, we introduce a modification to the algorithm, in which all subregions are looped through, in descending order of $A(S)$. If the current S is close to another node ($mse(pos[curr], node) < \tau$), the average of the position of both nodes is calculated, merging both nodes into a single one. If S fails this threshold instead, a new node is created. This process is repeated until a total of 50 nodes are generated, which we consider to provide a sufficiently complex graph, representing the given images accurately. This node-finding algorithm is described in Algorithm 2. In each node, information about a given anomaly and its surroundings is saved. A list of all parameters taken into consideration for each node can be found in Table 11.1.

Table 11.1: Features representing one node of the graph.

Features		
Location [px]	Color [RGB]	Dimensionality
X-axis location	Average subregion color	Number of subregions S_i
Y-axis location	Color standard deviation	Subregion size

Algorithm 2: IRAG

Input: S_j, pos_j (subregions and their positions), $nodec$ (number of nodes to find), τ (minimum distance between two nodes)

Output: $nodes$ (locations of the nodes found)

$n \leftarrow 0$;
$nodes_i \leftarrow (0,0)\ \forall i < nodec$;
$regions_i \leftarrow 0\ \forall i < nodec$;
$a_i \leftarrow A(S_i)\ \forall S_i$;
$dex \leftarrow \text{reverse}(\text{argsort}(a_i))$;
$k \leftarrow 0$;
while $n < nodec$ **do**
    $curr \leftarrow dex[k]$;
    $dist \leftarrow \text{mse}(pos_{curr} - nodes[j])\ \forall j < n$;
    if $\min(dist) < \tau$ **then**
        $closest \leftarrow \arg\min(dist)$;
        $nodes_{closest} \leftarrow \frac{nodes_{closest} \cdot regions_{closest} + pos_{curr}}{regions_{closest}+1}$;
        $regions_{closest} \leftarrow regions_{closest} + 1$;
    else
        $nodes_n \leftarrow pos_{curr}$;
        $regions_n \leftarrow 1$;
        $n \leftarrow n+1$;
    end
    $k \leftarrow k+1$;
end
return $nodes$;

Subsequently, graph connections are generated using a top-k algorithm on the location of the anomaly, connecting each node with the five other nodes with the highest proximity. The aim of this method is to capture a minimalist version of the pallet block's surface structure as a graph representation.

We show the features stored in each node in Table 11.1 and illustrate the node generation process in Figure 11.4.

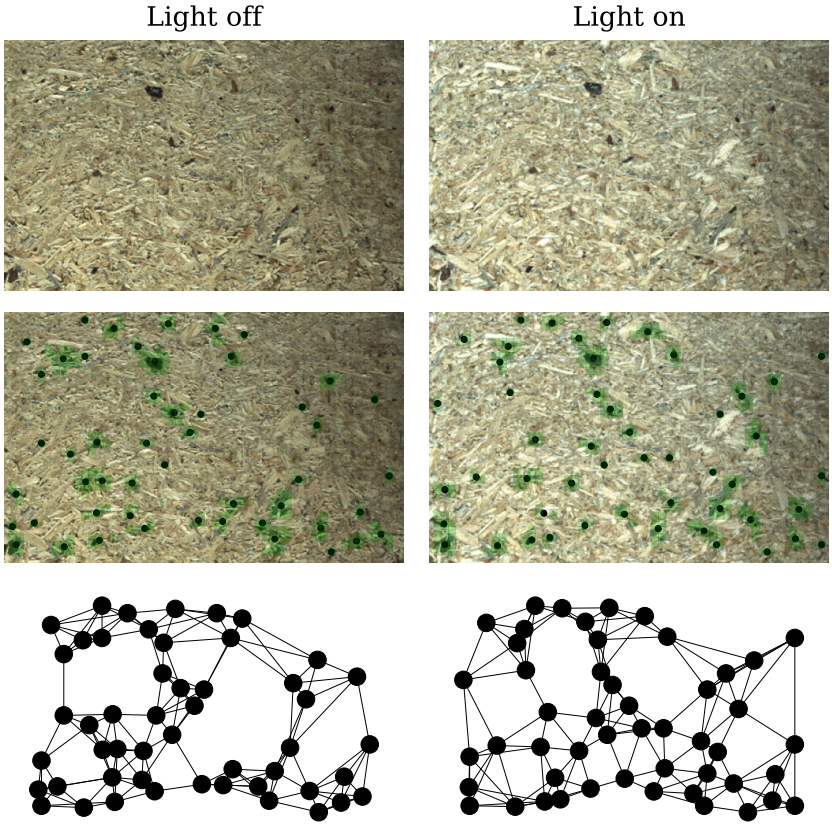


Figure 11.4: Transforming an image of a pallet block (top row; ID 120) into a graph (bottom row) by means of anomaly detection (middle row)

11.3.5 Siamese Graph Neural Network Training

We train the herein proposed Siamese graph neural network re-identification model using triplet loss (Hoffer & Ailon, 2015) as a loss function. This means that each input consists of three samples: a comparison sample X_i, a sample of the same pallet block T_i, and one of a different pallet block F_i. Our model $f(X)$ is trained so that the loss function L_T is minimized:

$$L_T = \sum_i Relu(diss(f(X_i), f(T_i)) - diss(f(X_i), f(F_i)) + \alpha) \tag{11.4}$$

Here, $diss$ represents the mean squared difference, and the ReLU function is defined as $Relu(x) = \frac{|x|+x}{2}$.

Intuitively, this approach aims at minimizing the embedding space distances between the same pallet block and maximizing the distance between different pallet blocks. We train the model described in Table 11.2 in batches of 100 images for up to 100 epochs with a patience of 5, using the Adam optimizer and a learning rate of 0.0001. We define the shape of each graph layer as $nodes \times features + nodes \times nodes$ (node features and adjacency matrix).

These graph convolutions are provided by the Python library spektral (Grattarola & Alippi, 2020): Each convolution combines the node values with the values of the nodes that they are connected to, using learnable weights. Subsequently, we combine the values of each node using a global pooling operation.

Table 11.2: Architecture of the Siamese graph neural network.

Layer type	Output shape	Activation function
Input	50 x 10 + 50 x 50	-
Graph Convolution	50 x 31 + 50 x 50	ReLU
Graph Convolution	50 x 31 + 50 x 50	ReLU
Global Mean Pooling	31	-
Fully Connected	250	ReLU
Fully Connected	250	ReLU
Fully Connected	250	ReLU
Fully Connected	50	Linear

11.3.6 Re-identification Methods

In order to improve the previously introduced graph method, we extend it into an ensemble. This requires further submodels.We chose to study heterogeneous submodels here for now, as there might be some information that is not represented by our graph-based approach.

Here, we suggest five siamese models for each dataset, all based on different approaches:

Image-based re-identification: The most direct approach for training a siamese neural network on image-based data is to use convolutional neural networks. The benefit of this approach is its ability to extract information available in the entire image. However, this also represents the most time-consuming approach, and is the most susceptible to overfitting. We choose here a simpler, non pretrained approach, when compared non-ensemble approaches to limit the required computational effort.

Since the original images in the dataset are high-resolution images, they are resized to a dimension of 400 × 230 px. The resolution of the original images varies, since they have been automatically cropped based on YOLO's bounding boxes, as is described in (Rutinowski et al., 2021b). Their average resolution, however, is 1814 × 1096 px.

The transformation in our approach consists of six convolutional layers, each increasing the number of features by 1.5 with a kernel size of 3. After every other convolution, we employ a max-pooling operator with a kernel size of 2 and add 3 dense layers to learn a final representation with 100 dimensions. We include an ablation study on this representation size in Appendix B.2. We employ ReLU as an activation function, a learning rate of 0.001 and a batch size of 256.

Graph-based re-identification: This previously described approach is specialized to the pallet block dataset, and our attempts to generalize it to the metal dataset were so far unsuccessful, as the galvanized metal plates do not contain informative anomalous points. Thus, we only use it on the first dataset.

Linear quantile re-identification: To compensate for the lack of applicability of the graph-based approach to the metal dataset, we developed a different model, exploiting the perspective-wise shearing in the galvanized metal dataset. For this purpose, we split the images into pixel columns and use the 20%, 50% and 80% quantile of the pixel values in this column as an input for a siamese neural network.

Brightness, average color and color variance based re-identification: The remaining three approaches again split a given image into sub-images of size 16 × 16 px each. We iterate over all sub-images of an image (with 50% overlap) and calculate 1 – 3 values for each, resulting in a representation with 768/2304 dimensions. For the *brightness*-based approach, we calculate the mean of each sub-image, while for the *average color*-based approach, we calculate a mean for each color and sub-image. Finally, for the *color variance*-based approach, we use the standard deviation of each color throughout the sub-image. The values for each sub-image are concatenated into a vector and used to train a siamese neural network. Each network uses three dense layers with 100 nodes each to generate a 50-dimensional representation. While these approaches ignore much of the available information given in an image, they are also straightforward and time-efficient, reducing the required training time by multiple orders of magnitude.

11.3.7 Ensembles

In this chapter, we consider ensembles that combine multiple siamese neural network representation learning approaches (e.g., f and g) and combine them using an ensemble cre-

ation function T into an ensemble e.

$$e(x) = T(f(x) \oplus g(x)) \tag{11.5}$$

Here, we consider five different transformations T that create such ensembles. While ensembles are used in many cases (see 11.2), they are only rarely used for siamese network-based re-identification and thus we propose our own transformation functions.

The simplest ensemble transformation presented in this work, the *Concatenation* transformation, normalizes each representation component using the mean and the standard deviation (s_{x_j}) for all considered samples and concatenates them. It can be seen as an ensemble method following the concept of bagging (Breiman, 1996).

$$T_{Concat}(x_0^i, ..., x_n^i) = (z_0^i, ..., z_n^i)\ ;\ z_j^i = \frac{x_j^i - \bar{x_j}}{s_{x_j}} \tag{11.6}$$

Here, x_j^i denotes the j-th component of the representation of sample x^i. Meanwhile, the *Neural Network Triplet* transformation is a stacking method (Mienye & Sun, 2022), which uses a neural network to represent $T_{NNtriplet}$ and to train it by minimizing 11.2 on the training set $\mathcal{T}$. The *Weighted Triplet* instead simply uses weighing factors:

$$T_{Wtriplet}(f(x) \oplus g(x)) = \alpha_f f(x) \oplus \alpha_g g(x) \tag{11.7}$$

These are found by minimizing Equation 11.2, using gradient descent. Subsequently, the *Weighted Accuracy* transformation tries to maximize the probability that the closest sample $y^j \in \mathcal{G}$ to a sample of the test set $x^i \in \mathcal{Q}$ depicts the same subject $i = j$. Since this probability is not continuous, gradient descent cannot be used for this task. Instead, the probability is optimized using the *flaml* library (C. Wang et al., 2021), which is designed to optimize non-continuous hyperparameter optimization problems, and thus does not rely on gradient descent. Additionally, a fifth ensemble, called the *Majority Vote* is created using a different approach to ensembling. For this ensemble, multiple submodels do not calculate the distance to each gallery sample $\in \mathcal{G}$, but simply order the gallery samples based on their distance to the query sample. We choose the most common index of this order through all submodels as an indicator of similarity between samples.

11.4 Experiments

Here, we consider first the performance of the graph-based method and then later extend it into ensembles.

11.4.1 Graph Model Performance

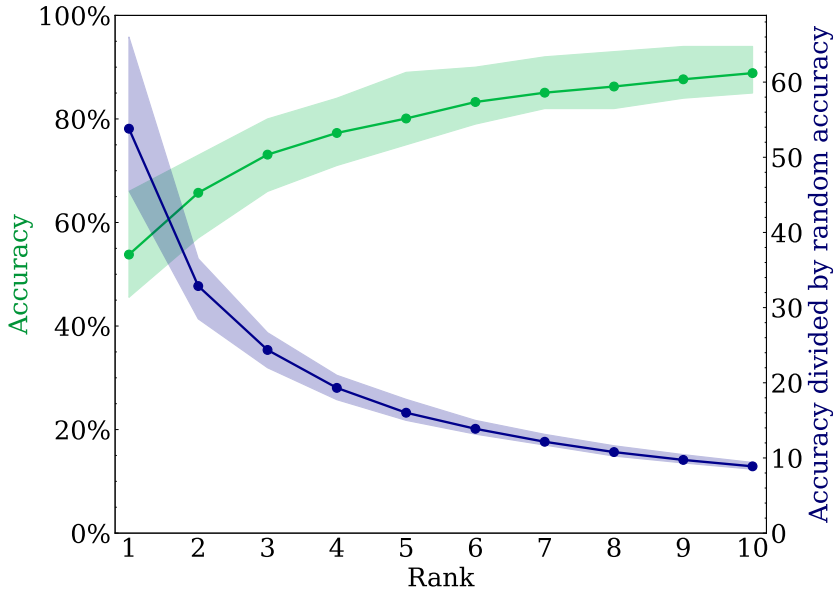


Figure 11.5: Ranked accuracy of the re-identification task plotted as a CMC curve.

We show the performance of our graph-based approach using a CMC (cumulative matching characteristics) curve. Here, we plot the rank-k accuracy against k in Figure 11.5. The resulting rank-1 accuracy of 54% is not yet entirely competitive against comparable methods, providing, e.g., > 72% rank-1 accuracy in (Rutinowski et al., 2021b). Nevertheless, with the results of these experiments being preliminary, we provide novelty through the methodology itself and demonstrate the feasibility of a graph representation-based abstract re-identification approach, which will be further improved upon in the following sections and chapters. It should also be taken into account that on no occasion, the accuracy provided by our model results in random guessing, as the blue curve in Figure 11.5, even with uncertainty, is never close to the null hypothesis of 1.

11.4.2 Graph Model Efficiency

Compared to the 45,000 features of the original images, we use a highly abstract representation with 900 times fewer features in our graph (500 features). This translates to a much more memory and processing time-efficient computation. Due to this, we achieve a training duration of < 10 min on a laptop compared to multiple hours of training on a cluster when using images. It is also much cheaper to store these graph representations. After using a common compression, our training set has a size of 581 MB in the original image

representation, while the graph representation only requires 3.7 MB. This is a factor of 157, which is less than the maybe expected 900 times reduction, as the images are more susceptible to compression.

11.4.3 Graph Model Robustness

Generally, abstraction is often measured as the ability to generalize better to slightly different tasks (See Section 2.3). As we are using here data that contains different rotations, we also want to measure how well this is fulfilled for our graph-based model. Thus we compare the re-identification results on the different kinds of images in the dataset (i.e., the different camera angles and perspectives used). A non-robust model might be better at re-identifying images of a certain type of perspective over another. We show the results of testing this hypothesis in Figure 11.6 by plotting the rank-k accuracy for each re-identification task that involves a certain perspective type of image (either as a gallery or a query image). The differences between perspective performances are limited, showing some generalization effects. However, some are still somewhat significant, especially for, for example, the comparison of RL to C, where we see no overlap of the uncertainties. We see this as a further sign that the graph model is not perfect, and we will revisit this experiment after extending our model into an ensemble (Section 11.4.7).

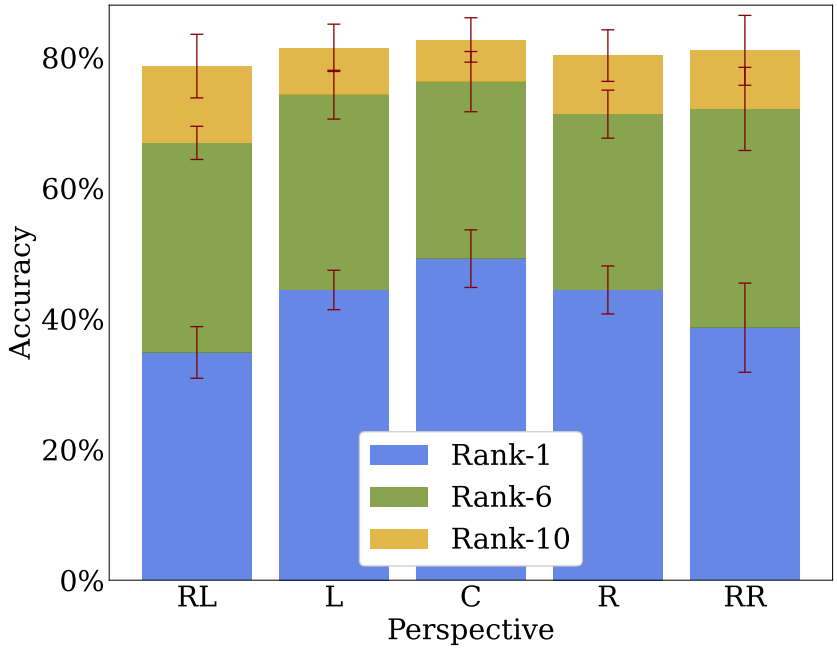


Figure 11.6: Rank-1, rank-6 and rank-10 accuracy of the graph model with their corresponding uncertainties, ordered by perspective.

11.4.4 Submodel Performance

To extend this into an ensemble, we first evaluate the individual submodels we propose and plot the resulting CMC curves in 11.7.

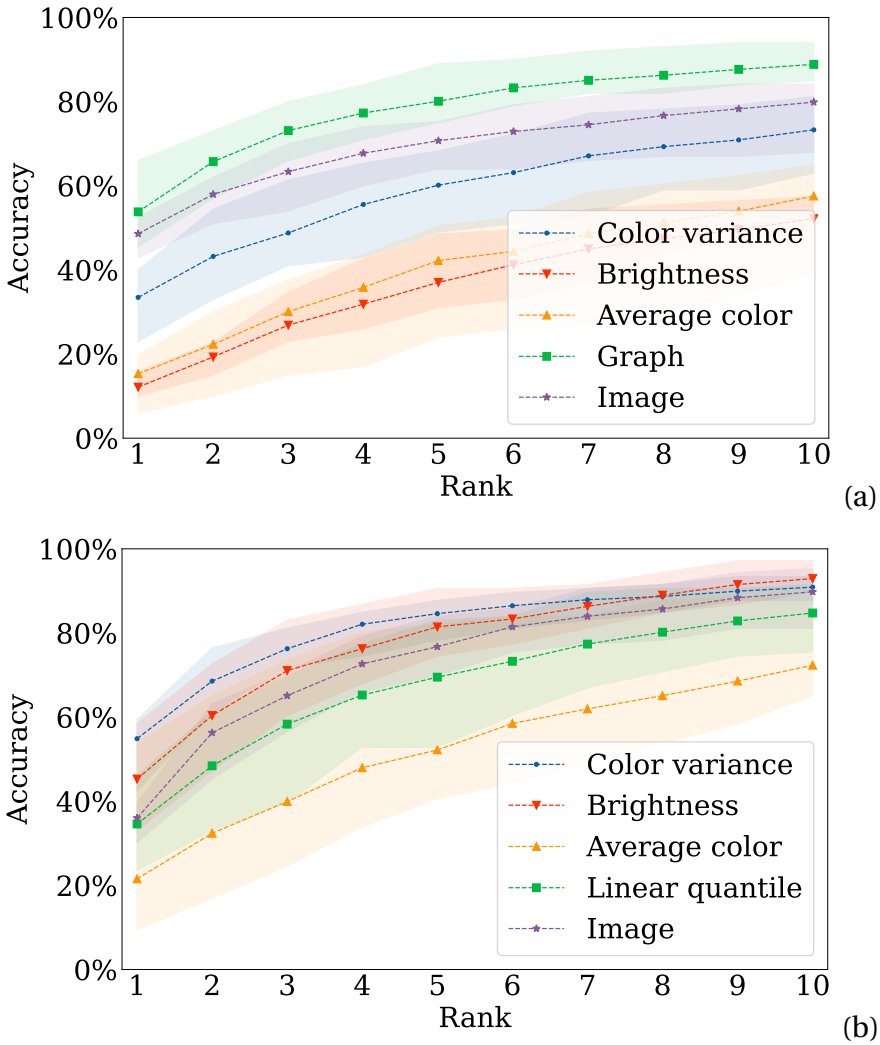


Figure 11.7: Rank-k accuracy for the ensemble submodels. The highlighted areas represent the highest and lowest scores per individual fold. a) pallet block dataset, b) metal dataset

The two methods averaging color values (*brightness* and *average color*) provide subpar results on the pallet block dataset, with their overall performance reaching a Rank-1 accuracy of $< 20\%$. An entirely different effect can be observed for the metal dataset, with the simpler *brightness*-based method outperforming a method with image-wide information. This might be a bit confusing, as this submodel simply has access to significantly less information and still outperforms the image-based model. We will further investigate the reason behind this effect in Chapter 12 of this thesis.

In both cases, using the *color variance*-based model provides a higher accuracy. We assume that this is due to high variances often representing unusual structures that can be used for re-identification, while in an average-based method these structures are averaged out. Even though the performance of these simple methods are not always competitive, they are an order of magnitude faster to train than other methods (i.e., $\sim 8s$ for the color variance model compared to about $20min$ for the Inception model) and thus can be valuable ensemble submodels.

Notice that on both datasets, the best performance was not achieved using the highest number of available features. This implies the notion that performance can further be improved using an abstract ensemble method, as a singular holistic data description already does not seem to be optimal.

11.4.5 Ensemble Performance

In this section, the performance of the ensemble models is studied. 11.8 show the performance of the five different ensemble models discussed in 11.3.7.

The ensemble models provide much higher performances than the individual models. Given the results in 11.8, the ensembling approach yielding the highest accuracy on the pallet block dataset seems to be the concatenation, which allows us to improve the Rank-1 accuracy from 54% to over 70%, compared to the graph-based submodel. On the metal dataset, the concatenation also performs well, but here, learning weights seems valuable, increasing the Rank-1 accuracy slightly more, from 55% to almost 78%, compared to the best submodel. It is quite counter-intuitive that the simplest (and unsupervised) approach to ensembling also yields the highest overall performance.

We studied three ways of weighing individual submodels and expected the weighing approaches to improve results, given that Figure 11.7 showed significant differences of the submodels' performances. We would also have expected similarly functioning approaches (like our *brightness* and *average color*-based approaches) to potentially cancel one another out. However, while weighing seems to increase the resulting performance in some cases, it seems not always to be an effective solution. One reason for this phenomenon might be the weighing approaches' susceptibility to overfitting.

The lowest (Rank-1) accuracy is obtained by the neural network approach, for which we use another neural network trained on the output of each siamese neural network to minimize the triplet loss (11.2) and generate a new common representation. This matches our expectation since the neural network also contains the highest number of parameters and thus has the highest potential for overfitting. Slightly better results are provided when a

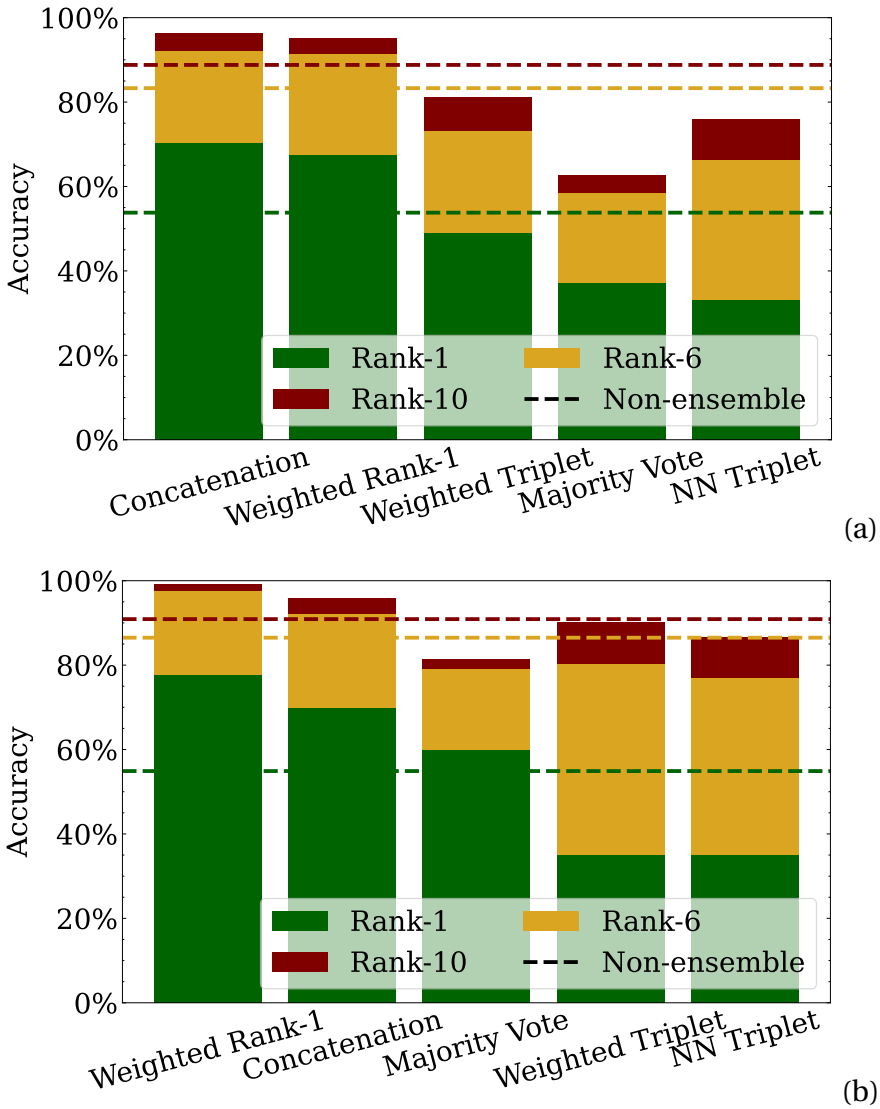


Figure 11.8: Rank-1, Rank-6, and Rank-10 accuracy for the five different ensemble approaches compared to our most effective non-ensemble model (the graph-based model and the color variance approach). a) pallet block dataset, b) metal dataset

weighting factor is added and optimized to the output of each submodel (*Weighted Triplet*). Still, the performance is inferior to that of the best-performing submodel. We expect this to be a result of a local minimum of the optimization task because when changing the function to be optimized from a triplet loss to a Rank-1 accuracy on the training set (*Weighted Accuracy*) the performance can be improved, up to a level comparable to the performance of the *Concatenation* approach. In any case, this implies that there remains untapped potential for the application of more sophisticated ensemble methods.

We tried another approach, in which the median prediction of each submodel is chosen

(*Majority Vote*). Still, this method does not perform well either.

Further ablation studies are found in Appendix B.

11.4.6 Comparison to other Re-identification approaches

After showing that an abstract ensemble approach can work for re-identification ensembles, in this section, we want to compare such an approach against the existing methodology (see 11.3). We choose to compare our models to a method similar to one used for Person Re-identification (Islam, 2023; Y. Sun et al., 2018). The approach here is to use a pretrained model, add three Dense layers for the output to match, and continue training the model. The main difference to common person re-identification models, is that we employ the InceptionV3 object detection model (Szegedy et al., 2016), instead of a model trained to re-identify pedestrians. We chose this model over, e.g., other state-of-the-art models like ResNet50(K. He et al., 2016) and MobileNetV2(Sandler et al., 2018) as it seems to provide significantly higher performance in our experiments. Still, this approach is not able to outperform our ensemble methods while requiring significantly more time to train. We also include the uncertainties over the cross-validation folds.

Table 11.3: Comparison of our method with alternative approaches on the pallet block and metal datasets.

Dataset	Model	Rank-1	Rank-10	Runtime [s]
Pallet	Ensemble	**0.703** $\pm$ 0.079	**0.964** $\pm$ 0.020	287
	Graph (Klüttermann et al., 2022)	0.526 $\pm$ 0.052	0.904 $\pm$ 0.045	50
	Image (Rutinowski et al., 2021b)	0.486 $\pm$ 0.032	0.799 $\pm$ 0.060	213
	Inception (Szegedy et al., 2016)	0.518 $\pm$ 0.05	0.918 $\pm$ 0.014	1,116
Metal	Ensemble	**0.777** $\pm$ 0.054	**0.992** $\pm$ 0.007	257
	Color Variance	0.549 $\pm$ 0.057	0.909 $\pm$ 0.022	8
	Image (Rutinowski et al., 2021b)	0.360 $\pm$ 0.033	0.898 $\pm$ 0.047	224
	Inception (Szegedy et al., 2016)	0.681 $\pm$ 0.059	0.951 $\pm$ 0.013	1,283

Additionally, it seems clear that even when excluding the pretaining time for the Inception model, it is still significantly slower than our abstract ensemble approach. Furthermore, we can accelerate our approach even further, at the cost of a minor reduction in accuracy. This is studied in Appendix B.1.

11.4.7 Ensemble Robustness

Similar to our analysis in Section 11.4.3, we can also consider the robustness of our ensemble in regard to different perspectives. We show this in Figure 11.9 for the concatenation ensemble. While we still see some differences between different perspectives, these are much smaller than when considering only the graph (Figure 11.6) and are here all smaller than the corresponding uncertainties, showing the improvements in robustness and generalization of our abstract ensemble approach.

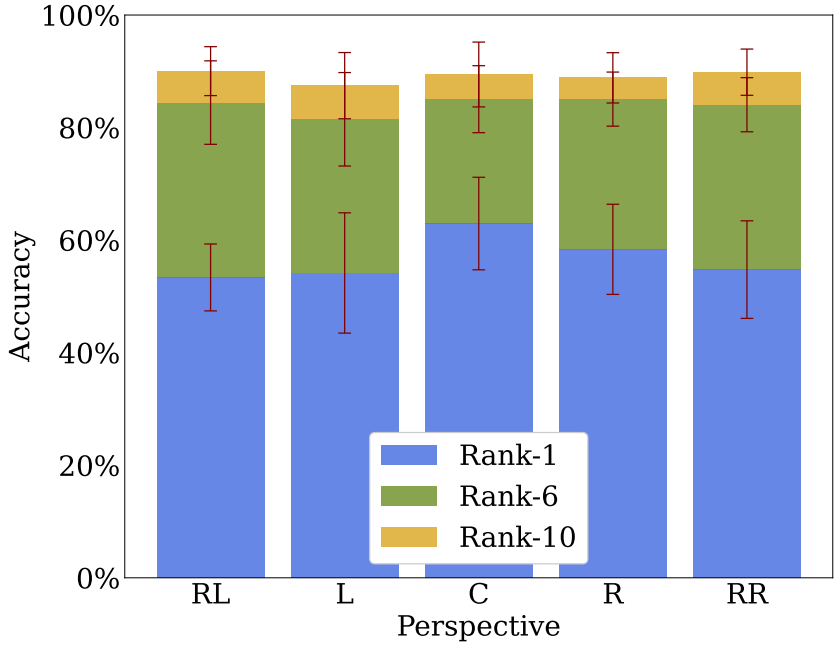


Figure 11.9: Rank-1, rank-6 and rank-10 accuracy of the concatenation ensemble with their corresponding uncertainties, ordered by perspective.

11.4.8 Reliability

Finally, we want to study the effect that using ensembles has on increasing the reproducibility and reliability of our resulting model. For this, we present the relative uncertainty of the Rank-10 accuracy of each of our individual and ensemble approaches in Figure 11.10.

We choose to present Rank-10 instead of Rank-1 accuracy, as these uncertainties are naturally quite noisy, which is reduced by the higher rank. Both ensembles that outperform each individual model in terms of accuracy also have the lowest uncertainty. They reach an uncertainty that is almost an order of magnitude smaller than the *average color* approach and clearly lower than all individual models. This increased reliability can be a crucial benefit to the implementation of our method in industrial settings.

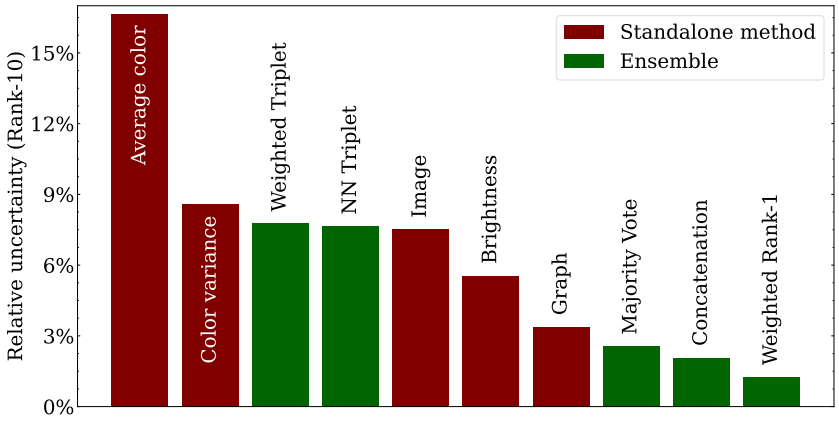


Figure 11.10: Relative uncertainty (accuracy uncertainty divided by accuracy) of the Rank-10 accuracy for different models.

11.5 Conclusion

In this chapter, we provided a novel graph representation-based approach aswell as the first study of heterogeneous siamese neural network ensembles for the re-identification of chipwood pallet blocks. While the accuracy of the graph-based model is only proof of concept, it is still very efficient and shows some potential to be more easily generalized to slightly modified datasets. By extending it into an ensemble, we reach more competitive Rank-1 re-identification accuracies of 70% and 77% on the pallet block and metal datasets. Additionally, our abstract ensemble approach is significantly faster than the usual large neural network approaches.

Similar to our findings in Chapter 3, one of our employed ensembling methods, the *Concatenation* method, a fairly simple, unsupervised approach, performs better than most of the more sophisticated ones evaluated in this chapter.

CHAPTER 12

THE PROBLEM OF CORRELATED REPRESENTATIONS IN CONTRASTIVE LEARNING

This chapter follows: Klüttermann, S., Rutinowski, J., & Müller, E. (2024). The phenomenon of correlated representations in contrastive learning. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN60899.2024.10649913

12.1 Introduction

Concluding our studies on re-identification, we study an unexpected phenomenon that we have noticed while writing the last Chapter. And as it turns out, this phenomenon leads to a better understanding of abstract ensembles for re-identification and allows us to create even better homogenous ensembles in an approach similar to DEAN for re-identification.

In contrast with the findings of other publications (Ciga et al., 2022; Koziarski & Cyganek, 2018; Sabottke & Spieler, 2020; Thambawita et al., 2021) and what could by now be regarded as common knowledge, we found that using data containing less information, i.e., lower resolution images or reduced description sizes as shown as submodels in Chapter 11, yielded a higher re-identification performance. This is a novel phenomenon and quite counter-intuitive.

A minimal example that encapsulates this behavior is shown in Figure 12.1, in which a drastic increase in accuracy based on a reduction in resolution can be observed. This phenomenon can be considered paradoxical, because neural networks are understood to be

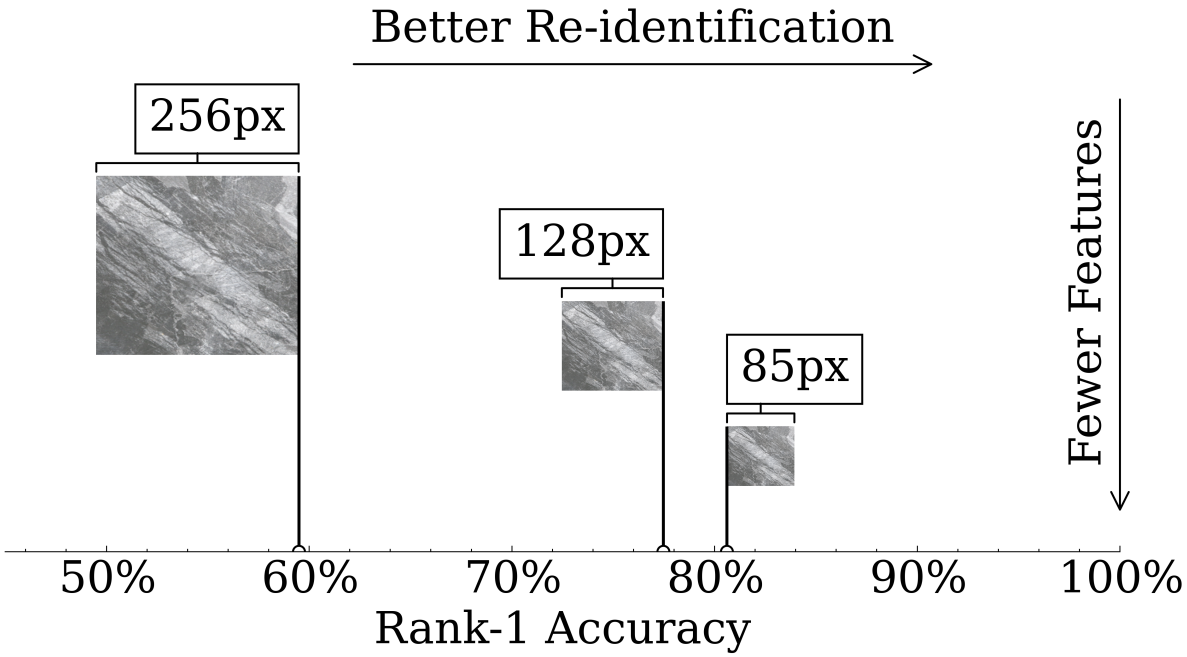


Figure 12.1: Re-identification accuracy in relation to image resolution. Counter-intuitively, a decrease in resolution (the original resolution of 256px being divided by 2 (128px) and 3 (85px)) leads to an increase in accuracy. The images seen in this figure represent galvanized metal surfaces and are taken from (Rutinowski, Endendyk, et al., 2022).

able to approximate functions to whatever information they are fed, and thus minimizing a triplet loss function should produce as good a representation as possible (Hermans et al., 2017; Schroff et al., 2015). And since we can learn additional representations on a higher resolution, these should work at least as well.

We believe this phenomenon to be linked to a novel problem in contrastive learning, that manifests as a high degree of correlation between extracted features. Therefore, in this chapter, we will study the effects that various changes to a common re-identification model have on this phenomenon. We will study this on three datasets and show how abstract, DEAN-like ensembles can mitigate the effect of this phenomenon. We will study and characterize this phenomenon by linking it to what we call the occurrence of *correlated representations*, which we believe to be its precursor. By mitigating the occurrence of correlated representations, that are neither dependent on a specific dataset nor a specific model architecture, we believe to be able to provide the research community with a way of significantly improving the performance of most re-identification approaches and further understand abstract ensembles.

12.2 Related Phenomena

Recently, researchers from Facebook AI (Jing et al., 2022) studied what they called *dimensional collapse*, occurring during contrastive learning tasks. This term describes the phenomenon of embedding vectors spanning a subspace that is of a lower dimensionality than the available embedding space. In their experiments, they use a version of contrastive loss, and in contrast to mode collapse (I. Goodfellow et al., 2014), observe only the collapse of parts of their embedding vector. Thus, while the vector does not collapse to a single mode or point in space, information is unnecessarily lost, by not using the entire embedding space dimensionality.

It is understood that a loss function like triplet or contrastive loss is the mechanic at play, preventing complete collapse by virtue of using its negative sample to prevent vectors representing different image types from collapsing around the same points. In order to remedy the phenomenon of dimensional collapse,Jing et al. (2022) propose the use of linear projectors, with which they have demonstrated promising results.

While also being a phenomenon occurring in (self-supervised) contrastive learning, dimensional collapse differs from the phenomenon we observe. We believe this to be the case, since what we observe is notably the increase in correlation between learned features, along with the simultaneous increase in data dimensionality and decrease in prediction accuracy. Both are not affected by small changes to our network architecture, as would be the case when employing a projector.

Another approach related to such phenomena is the *Barlow Twins* model (Zbontar et al., 2021). Like other contributions (Hua et al., 2021; Miklautz et al., 2020) it addresses the effect that representation learning often results in redundant features (instances in the vector representation that are linearly dependent or zero), and is a method to remove those features, resulting in increased performance for self-supervised learning or clustering tasks.

Another perspective on the phenomenon of correlation and redundancy is given by pruning (Ayinde et al., 2019), focusing on accelerating the learning process. While some of these contributions are somewhat related to the herein described phenomenon, we believe to study a novel phenomenon in contrastive learning, which we will elaborate upon in the proceeding sections.

12.3 Experimental Setup

This section will explain the experimental foundations of this chapter. As we will conduct multiple different experiments, we maintain a consistent setup for the experiments in this chapter. Thus, for simplicity's sake, we describe our approach here and will only highlight deviations from it further down the line.

12.3.1 Neural network structure

For this work, we train a neural network $f(x)$ to learn a representation of a given input image. To do so, we randomly generate 10,000 combinations of an anchor image (x_a^i), an image of the same entity (x_b^i), an image of another entity (x_c^j), and optimize them using the triplet loss function given previously in Equation 11.4.

We select $\alpha = 1.0$ and for the function $f(x)$ to output a 100 dimensional representation. We build the function $f(x)$ through two convolutions with a depth of 16 features, a ReLU activation function and a filter size of 3. After these convolutions, we add a max pooling layer, which reduces each dimension by a factor two. This structure is repeated two more times, after which we flatten the output and use three fully connected layers of 128 nodes with ReLU activation functions and one fully connected layer of 100 nodes without any activation function to generate the representation. The architecture is visualized in Table 12.1:

Table 12.1: Neural network architecture for our experiments.

	Layer	Nodes	Kernel	Activation
3x	Convolution	16	3x3	ReLU
3x	Convolution	16	3x3	ReLU
3x	Max pooling	-	2x2	-
	Flatten	-	-	-
3x	Fully connected	128	-	ReLU
3x	Fully connected	100	-	-

To train the model, we use Adam as an optimizer with a learning rate of 0.001 for at most 100 epochs in batches of 128. We also employ early stopping (Prechelt, 2012) with a patience of ten epochs. All our experiments are conducted using an NVIDIA A100 graphics card with 40GB of VRAM.

12.3.2 Datasets

To apply our model, we use three different re-identification datasets, which are shown in Figure 12.2. Two of these datasets are datasets related to industrial applications already introduced in Chapter 11, while the third one, *Market-1501* (L. Zheng et al., 2015), is a widely used pedestrian re-identification dataset. It consists of 32,668 images of 1,501 individuals that are recorded by six cameras. Since this dataset records a public space, the amount of images per individual are not equally distributed, unlike for the previous datasets. For this dataset, each individual is, on average, recorded 3.6 times.

By using multiple datasets, we intend to demonstrate that the phenomenon and suggested solutions in this work are not restricted to a single dataset.

Figure 12.2: Example images of the three datasets used in this study (dataset A, *galvanized-636* on the left-hand side, dataset B, *pallet-block-502* in the center, and dataset C, *Market-1501*, on the right-hand side).

Going forward, for simplicity's sake, we will call the *galvanized-636* dataset from Chapter 11 A, the *pallet-block-502* dataset from Chapter 11 B, and *Market-1501* dataset C.

12.3.3 Cross-Validation

The performance of the models used in this work is gauged using standard ranked accuracy, as is common in re-identification publications (Rutinowski et al., 2021b). To increase the reliability of our results, we employ cross-validation. We stick to the splits in Chapter 11 to ensure the comparability of our results. Finally, we provide uncertainties for our results by calculating the standard deviation across the various folds (and dividing by the square root of the number of folds).

As cross-validation is not common for dataset C, we employ the dataset split proposed by its creators, further guaranteeing comparability.

12.3.4 Quality metrics

In order to quantify the correlations that we want to study, we define what we call the *mean Absolute Correlation* (mAC) metric in Definition 10.

Defnition 10 (mean Absolute Correlation (mAC))**.** *Given the representation r_i^μ of the feature μ of sample i, we define mAC as:*

$$mAC = \frac{2}{D \cdot (D-1)} \sum_{\nu}^{D} \sum_{\mu>\nu}^{D} \cdot \| corr^{\mu\nu} \| \tag{12.1}$$

where $corr^{\mu\nu}$ represents the Pearson correlation between μ and ν

$$corr^{\mu\nu} = \frac{1}{N} \cdot \sum_{i}^{N} \frac{(r_i^\nu - mean(r^\nu)) \cdot (r_i^\mu - mean(r^\mu))}{std(r^\nu) \cdot std(r^\mu)} \tag{12.2}$$

We suggest studying the mAC metric since we believe there to be an interdependence between its value and the achieved accuracy of a re-identification model. Additionally, we also define a metric to study dimensional collapse with Definition 11.

Defnition 11 (Correlated Feature)**.** *Given the representation r_i^μ of the feature $\mu \in [1, D]$ of sample i, we define the amount of correlated features as:*

$$CF = D - rank(r) \tag{12.3}$$

Here $rank(r)$ represents the matrix rank of r.

12.4 Correlated Representations

A data representation of a higher dimensionality generally allows learning a more complicated distance or loss function. As a representation including correlations or redundancies does not utilize all of the dimensions it has access to, it stands to reason that it is also not the most efficient representation. To examine this topic further, we herein propose Definition 12.

Defnition 12 (Correlated Representation)**.** *A representation (in the sense of a learned feature) is considered redundant if it does not improve a model's accuracy due to it being correlated to other features.*

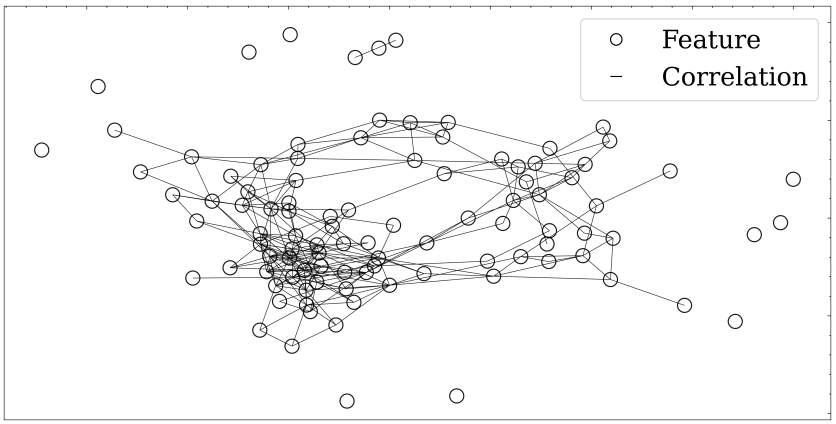


Figure 12.3: Visualization of the correlations between learned features for dataset B when using 256px images for training. Each feature is represented as a node, while the edges between them represent correlations of either $> 75\%$ or $< -75\%$.

An example of such correlations can be visualized by using a graph representation of the mAC defined in the previous section (see Figure 12.3). In such a visualization, a link, represented as edges, between two features, represented as nodes, means that these features hold a correlation of at least 75%. 89% of features represented in this figure are connected to at least one other feature and thus are at least partially redundant. We suggest using this amount of interconnections as an indicator of correlated representations.

As previously mentioned, we find there to be an interdependence between mAC and the accuracy of a re-identification model. We demonstrate this relationship in Figure 12.4. The figure shows the achieved accuracy depending on the image resolution used during training in relation to mAC. A trend that closely follows a linear dependence fit can be seen, i.e., a decrease in mAC seems to lead to an increase in accuracy.

This is related to the effect of dimensional collapse, as has been studied before in (Jing et al., 2022). In this case, as measured by Equation 12.3, the features become linearly dependent on each other and thus the effective dimensionality of the learned representation is smaller than its amount of features and thus CF increases. This effect also occurs in almost all our experiments, but with mAC we also measure an additional effect. This can be seen, as minimizing the amount of correlated representations without affecting the mAC is possible, especially since it is easy to control them using specific hyperparameter choices.

While a prominent pattern is visible for the relationship of mAC and the achieved accuracy of a model, the amount of non-correlated representations paints a different picture. As can be seen in Figure 12.5, some relationship between the amount of non-correlated features and the achieved accuracy could be present. However, this relationship does not seem

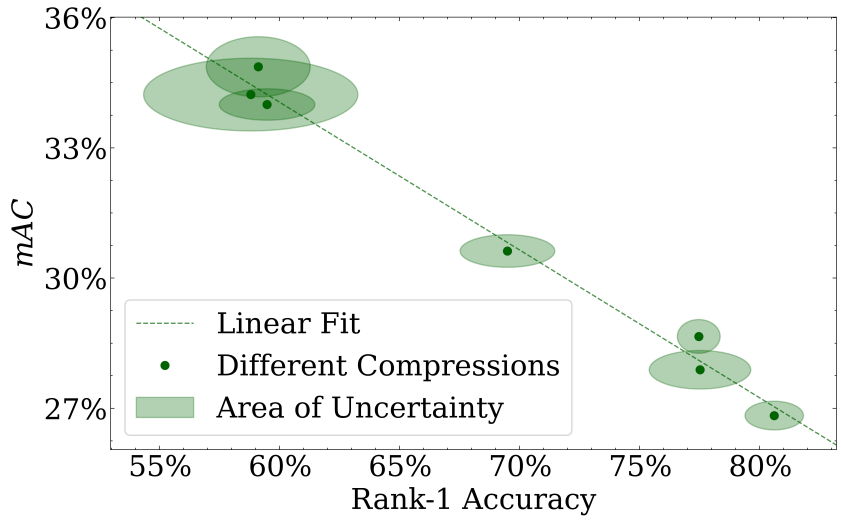


Figure 12.4: Accuracy and mAC for different networks on different compression levels on dataset A. These range from 256px in each direction to $\frac{256}{7} \approx 37$px. Notice the evident correlation (−99.4%) between the accuracy and mac value.

to be as linear and obvious as the one shown in Figure 12.4. Nonetheless, with a decrease in correlated features, an increase in accuracy also likely occurs.

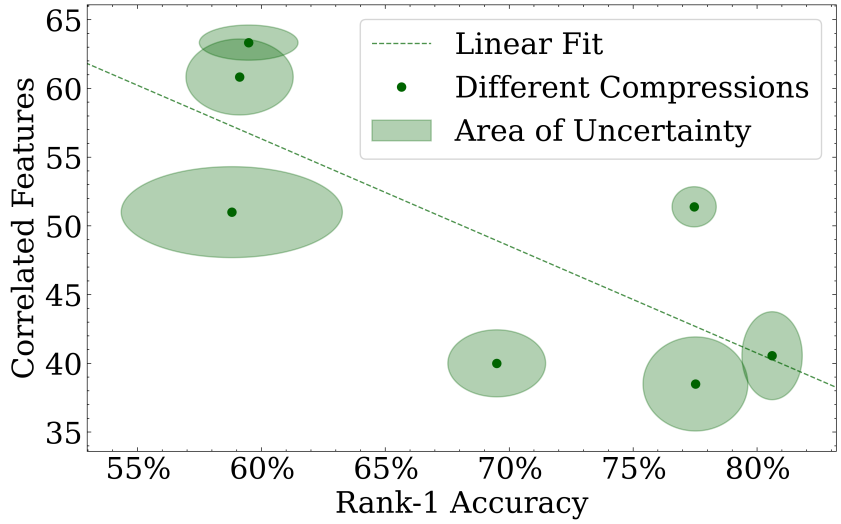


Figure 12.5: Accuracy and amount of correlated features (calculated as the difference between the dimension of the prediction matrix and the number of features) for models on different compression levels on dataset A. While there is still correlation (−75%), it is less evident than in Figure 12.4.

We believe correlated representations to occur since our neural networks contain bias. While neural networks are often considered to have a high variance and a low bias (Belkin et al., 2019), they can still only approximate arbitrarily complex functions for the impracti-

cal case of an infinite amount of nodes (Cybenko, 1989). This implies that realistic neural networks still contain bias. While the neural networks used in this chapter are still able to approximate a fairly complex function, this is not what we aim to achieve in contrastive learning. Instead, we strive to learn multiple complex functions, effectively dividing the number of available neurons per function and thus also their possible complexity.

This forces our networks to converge towards an optimum between the complexity of each learned feature and the amount of them it can learn. Notice that in both cases, the learned representation is not optimal. This also explains the observation in Figure 12.1. While there is more information available in higher-resolution images, accessing each piece of information requires more complicated functions, thus reducing their available amount and creating a worse representation.

12.5 Factors Influencing Correlation

To test this assumption and suggest possible solutions in the following sections, we test three factors that we assume to impact the extent to which correlated representations appear.

12.5.1 Representation dimensionality

If the complexity given by a neural network is limited, increasing the amount of output features should not increase the quality of the resulting representation. When there is another reason for the high amount of correlations between learned features, like a deficiency of the loss function, increasing their amount might increase the amount of information stored in them, thus improving prediction results. To test this, we present mAC and the number of correlated features for various representation dimensions in Figure 12.6.

We see the expected increase in correlated features as well as an almost constant mAC and amount of useful features. And while not shown here, the accuracy also stays almost constant. When adding more features to the learned representation, these will become redundant and added to the correlated features set, confirming our assumption.

12.5.2 Training behavior

In this section, we studied how the amount of correlated features and mAC change in relation to the training duration, i.e., the amount of epochs used. This study is presented in Figure 12.7 for datasets A and B, respectively.

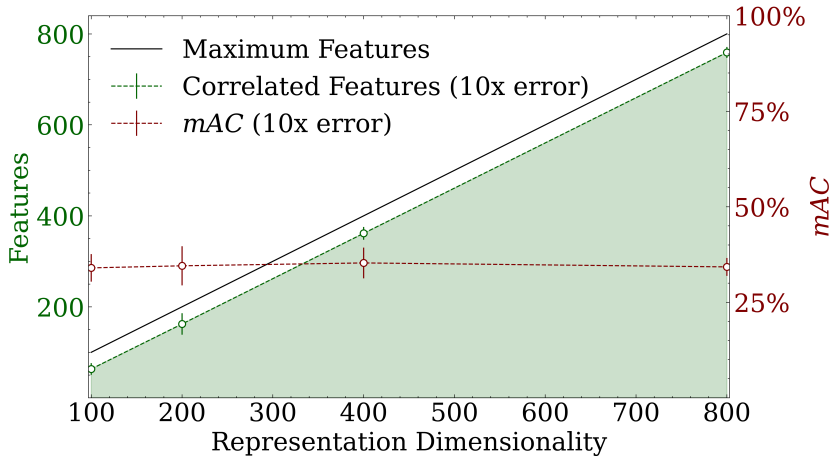


Figure 12.6: Amount of correlated features and mAC as a function of the representation dimensionality on dataset A. Notice that while the amount of correlated features grows with the representation dimensionality, this is only because the amount of features increases and the amount of non-correlated features (and mAC) stay approximately the same. We multiply the error by ten to make it more visible.

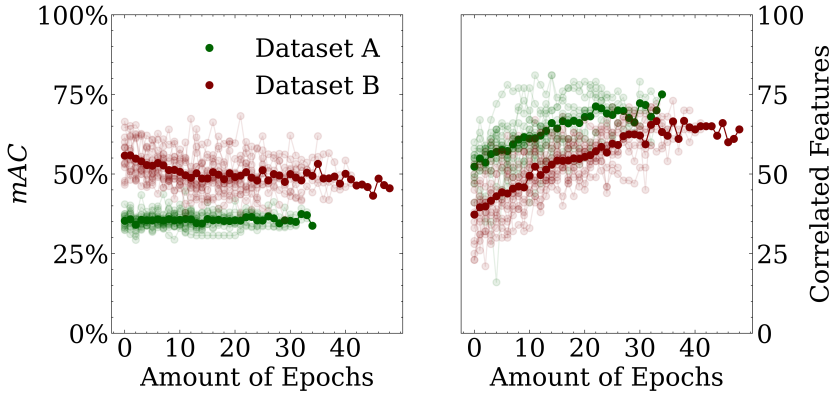


Figure 12.7: Amount of correlated features and mAC during training on datasets A and B. We show three repetitions times 5 and 6 folds, respectively, and average the results. Notice how mAC remains relatively constant while the amount of correlated features grows continuously.

For this experiment, mAC seems to stay nearly constant. This might explain the higher correlation with the resulting accuracy, as the value is not affected by changes in the training process.

Importantly, the amount of correlated features tends to start as a smaller overall percentage and increases continuously and nearly linearly during training. This again matches our assumption: There might be enough hidden nodes to generate a high amount of basic features. However, when more complex features are learned during training, more nodes are needed to generate the former. Thus, only a lower amount of more complex features

can be created, and the remaining features are then just linear combinations of those.

12.5.3 Hyperparameters

Finally, we test the influence that hyperparameter changes have on the degree to which correlated representations occur. However, since testing all possible hyperparameters and their combinations would reach beyond the scope of this chapter, we choose to focus on changes in the learning rate and activation function. We do so since both tend to greatly affect neural networks and especially re-identification models.

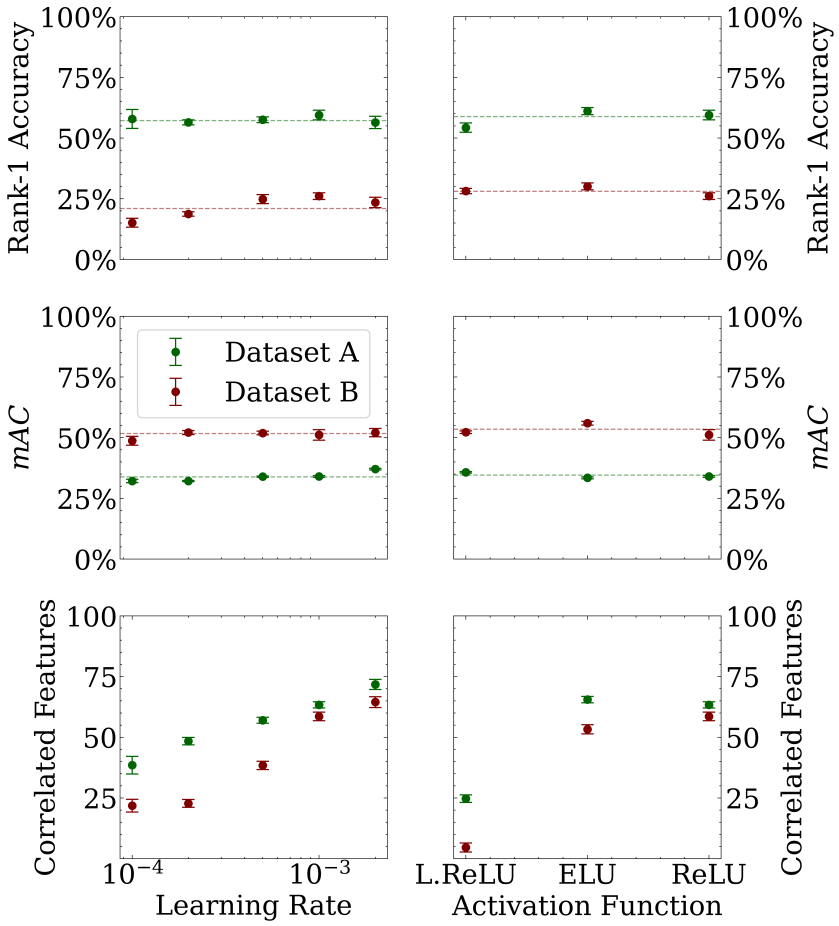


Figure 12.8: Relation between accuracy, mAC, and the amount of correlated features depending on learning rate and activation function changes for datasets A and B. While both accuracy and mAC remain nearly constant when varying the learning rate and the activation function, the amount of correlated features changes drastically. To highlight this, the average accuracy and mAC are indicated by a dotted line.

In Figure 12.8, we present the changes in accuracy, mAC, and the amount of correlated features depending on learning rate and activation function changes. We vary the learning

rate between 0.0001 and 0.02 and use ReLU, ELU and Leaky ReLU as activation functions. We chose to experiment with ReLU and its derivatives, as activation functions with multiple almost constant regions (like sigmoid or tanh) tend to provide drastically worse results.

While the accuracy and especially mAC remain almost constant, both the activation function as well as the learning rate show a noticeable influence on the amount of correlated features. Especially Leaky ReLU (J. Xu et al., 2020) leads to the lowest amount of correlated features. We believe this to be the case due to the observation that correlated representations, in the way we gauge them, are closely related to the phenomenon of dead neurons (neurons that always output a constant value regardless of input (L. Lu et al., 2020)). When half of the neurons in a layer are dead, at least half of the outputs of the next layer will be correlated (ignoring effects caused by the activation function). Dead neurons are more likely to appear when the activation function possesses regions in which only minor changes occur. Thus, changing the activation so that it does not contain these regions (as Leaky ReLU does) also decreases the amount of correlated features.

Still, this does not solve the fundamental problem of limited complexity, as the almost constant mAC and accuracy indicate. Furthermore, it shows that mAC is the more desirable metric. A small amount of noise is able to remove a linear dependency between two features (decreasing the amount of correlated features), while it does not significantly decrease the average correlation (mAC).

12.6 Solutions

After studying the effects of several changes on the phenomenon of correlated representations, we present first suggestions to mitigate this effect, in order to potentially improve the performance of re-identification models. As seen in the preceding sections, a neural network at some point reaches a complexity limit of representations learned. Thus, to reduce the occurrence of correlated representations, we will focus on methods to increase this limit by reducing the bias of our networks.

12.6.1 Neural network parameters

One way in which the bias of a neural network can be decreased is to increase the amount of features of each layer. This is the usual approach taken by non-abstract re-identification methods, using very complex neural networks (Fu et al., 2021). Instead of increasing the

number of features described in the experimental setup, we increase this number using varying multipliers. The resulting effects are shown in Figure 12.9.

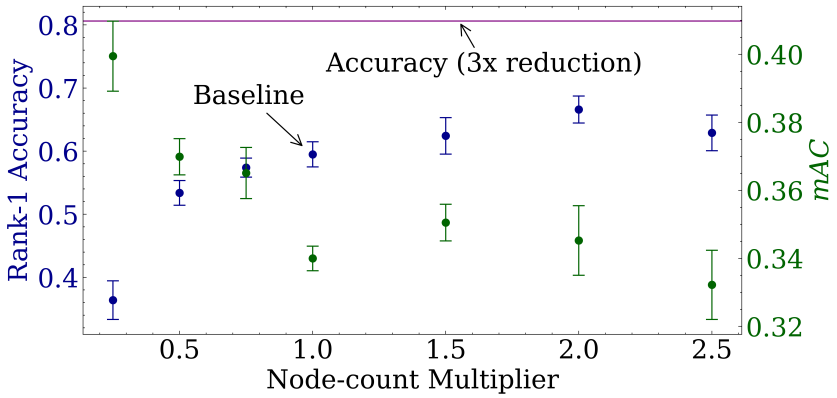


Figure 12.9: Accuracy and mAC on dataset A when multiplying the amount of features after each hidden layer. While this can improve both values, its effects remain less significant than the effect stemming from using low-dimensional inputs (depicted as the red line at the top of the diagram).

While this decreases mAC and increases accuracy, there is a limit to how much this affects the results. In addition, the expenses in terms of both memory and time are a quadratic function of the multiplication factor, making it difficult to bring this approach to scale. On the graphics card used for this experiment, we were not able to scale up our network by a factor of 3 or more.

12.6.2 Ensembling

Instead, following our experience from previous chapters, we suggest decreasing the bias of a model by replacing it with a homogenous abstract ensemble approach. This is also somewhat validated by our findings in Appendix B.5. To study this, we divide the 100 dimensional representation learned by a neural network $f(x)$ into k parts and let each part be learned by an independent network of the same architecture. Afterward, each sub-representation is concatenated into a similar 100 dimensional representation (for the case of $k = 3$, the dimensionality is 99). This allows us to compare the generated features, as shown in Figure 12.10.

We notice that ensembling is more than enough to outperform the benefits a lower dimensional representation offers (our best results through lower dimensional representations are shown as horizontal lines in the figure). The main benefit of ensembling seems to be that the non-correlated representations of one model are not the same as those of the

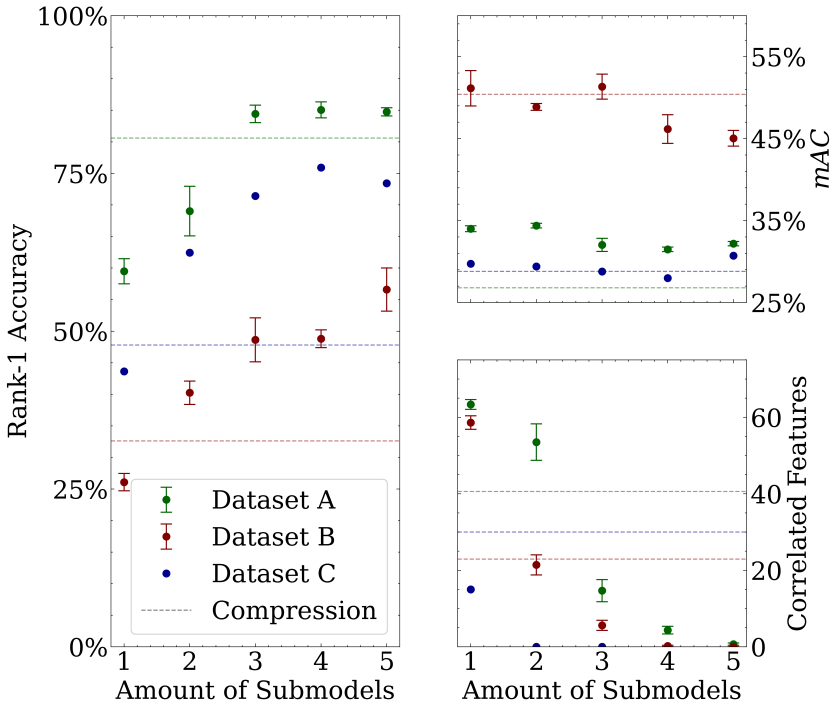


Figure 12.10: Accuracy, mAC, and amount of correlated features, when dividing the 100 dimensional representation into k independent models contributing $\lfloor \frac{100}{k} \rfloor$ parameters to the representation. All datasets are used for this experiment. We show the best results achieved through compression through horizontal lines.

next one. Notice that the amount of parameters is approximately the same for an ensemble of five submodels (8.3 million on dataset A) as for a model with 2.5 times more nodes in every hidden layer (10.3 million). Yet, the ensemble significantly increases performance, proving that our abstract ensemble approach is a more viable solution than increasing the model complexity.

We argue that this is the case since when the outputs of the hidden layers are not the same, it is less likely that two features learn to represent the very same aspect of an image. Additionally, the ensemble can easily be parallelized, further decreasing training durations.

Interestingly, while mAC seems to decrease thanks to ensembling, it is not always able to reach the same mAC as was reached through compression. This implies that while abstract ensembles outperform larger models, they might still not yet the optimal solution.

The fixed representation dimensionality still limits the results shown in Figure 12.10. At some point, the benefit of more submodels is hindered by fewer non-correlated features per submodel. To improve our results further, we test ensembles of more submodels with the same representation dimensionality. We use submodels with a representation size of 25/100, as this should allow us to capture a majority of non-correlated features without

unnecessarily bloating the representation dimensionality. These ensembles are shown in Figure 12.11.

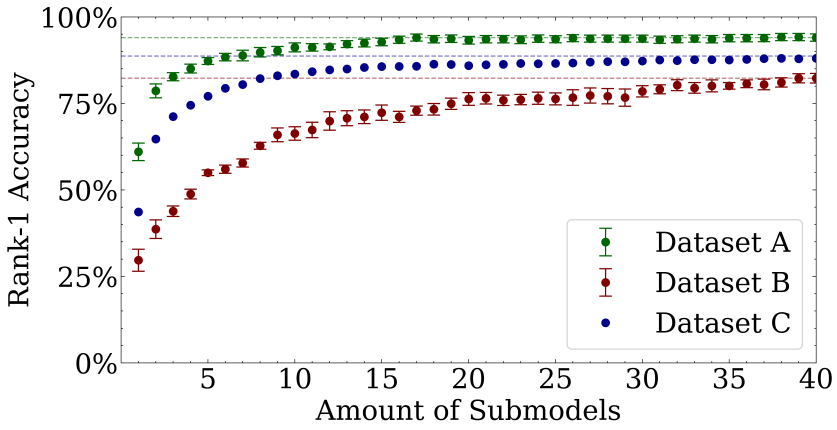


Figure 12.11: Accuracy of an ensemble of 25 dimensional submodels (for dataset A and B, but 100 dimensions for dataset C because of the lower amount of correlated features) as a function of the amount of models.

An increase in representation dimensionality helps the ensemble drastically, improving the already competitive performance in Figure 12.10 of 84.7% (dataset A) and 56.6% (dataset B) to 94.0% (dataset A) and 82.3% (dataset B). Interestingly enough, ensembles on all datasets converge at different amounts of submodel used. While dataset A reaches its peak performance from approximately 14 submodels, dataset B requires more than twice the amount to converge (approximately 35). This might be due to a more heterogeneous set of data in dataset B, displaying chipwood that might hold more individual features than the images of galvanized metal plates in dataset A. Similarly, the high amount of features in pedestrian re-identification tasks means that dataset C does not yet converge completely in the shown range. Finally, while the achieved performance of 88.7% is only vaguely comparable to other modern solutions, it shows that the same phenomenon of correlated representations also appears on a state-of-the-art dataset for a common pedestrian re-identification tasks.

12.7 Conclusion

In this chapter, we presented, characterized, and studied the phenomenon we call correlated representations. We conducted some first experiments in which we attempted to demonstrate the occurrence of this newly defined phenomenon when performing a common contrastive learning task. We applied our models to multiple re-identification datasets,

showing that the phenomenon occurs in contrastive learning-based re-identification tasks in general. We delved deeper into the effects that correlated representations have on re-identification accuracy, explained it through the bias of our neural networks, and finally proposed some preliminary solutions. By doing so, we were able to mitigate the accuracy-decreasing effect of the herein presented phenomenon and increase the performance further, by suggesting a DEAN-like abstract ensemble approach. We show that our abstract ensembles are more effective than a more complex neural network approach.

Nonetheless, our results are preliminary and invite future research, e.g., by using more datasets and models, or by applying our metrics to model selection. Notably, while the resulting accuracy is promising, the improvement of one of our metrics, mAC, is not yet as substantial as we would have expected, potentially allowing us to improve the performance even further.

Part V

Concluding Remarks

CHAPTER 13

DISCUSSION AND OUTLOOK

13.1 Overview

In this thesis, we studied the use of abstract ensembles for anomaly detection. Through a combination of theoretical analysis and experimental investigations, we demonstrate that such an ensemble can find anomalies accurately and more reliably and can do so in less time compared to the usual approach of possibly complex single models. We also show that they can help create more adversarial robust models and are more interpretable.

We introduce three anomaly detection algorithms based on the concept of abstract ensembles, each designed for a specific use case. Our general abstract ensemble approach for anomaly detection, DEAN, is highly adaptable to different tasks and delivers competitive performance, though it is relatively slow. To address this limitation, we propose SEAN, which is over 20000 times faster, making it highly practical, albeit with a slight trade-off in performance. Finally, we are the first to propose using test-time training for anomaly detection with our algorithm DOUST. While optimizing a DOUST model is less reliable than optimizing DEAN or SEAN and requires large amounts of unlabeled data, it achieves a substantial performance improvement. All three methods are benchmarked against state-of-the-art competitors, consistently maintaining highly competitive performance. This even stays true, when better hyperparameters are introduced for the competing methods, further validating their effectiveness for anomaly detection.

Afterward, we study if we can extend our abstract ensemble approach to other machine learning tasks, especially re-identification, where we also find that such an abstract ap-

proach can be more effective than simply increasing a model's complexity.

Overall, we have shown that abstract ensembles:

- produce highly *accurate* models, with DEAN (Chapter 4) and SEAN (Chapter 5) performing better than most state-of-the-art competitors and DOUST (Chapter 8) performing even better under certain conditions. Similarly, our re-identification method also performs highly competitively (Chapter 12).
- can be *faster* than non-ensemble methods, with SEAN (Chapter 5) being faster than every competitor considered and our heterogenous ensemble for re-identification in Chapter 11 being significantly faster than large neural network approaches.
- show more *robustness* to adversarial attacks than other deep learning approaches (Chapter 6) and that the ensemble structure can be exploited to increase properties like the adversarial robustness even further.
- allow for *interpretability* both because of the simplicity of our submodels (Chapter 11) and through exploitation of the ensemble structure. The latter, as studied in Chapter 7, allows to accurately approximate Shapley values for such ensemble methods in polynomial time and can be used to further understand the type of features considered anomalous by an anomaly detection algorithm.
- are more *reliable* than non ensemble methods, with the repetition uncertainty in Chapter 4 and Chapter 11 being lower than of competitors and the impact of hyperparameters being close to neglectable in certain ranges, as seen in Chapter 4 and Chapter 5.

Thus, we conclude that abstract ensemble approaches offer a versatile and powerful framework for anomaly detection, re-identification, and possibly further machine learning tasks, surpassing traditional methods in accuracy, efficiency, robustness, interpretability, and reliability.

13.2 Limitations

While the abstract ensemble approach studied in this thesis can likely be applied to many different tasks, our analysis in this thesis has some limitations.

We study anomaly detection only in the weakly supervised case (with only access to labeled normal data) and implicitly in the unsupervised case (without any labeled data).

Interesting results might also be obtained when considering the unsupervised case explicitly or when given access to labeled anomalies in the (semi-) supervised case.

Similarly, we mainly studied tabular and image-based data here. While some benchmark datasets (Han et al., 2022) are also constructed from e.g. NLP data, these were converted into a tabular format before usage. Explicitly studying other types of data, like NLP, graph, or time-series data, would likely provide their own challenges and might produce more powerful methods in these specialized fields. Examples of these include the use of test-time training for feature drift or the scalability of our methods to extremely large graphs.

While our methods (especially Chapters 5 and 11) are relatively fast, this speed is mainly the result of more abstract models and parallelization. Further speedup could likely come from more efficient GPU usage. This was not studied in this thesis because of drastically easier access to many CPU over GPU cores.

We have considered at most basic hyperparameter optimizations in many of the previous chapters. For our work on anomaly detection, this is partially justified by the lack of available feedback in common weakly supervised anomaly detection tasks. We were not able to extend our more fair approach to hyperparameter optimization in Chapter 10 because of computational limitations. Similarly, we have not considered the impact of hyperparameters in our work on re-identification. This is because we care more about the general viability of our approach than for highly competitive results.

Additionally, our analysis of re-identification methods was mainly focussed on the re-identification of logistic entities, like pallet blocks and metal crates. And while the results of Chapter 12 seem to also generalize to a common person re-identification dataset, further studies on more common re-identification tasks might result in interesting observations.

13.3 Future Work

Overall, ensembles for AD are a big field, and many ideas are still unexplored. We want to highlight the work of two thesis students here. Rao (2022) argues for training multiple submodels at the same time to improve submodel variances, and P. H. Nguyen (2023) argues for a boosting method that also considers the anomaly scores of the previous model. This shows that the research on ensemble methods for anomaly detection is not yet fully explored. Similarly, Chapter 5 has shown that it is possible to extend classical ensemble techniques like feature bagging further.

One possible direction might be to test if we can create even simpler and more abstract submodels than DEAN and SEAN.

One problem that this might solve is visible in the results of Chapter 7: Training models on very complicated data like CelebA results in a misalignment between the type of anomaly found and searched for. Training simpler models allows cost-effective training on larger datasets, like, for example, YFCC100M (Thomee et al., 2016), which historically often helps improve the performance of machine learning models (Vaswani et al., 2017). Such a study could further benefit from studying transfer learning for anomaly detection, which might be supported by a pruning approach similar to the one shown in Chapter 6.

Similar approaches could also be useful for trustworthy machine learning applications like fairness (currently studied by a thesis student (Benning, 2025)) or to adapt to the occurrence of feature drift.

We also believe that modifications of our test-time training approach in Chapter 8 would be interesting. Some extensions have been studied by thesis students. Ok (2024) studied using test-time training for classical anomaly detection algorithms and Hariharan (2024) uses test-time data to select an anomaly detection model and set of hyperparameters to use for a specific task, similar to the analysis undertaking in Chapter 10. Further work could include applying test-time training to extremely fast algorithms like SEAN to solve the reliability problems of our approach. Alternatively, one could use test-time training to create ensembles similar to the supervised case. Still, all of these approaches likely require a deeper understanding of the minimum amount of data required for the detection of anomalies, as studied in Chapter 9. For this, we expect that either an analysis with further datasets, including real-world ones, or studies in human perception could help.

While we have applied a general abstract ensemble approach towards re-identification, it might also be interesting to study the further concepts we developed for anomaly detection applied to re-identification. This could include adversarial robustness, interpretability, or a large ensemble of very fast shallow approaches. Some initial work towards the last one was already studied by a thesis student (Grimme, 2023). This might be especially interesting, as Figure 12.10 implies that such ensembles might not be the optimal solution for the problem of correlated representations we discussed in Chapter 12. Additionally, trustworthiness in re-identification becomes more and more important. This was studied in an unrelated paper (Rutinowski, Klüttermann, et al., 2024). As abstract ensemble results in a more trustworthy approach towards anomaly detection, it might also be possible to extend this to re-identification.

There is also a connection between re-identification and anomaly detection, as novel objects keep appearing in re-identification applications. This was also studied by a thesis student (Sadari, 2024). However, as we propose approaches based on abstract ensem-

bles for both, it might be possible to create an integrated approach for doing both re-identification and anomaly detection at the same time. This would allow using the information from the re-identification representation to improve the anomaly detection, and the normality of a sample according to the anomaly detection part to help weight its impact during the re-identification training.

By showcasing the effectiveness of abstract ensembles across diverse use cases, this thesis highlights their potential to advance anomaly detection and related fields. We hope that these findings encourage further research into their applications, fostering new methodologies and their broader adoption in machine learning.

APPENDIX A

APPENDIX A - FURTHER IDEAS AND EXPLANATIONS ON TEST-TIME TRAINING

A.1 Setup Details

A.1.1 Experimental Setup

A.1.1.1 Model Charateristics

Our models use a neural network ($f(x)$) built from three layers with 100 nodes and one output layer with 1 node. Following the analysis in Section 4.4.1, all layers use no bias term. Every layer except the last one is activated with a relu activation; the last one instead uses the modified sigmoid activation, as shown in Equation 8.2. We use a sigmoid-like activation because a fixed output range simplifies our loss, but we have to shift its values slightly to make sure the function $sigmoid(0) = \frac{1}{2}$ is not a trivial solution.

This network is trained in two steps. In the first training step, we only consider training data and minimize the DEAN-like loss in Equation 8.1: Every (training) sample should be mapped close to $f(x) = \frac{1}{2}$, so close to the center of the available output range. For this optimization, we use 5 epochs and a batch size of 100. We did only slightly optimize these parameters and think there are likely better values that could be chosen, but we also don't believe them to affect our results strongly (See Chapter 10). This first training step results in all samples being mapped into a range that we can optimize easily (there is no $f(x) \approx 0, 1$). Additionally, the output values are close to each other for normal samples. This already

works as an anomaly detector similar to our analysis in Chapter 4, but works drastically worse when compared to the second optimization step. We choose a low number of epochs here, as having learned imperfect representations seems to allow the second step to change our predictions more easily.

For this second training step, in which we also use test-time data (and which thus can only be done as part of the evaluation), we declare labels of 0 for each training sample and 1 for each test sample and try to minimize the distance between label and function output $f(x)$. This is done, for example, with the loss function given in Equation 8.3, or the other loss functions studied in Chapter A.4. Here, we train for 50 epochs with the same batch size of 100. During the second training loop, we expect normal samples to be pulled to a low value and abnormal samples to the maximum value of 1. This means that after training, we can use the function output $f(x)$ as an indicator of anomalousness: The higher $f(x)$, the more likely x is an anomaly.

A.1.1.2 Ensembles

Instead of comparing the performance of a singular model, we combine 100 models into an ensemble:

$$F(x) = \frac{1}{100} \sum_{i=1}^{100} f_i(x) \tag{A.1}$$

This averages out slightly worse neural network initializations, making our results more reproducible and increasing their performance slightly. We chose to combine 100 models, as this is the same number of models as are used by the isolation forest competitor. Ensembles are studied in the ablation study in Section A.2.5.1.

A.1.1.3 Simulated Experiment Setup

The experiment in Figure 8.6 requires simulated data. Simulated data allows us to change the size of the test set in arbitrary ranges and still compare different test sets fairly.

We choose data that is sampled from a ground truth Gaussian distributions in ten dimensions. The normal samples have a mean of $\mu = \vec{0}$ and standard deviation of $\sigma = \vec{1}$. The anomalous samples follow a similar distribution centered around $\mu = \vec{1}$.

On a training set of N samples and a test set build from $0.99 \cdot N$ normal samples and $0.01 \cdot N$ anomalies, we train our algorithm (As described in Appendix A.1.1.1). We ignore the small performance gain described in Appendix A.2.5.1, and train just one submodel

for each ensemble. Instead, we train models on 1000 differently sampled datasets for each value of N and average their performance (compared to their predictions as in the ensemble). This removes uncertainties that we get from the random generation of samples. These are especially critical for low N; for the minimum value of $N = 1000$, we only generate $1000 \cdot 1\% = 10$ anomalies.This means a single sample randomly overlapping with the normal distribution can result in a difference of 5% in ROC-AUC (see Appendix A.2.1.1).

For our competitor, we choose a random forest trained on the maximum number of samples studied for our algorithm (one million). This algorithm is repeated ten times, and the performance is averaged, even though the repetition uncertainty is neglectable.

A.1.2 Comparison Algorithms

A.1.2.1 Weakly Supervised Algorithms

We choose our competitor algorithms following a recent survey (Han et al., 2022). We take the three best algorithms: An isolation forest (ifor) (F. T. Liu et al., 2008), a k-nearest neighbor algorithm (knn) (Gu et al., 2019), and a cluster-based extension of the local outlier factor (cblof) (Z. He et al., 2003). While many more algorithms are possible, we want to keep the number of algorithms limited (increasing the number of algorithms increases the critical difference needed for a significant difference). Following the survey, many more algorithms perform only statistically insignificantly worse than these three algorithms, and thus, we can not say that these algorithms are the best on general data. But since we also use datasets from the same survey, we can say that these are likely the best-performing algorithms on these commonly used datasets. Following this reasoning results in us only using shallow competitors but comparing them to a deep learning method. However, as clearly shown in the survey, using deep learning competitors would only result in an even more significant difference between DOUST and the competitors.

We also use the same implementation as used in the survey. It is implemented in pyod (Y. Zhao et al., 2019), and we follow standard hyperparameters. The exception is the knn algorithm, where changing the number of nearest neighbors considered from $k = 5 \rightarrow k = 1$ improves this competitor's performance (Gu et al., 2019).

A.1.2.2 Supervised Algorithms

We use a supervised algorithm not as a competitor but more as an upper limit for anomaly detection algorithms. When an algorithm does not have access to any labels, how can it perform better than one that has labels? But in practice, there are two reasons why this does

not have to be the absolute upper limit. Training on highly imbalanced data can result in trivial optima, and test time training can allow us to specify our model to work best on the specific test data used.

Since supervised test time classification methods are not reliable enough to be a viable competitor, especially when considering limited data, we accept that our limit might not be absolute. And as Figure 8.6 shows, we are able to outperform it (even when the data is not imbalanced).

To work with the imbalanced data, we choose to use a random forest algorithm (Breiman, 2001) as our supervised algorithm. We use the sklearn implementation (Pedregosa et al., 2011) with standard hyperparameters. A random forest works well with low amounts of data (allowing us to learn only on the test set), and the invariance to normalization also makes training them more reliable. We also considered comparing this against the best semi-supervised algorithm from the previously mentioned survey, XGBOD (Y. Zhao & Hryniewicki, 2018), but chose not to do so, as the performance is worse than the supervised algorithm.

Similarly, we do not compare against PU-learning approaches. PU-learning approaches for anomaly detection tasks are rare, and thus, there is no agreement in the community on a viable comparator. Our implementation of (J. Zhang et al., 2017) does barely reach a comparison performance comparable to our weakly supervised competitors.

A.1.2.3 Crossvalidation

With exception of the weakly supervised algorithms, the other algorithms require access to labeled anomalies. Since we only have anomalies in our test set, we also have to use this set for training. This can result in overfitting and an unfair comparison. To solve this, we use cross-validation (Refaeilzadeh et al., 2009). We equally distribute every normal and abnormal sample in the test set into five equally sized groups. Then we train five models, where each model is trained on four datasets, and make predictions for the fifth. These predictions are combined, and we calculate the ROC-AUC on the combined dataset.

A.2 Further Ideas, Ablations and Limitations

A.2.1 ROC-AUC and Datasplits

Anomaly detection is usually evaluated with one of three metrics. Next to the ROC-AUC (Hanley & Mcneil, 1982), the AUC-PR (Boyd et al., 2013) and the F1-Score (Goutte & Gaussier,

2005) are also commonly used. But for our analysis, only the ROC-AUC can be used, which is what we want to show here.

The F1-Score can not be used since it requires boolean predictions (either normal or anomaly, not an anomaly score like we and our competitors give). While we can convert anomaly scores into these boolean predictions by using a threshold, this requires consistently choosing comparable, highly efficient thresholds. This makes every comparison much harder since a badly performing algorithm might only perform badly because its predictions make it harder to choose this threshold. Also, the reliance on boolean inputs generally increases the repetition uncertainty.

We can also not use the AUC-PR because this metric depends on the fraction of anomalies in the test set (ν). And since we directly measure the change in performance of our algorithm when changing ν, we want to make sure that we can separate performance changes from changes in our evaluation metric.

Luckily, the ROC-AUC metric fulfills both requirements, being invariant to ν and working with continuous anomaly scores. Still, that does not mean that the ROC-AUC is perfect for our analysis (See Appendix A.2.1.1).

A.2.1.1 ROC-AUC biases

While the ROC-AUC is the only common metric invariant under the fraction of anomalies in the training set ν, making it invaluable for our analysis, this does not mean it is perfect. One drawback that we want to mention here is the difference in handling false positives and false negatives.

The ROC-AUC $roc(A,B)$ is defined as the probability of a random value in B being higher than a random value in A. To illustrate the difference, we assume the existence of worst-case samples: $w_B < a \; \forall a \in A$ and $w_A > b \; \forall b \in B$. We further use $N_{A/B} = \|A/B\|$ to represent the number of samples in A/B. Using the attributes discussed in Appendix A.2.2, we find:

$$roc(A + w_A, B) = \frac{N_A}{N_A + 1} roc(A,B) \tag{A.2}$$

and

$$roc(A, B + w_B) = \frac{N_B}{N_B + 1} roc(A,B) \tag{A.3}$$

So, when adding worst-case samples, the ROC-AUC chances by $\frac{1}{N_{A/B}+1}$. This is asymmetric since anomalies are usually rare, and thus $N_A >> N_B$.

Missing an anomaly (false negative) is punished significantly more than considering a normal point as abnormal (false positive). Because this difference disappears as long as the fraction of anomalies in the test set is $\nu = \frac{1}{2}$, it is generally a good idea to split a dataset in such a way.

In Figure 8.5, the green line performs at least similarly well to the blue line. But if we calculate the total number of mistaken samples, this changes, as the green line consistently makes the most mistakes.

The green line finds false positives on both sides. Resulting in, on average, $2 \cdot N \cdot f$ mistakes. A blue model, in the worst case where it only guesses, still picks the right side half of the time, resulting in $N \cdot f + \frac{A}{2}$ mistakes. So the green line makes more mistakes, as long as $N \cdot f > \frac{A}{2}$. In our example $A = 20$ and $f \approx 0.023$. So for $N > \frac{20}{2 \cdot 0.023} \approx 435$, the green line makes the most mistakes, even when the blue model does not work at all. For smaller values, the blue AUC is higher than the green one, implying that the most mistaken samples in our thought experiment are consistently made by the more classical green algorithm.

A.2.2 Using the ROC-AUC for model selection

There is a relation between the ROC-AUC between the training and test dataset and the ROC-AUC between the normal and abnormal parts of the test set.

For this, we assume we have data following two distributions. p_{normal} for normal samples and $p_{abnormal}$ for the anomalies.

The training set T contains N_{train} samples, while the test set E contains $N_{test} = N + O$ samples. Here $\frac{O}{N+O} = \nu$ is the fraction of anomalies in the test set.

The ROC-AUC $roc(A, B)$ is defined as the probability that a random element of B is higher than A. This implies three important attributes (Hanley & Mcneil, 1982).

$$roc(A, A) = \frac{1}{2} \tag{A.4}$$

$$roc(A, B + C) = \frac{\|B\|}{\|B\| + \|C\|} roc(A, B) + \frac{\|C\|}{\|B\| + \|C\|} roc(A, C) \tag{A.5}$$

$$roc(A, B + B) = roc(A, B) \tag{A.6}$$

This allows us to rewrite

$$\begin{aligned} ROC_{train/test} &= roc(T,E) \\ &= roc(p_{normal}, (1-\nu)\cdot p_{normal} + \nu \cdot p_{abnormal}) \\ &= \frac{N}{N+O}\cdot roc(p_{normal}, p_{normal}) \\ &\quad + \frac{O}{N+O}\cdot roc(p_{normal}, p_{abnormal}) \\ &= \frac{1-\nu}{2} + \nu \cdot ROC_{normal/abnormal} \end{aligned}$$

So, the difference between the ROC-AUC of training and test set, and the ROC-AUC score of normal and abnormal samples is a linear relationship (ν is a constant for a given dataset).When $\nu \to 0$, $ROC_{train/test} \to \frac{1}{2}$ regardless of $ROC_{normal/abnormal}$ and so small uncertainties resulting from differences between training and test distribution can dominate the calculation. These uncertainties stem from the fact two datasets sampled from the same distribution only fulfill Equation A.4 if infinite samples are drawn. Thus, similarly to our experiments in Section 8.5, these uncertainties disappear when enough samples are used.

This solves another problem in anomaly detection: Since usually no labeled anomalies are known, it is not possible to evaluate how well a certain model performs. Thus, tasks like model selection or hyperparameter optimization are (almost) impossible (Ma et al., 2023). So, most applications have to rely on only limited experience when choosing their models, likely resulting in often subpar performance. But having a metric, like the $ROC_{train/test}$, that can be calculated without knowledge of anomalies can solve this since when choosing the algorithm/hyperparameter combination with the better metric, it also means it has a better anomaly detection performance.

This likely could be used to improve our algorithm further by optimizing its hyperparameters for each dataset. We have not studied this here.

A.2.3 About upsampling

As described in Section 8.5, downsampling our data has a significant impact on our detection quality. In fact, the effect is more significant compared to practical applications since we decrease both the fraction of anomalies in the test set and the size of our dataset. Both actions make our algorithm work worse. Additionally, this forces us to remove some datasets which contain too few anomalies (16 of 47, see Section 8.3.2).

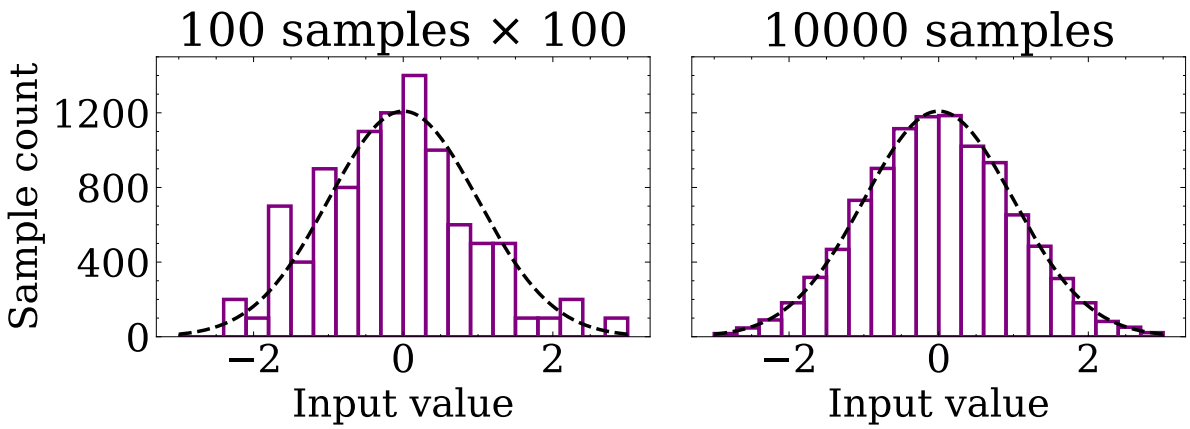


Figure A.1: Two datasets sampled from the same distribution. The left one is upsampled 100 times while the right contains 100× more samples.

This implies the question of why we do not simply upsample the existing normal data to reach the same anomaly fraction in the test set. Consider the samples shown in Figure A.1: When repeating samples, they do not follow the same distribution as when generating new data. In particular, anomalies are handled imperfectly, as they can get multiplied (right side) or stay absent (left side).

So while the separation between normal and abnormal samples might be more favorable (see the start of Section 8.5), the problem of anomalies not being clearly defined stays the same when we upsample a dataset. And since this does not happen in real data, we choose to use downsampling to be more realistic.

Similarly, we can not move samples from the training set to the test set, since this would mean that also the performance of our competitors changes. Also, this would not allow reaching values of $\nu = 1\%$ consistently, since the training set is limited.

A.2.4 Weighting loss terms

We can adapt Equation 8.3, by introducing a weighting factor ω to the :

$$L = \frac{1}{\|X_{train}\|} \sum_{x \in X_{train}} x^2 + \omega \cdot \frac{1}{\|X_{test}\|} \sum_{x \in X_{test}} (1-x)^2 \tag{A.7}$$

This modifies our optimal separation (Similar to Equation 8.4 we again assume $\|X_{train}\| = \|X_{test}\| \to \infty$)

$$L \propto \int_{\mathcal{R}^d} dx\, p_{normal}(x) \cdot (f(x)^2 + \omega \cdot (1-\nu) \cdot (1-f(x))^2)$$
$$+ p_{abnormal} \cdot \omega \cdot \nu \cdot (1-f(x))^2$$

This loss is minimal when $f(x) = 1$ for all abnormal samples, and $f(x) = \frac{1-\nu}{1+\frac{1}{\omega}-\nu}$ for all normal ones. This means, in the optimal case, normal and abnormal samples are separated by $\Delta = \frac{1}{1+\omega-\nu\omega}$, which is larger than the unweighted separation ($\Delta = \frac{1}{2-\nu}$) as long as $\omega < 1$.

Since the separation, and thus the difference between normal and abnormal samples, is minimal, when anomalies are rare, this can counteract one effect that makes *DOUST* work worse for low ν. We evaluate this experimentally in Figure A.2.

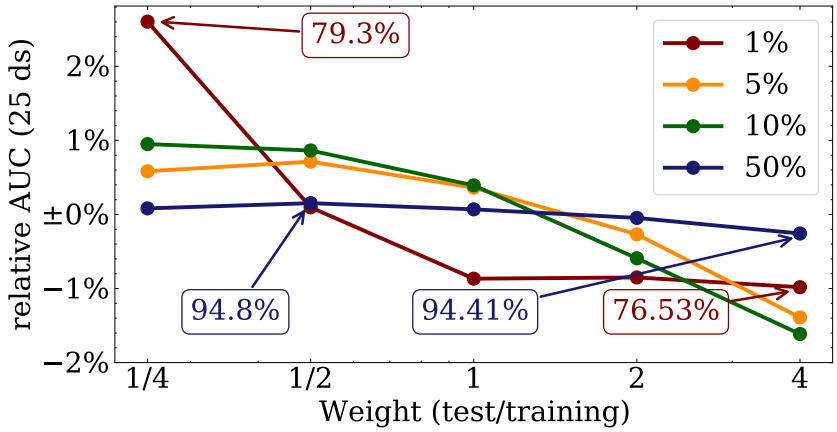


Figure A.2: Average ROC AUC, relative to the average ($\frac{auc-mean(auc)}{mean(auc)}$), for different values of ν (legend) and different weights ω (x axis).

As expected, we see a lower weight increasing the anomaly detection performance. Additionally, this effect is stronger the lower ν is, with a $> 3\%$ change for $\nu = 1\%$, but a $\approx \frac{1}{2}\%$ change when $\nu = 50\%$.

While it seems to be possible to increase the performance even further by choosing smaller and smaller weights, the magnitude of this effect is simply not strong enough to make models trained on various ν values comparable.

Especially since training with the lowest value of ν is not always optimal. A low ν means that the second term in the loss A.7 can be neglected with lower cost, resulting in the local minima of $f(x) = 0$ becoming more likely (Even though for all $\omega > 0$ this is still only a local minima). Additionally, we always assume that our training data is entirely normal, but in practice, it might also be slightly contaminated (with a fraction $\gamma < \nu$). This would modify our loss calculation to:

$$L \propto \int_{\mathcal{R}^d} dx\, p_{normal}(x) \cdot ((1-\gamma) \cdot f(x)^2 + \omega \cdot (1-\nu) \cdot (1-f(x))^2) \\ + p_{abnormal} \cdot (\gamma \cdot f(x)^2 + \omega \cdot \nu \cdot (1-f(x))^2)$$

Which is minimal when $f(x) = \frac{\omega \cdot (1-\nu)}{1+\omega(1-\nu)-\gamma}$ for all normal samples and $f(x) = \frac{\nu \cdot \omega}{\gamma + \nu \cdot \omega}$ for all abnormal ones, and thus $\Delta = \frac{(\nu-\gamma)\cdot\omega}{(1+\omega-\gamma-\omega\cdot\nu)(\omega\cdot\nu+\gamma)}$. Here, an as low as possible value of ω is no longer optimal. For example, when we assume $\nu = 0.1$ and $\gamma = 0.01$, the optimal value of ω is $\frac{\sqrt{11}}{10} \approx 0.33$.

This can also be seen in Figure A.2, where the optimal weight for $\nu = 5\%$ and $\nu = 50\%$ is $\omega = \frac{1}{2} > \frac{1}{4}$.

Choosing the right weighting factor is a way for DOUST to be further optimized, especially since Appendix A.2.2 allows finding this hyperparameter without knowing labeled anomalies. But as this would greatly increase both runtime and complexity while likely only providing marginal improvements, we leave this for further study.

Using these results, it might also be possible to estimate the fraction of anomalies in both training and test set (γ and ν), but we have not studied this.

A.2.5 Ablation Studies

A.2.5.1 Ensemble Use

To justify parts of our model setup, we add two ablation studies. The first one considers whether it is useful to combine multiple models into an ensemble, or if we should consider only the performance of one submodel.

In weakly supervised learning, deep ensembles generally provide assurance against suboptimal initialization (See Klüttermann and Müller (2023) or Chapter 3), and here they also behave similarly. In Figure A.3, we show that the ensemble improves the model a bit but also that this improvement is not very strong. The biggest improvement is seen for low values of the anomaly fraction ν, where the submodels have the highest uncertainty.

We use ensembles in this chapter, as we aim to have the best possible results, but in practice (or for simpler tasks, like in Figure 8.6), the runtime increase of a factor 100 might not be justified by a small improvement.

A.2.5.2 Feature Bagging

Additionally, in contrast to the original DEAN algorithm (Chapter 4), we suggest not using feature bagging (Lazarevic & Kumar, 2005). To justify this, we also compare this perfor-

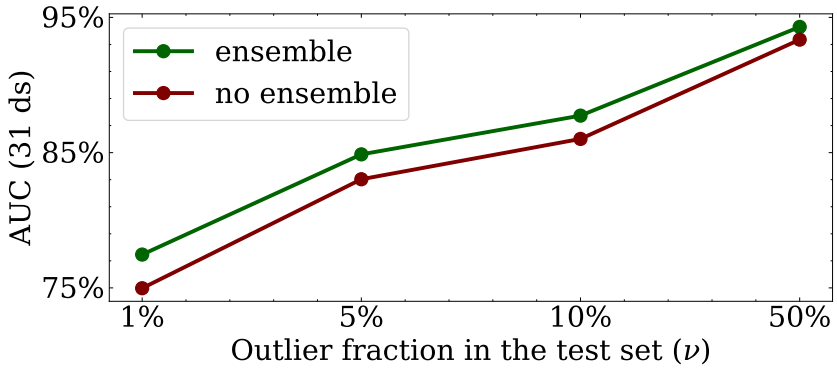


Figure A.3: Average ROC-AUC performance when using ensembles, compared to when not using an ensemble.

mance to the one we would reach with 50% feature bagging in Figure A.4 (each submodel randomly has access to half the features).

The decrease is more significant here, no longer performing comparably to the supervised algorithm. We think this is the case since often only some features allow for separation, and thus, a high fraction of models can only learn possibly counterproductive separations.

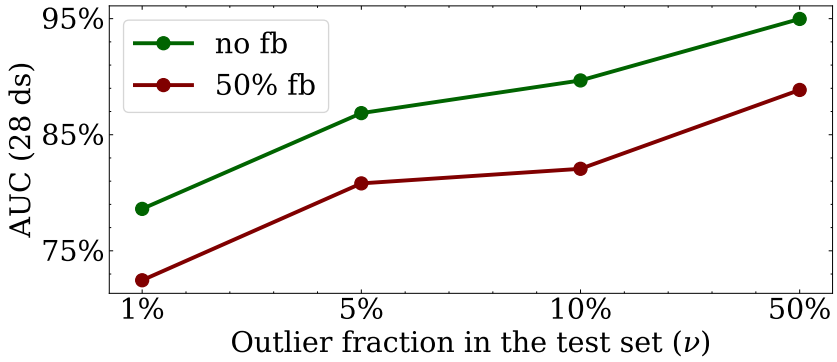


Figure A.4: Average ROC-AUC performance when using or not using feature bagging (here 50% of features are used for each submodel)

A.3 Training behaviour

To clarify the problems with our training behaviour, we have included training curves showing the loss of our DOUST method in training phase two in Figures A.5 and A.6. While the

first dataset is able to be trained reasonably well, finding a single successful training loop for the second one required us to increase the batch size.

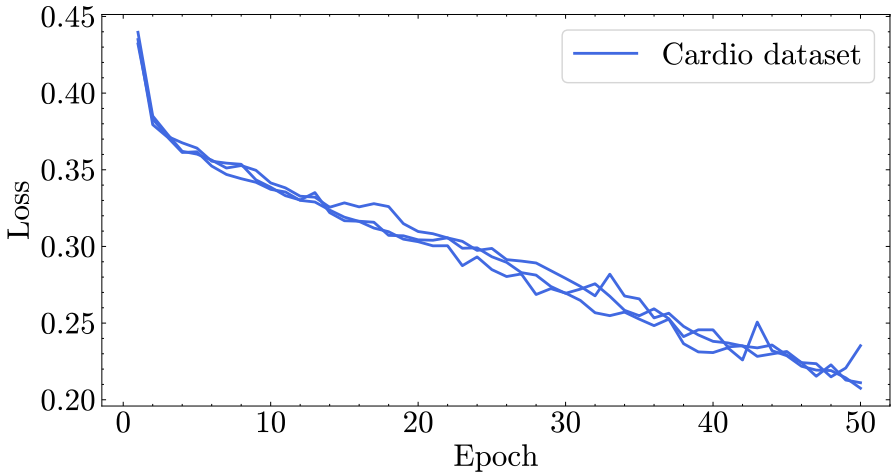


Figure A.5: Loss (Phase 2) as a function of the training epoch for the Cardio dataset. We show three repetitions with different random neural network initializations.

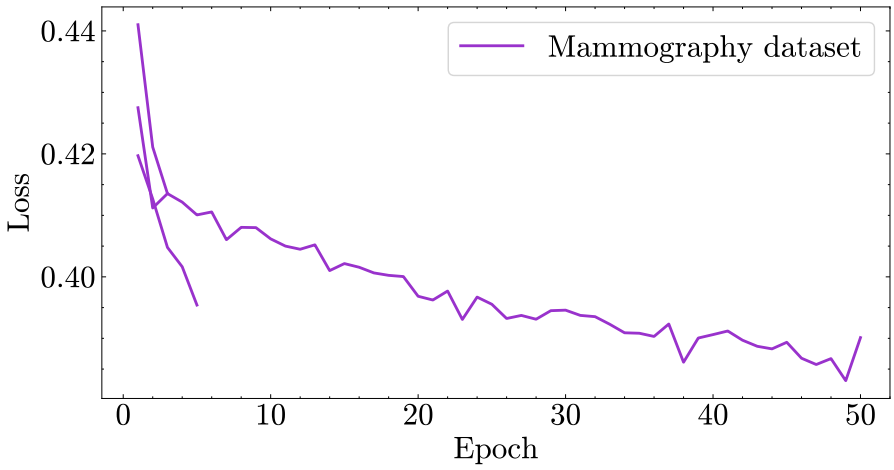


Figure A.6: Loss (Phase 2) as a function of the training epoch for the Mammography dataset. We show three repetitions with different random neural network initialization. When a training curve stops before 50 Epochs have been concluded, its training failed with a NaN-valued loss. To achieve the finished training, we used a higher batch size (×2).

A.4 Comparison of alternative loss functions

A.4.1 A fraction independent loss

One idea to create a model that does depend less on the fraction of anomalies in the test set ν, is to use a loss function that does not change when we change ν.

For this, we search for a distance function $d(A,B)$ between datasets A and B that fulfills three axioms.

Our first axiom is scale invariance (Equation A.8). When changing the size of a dataset, without changing its distribution, $d(A,B)$ is not affected.

$$d(A, \nu \cdot B) = d(A,B) \tag{A.8}$$

Our second axiom is an addition law, in A.9.

$$d(A, B+C) = max(d(A,B), d(A,C)) \tag{A.9}$$

This equation allows us to ignore the normal samples in the test set, and only focus on the abnormal ones. (See Appendix A.2.2, for an example of a function, the ROC-AUC, which fulfills Equation A.8, but not this axiom and the problems associated with this)

And finally, we assume asymmetry in Equation A.10. Demanding asymmetry allows us to narrow down the list of potential functions and simplifies our calculations, but is not necessarily needed this strongly.

$$d(A,B) = -d(B,A) \text{ and thus also } d(A,A) = 0 \tag{A.10}$$

If we find a function $d(A,B)$ that fulfills these axioms, we can infer a loss function $L = -d(X_{train}, X_{test})$ that is independent of the fraction of anomalies in the test set.

Proof.

$$\begin{aligned} L_d &= -d(X_{train}, X_{test}) \\ &= -d(X_{train}, (1-\nu) \cdot X_{normal} + \nu \cdot X_{abnormal}) \\ &= -max(d(X_{train}, X_{normal}), d(X_{train}, X_{abnormal})) \\ &= -max(d(X_{train}, X_{abnormal})) \end{aligned}$$

■

A simple distance function, that (almost) fulfills all three axioms, is

$$d(A,B) = max(A) - max(B) \tag{A.11}$$

Proof. Equation A.8: $d(A, v \cdot B) = max(A) - max(v \cdot B) = max(A) - max(B) = d(A, B)$ (Technically, this assumes that the highest values in B and $v \cdot B$ are the same. This is usually only true if the distribution has no tails, and $\|A\|, \|B\| \to \infty$. But in practice they are at least close, which is enough to construct a loss function from this.)

Equation A.9:

$$\begin{aligned} d(A, B+C) &= max(A) - max(B+C) \\ &= max(A) - min(max(B), max(C)) \\ &= max(max(A) - max(B), max(A) - max(C)) \\ &= max(d(A,B), d(A,C)) \end{aligned}$$

Equation A.10:

$$\begin{aligned} d(A,B) &= max(A) - max(B) \\ &= -(max(B) - max(A)) = -d(B,A) \end{aligned}$$

■

We test this loss experimentally. For this, we also extend our usual batch size from 100 → 500 to account for possible differences from not fulfilling our first axiom exactly.

The resulting comparison can be seen in Figure A.7. While the performance seems indeed to be stable under changes of the anomaly fraction, the total performance is also only average when compared with our competitors.

There might be another function fulfilling our axioms that produces a better performance. A simple way to extend our loss is to change $d(A,B) = max(A) - max(B)$ to $d(A,B) = max(f(A)) - max(f(B))$ with any function $f(x)$ as this similarly fulfills our axioms.

A.4.2 Alternative losses considered

In this section, we list other loss functions that we at some point considered. This allows us to show how our algorithm's performance changes when using different losses, hopefully contributing to a better understanding of how test-time training anomaly detection works.

Next to the losses mentioned here, we also briefly considered a mean average error, and a loss similar to L_d replacing max by min. Both were not studied further, as their initial performance was significantly worse.

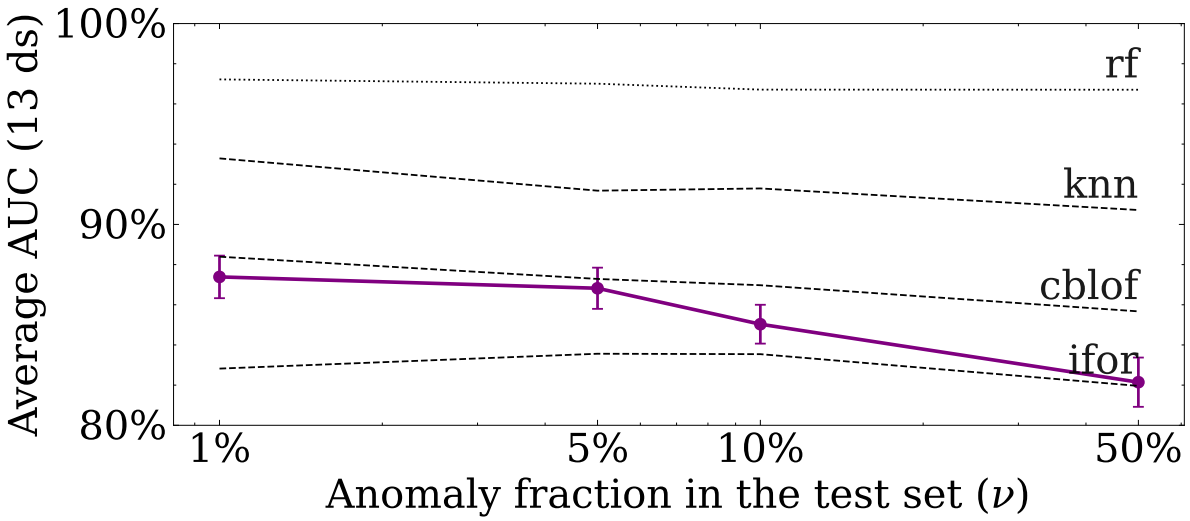


Figure A.7: Performance of our algorithm, when using fraction independent loss, as a function of the fraction of anomalies in the test set. Our competitors are shown in the background.

A.4.2.1 MSE

The loss function discussed in Chapter 8 is very similar to a Mean-Squared Error loss when assuming the label for training data to be 0 and for test data to be 1. As a kind of ablation study, we also study an unmodified mse loss:

$$L_{mse} = \sum_{x \in X_{train}} f(x)^2 + \sum_{x \in X_{test}} (1 - f(x))^2 \tag{A.12}$$

As Figure A.8 shows, the behavior is basically the same as in Figure 8.3, and so the difference can easily be neglected. This can also be seen by looking at the loss term weighting studied in Appendix A.2.4, since the difference between both loss functions is similar to a weighting term, which depends on the difference in size between the training and test set. But we still think that Equation 8.3 represents the better choice, since in real-world examples, there might be drastically more significant differences between the size of the training and test set and averaging each part allows to easier control this.

A.4.2.2 mse+mae

While a mean average loss does not perform well and was quickly discarded, we still think that a loss that is more susceptible to small differences could be useful. For this, we study

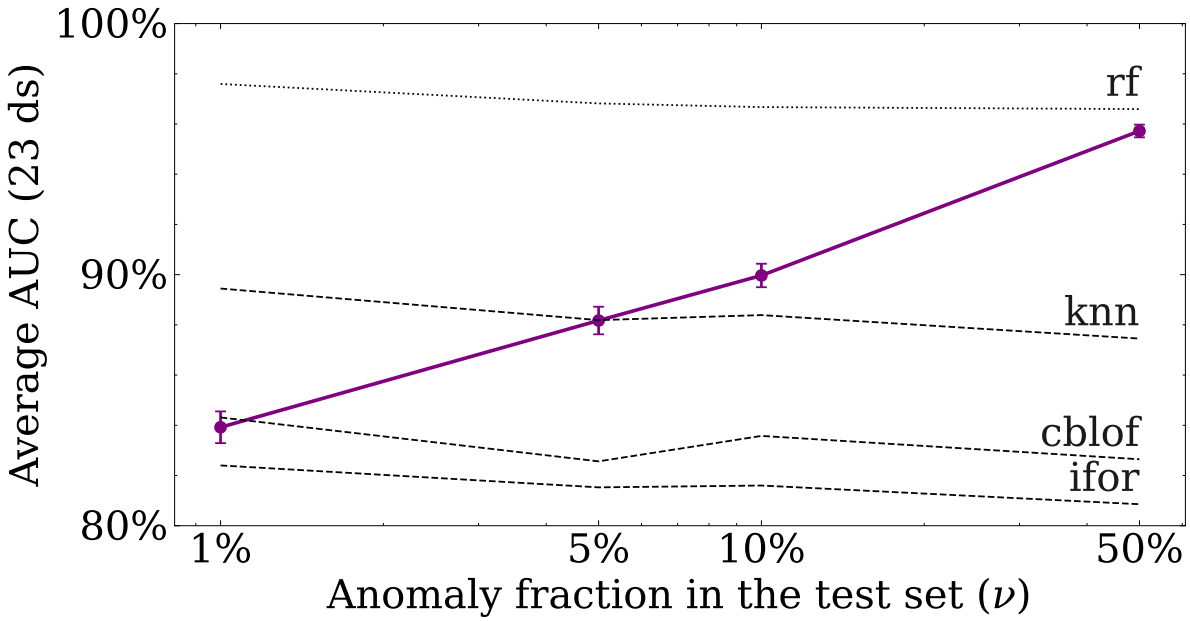


Figure A.8: Performance of our algorithm, when using an unmodified mse loss, as a function of the fraction of anomalies in the test set. Our competitors are shown in the background.

the sum of a mse and a mae loss:

$$L_{mse+mae} = \frac{1}{\|X_{train}\|} \sum_{x \in X_{train}} (x + x^2) + \frac{1}{\|X_{test}\|} \sum_{x \in X_{test}} ((1-x) + (1-x)^2)$$

But as Figure A.9 shows, the effects are also minor at most, and the performance is slightly worse.

A.4.2.3 Unmoving normal loss

Our usual loss requires both the prediction of normal and the prediction of abnormal samples to change in the second training step (during evaluation). When we set the expected value of normal samples to $\frac{1}{2}$, the prediction of no sample in the training set would need to change during the second training phase, potentially simplifying the training procedure.

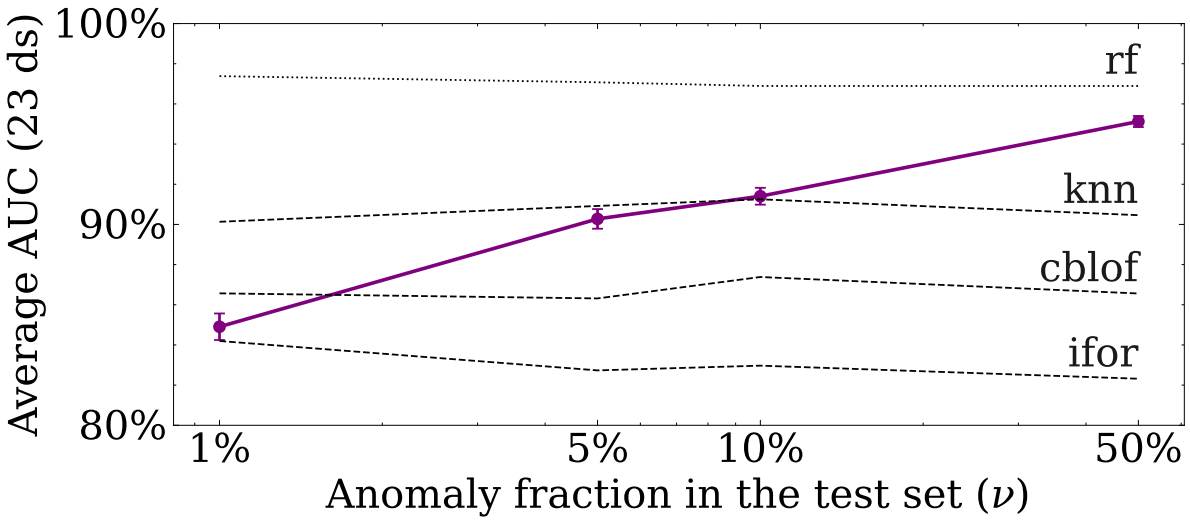


Figure A.9: Performance of our algorithm, when using mse+mae loss, as a function of the fraction of anomalies in the test set. Our competitors are shown in the background.

$$L_{unmoving} = \frac{1}{\|X_{train}\|} \sum_{x \in X_{train}} \|x - \frac{1}{2}\| + \frac{1}{\|X_{test}\|} \sum_{x \in X_{test}} (1 - x)$$

But as Figure A.10 shows, this also does not change much.

A.4.2.4 MeanMax Loss

Finally, the combination of both a fraction-independent *max* loss (that more strongly punishes high deviations) for the training samples, and an average (which can easily be separated into a normal and an abnormal part) for the test set, can provide interesting results:

$$L_{meanmax} = max_{x \in X_{train}} x + \frac{1}{\|X_{test}\|} \sum_{x \in X_{test}} (1 - x) \tag{A.13}$$

As Figure A.11 indicates, while the performance is only average for low ν, it stays stable. And while the performance of $\nu = 50\%$ is not as good as when using Equation 8.3 as a loss function, it is still better than all weakly supervised competitors. A similar approach could allow using test data to specialize predictions when it is useful but to rely on unspecialized separation when it is not.

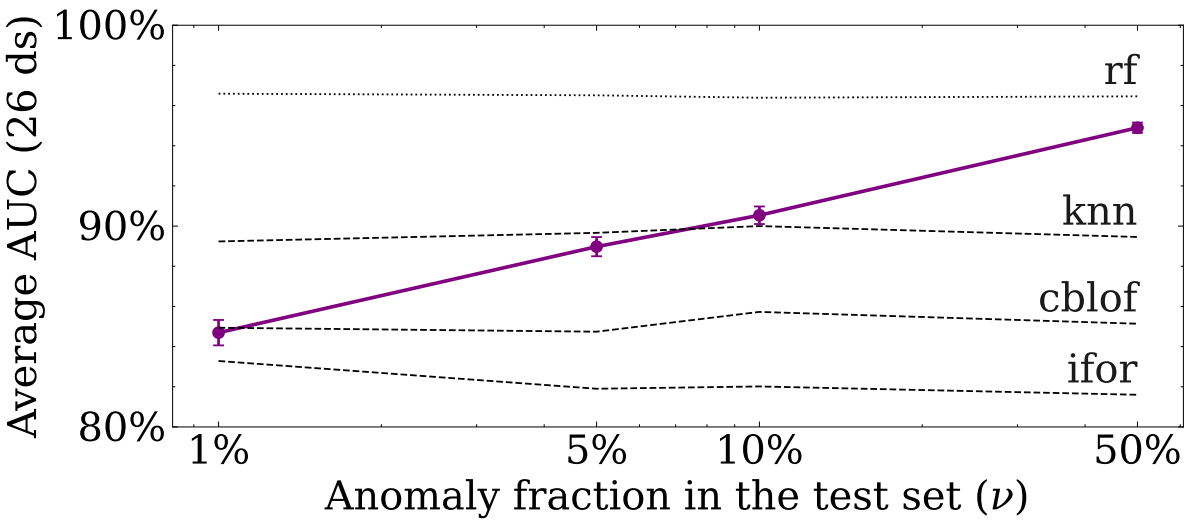


Figure A.10: Performance of our algorithm, when using unmoving normal loss, as a function of the fraction of anomalies in the test set. Our competitors are shown in the background.

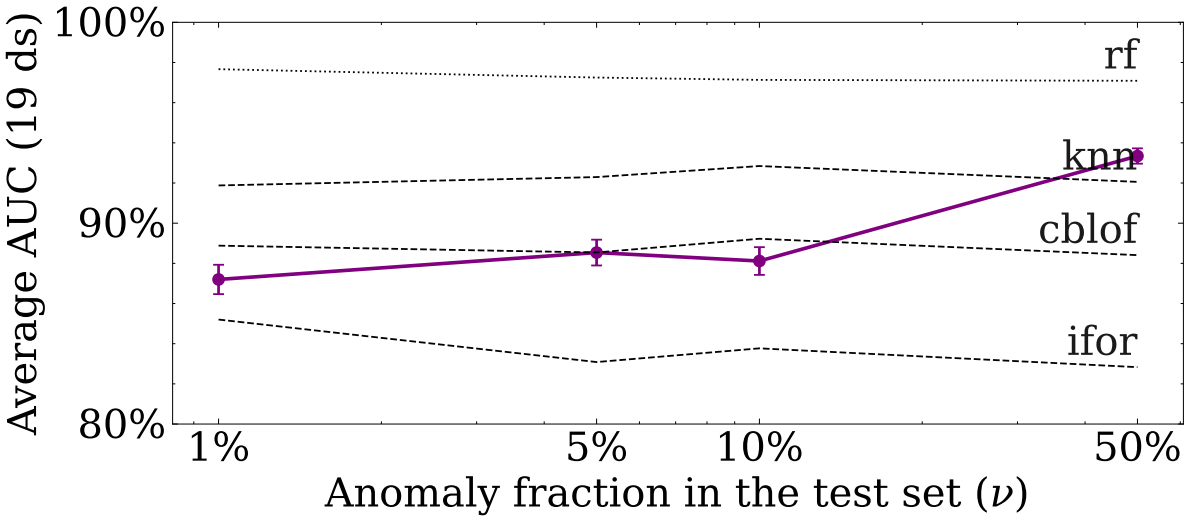


Figure A.11: Performance of our algorithm, when using meanmax loss, as a function of the fraction of anomalies in the test set. Our competitors are shown in the background.

APPENDIX B

APPENDIX B - FURTHER EXPERIMENTS TOWARDS ABSTRACT RE-IDENTIFICATION.

B.1 Time Efficiency

While each submodel improves the ensemble performance, some of our models require significantly more time to be trained than others. This leads to a trade-off between training effort and re-identification performance. Motivated by this, we display both the Rank-1 accuracy and the training duration of partial ensembles in Figure B.1.

An ensemble of all three *color*-based methods can be trained in a few seconds and still performs almost as well as the *graph*-based method with a significantly higher training cost. This highlights another use case of heterogeneous ensembles: besides increasing the overall performance of a re-identification method it may also contribute to decreasing the time required for achieving comparable performance. We expect an ensemble of many more simple submodels to outperform single complex models and be easier to use.

B.2 Ablation study on representation sizes

We tested various representation sizes for the color variance approach, to determine which representation size to choose (in this case, a representation size of 50). This is shown in Figure B.2.

Here, the performance seems to plateau from a representation size of 50 onward, which

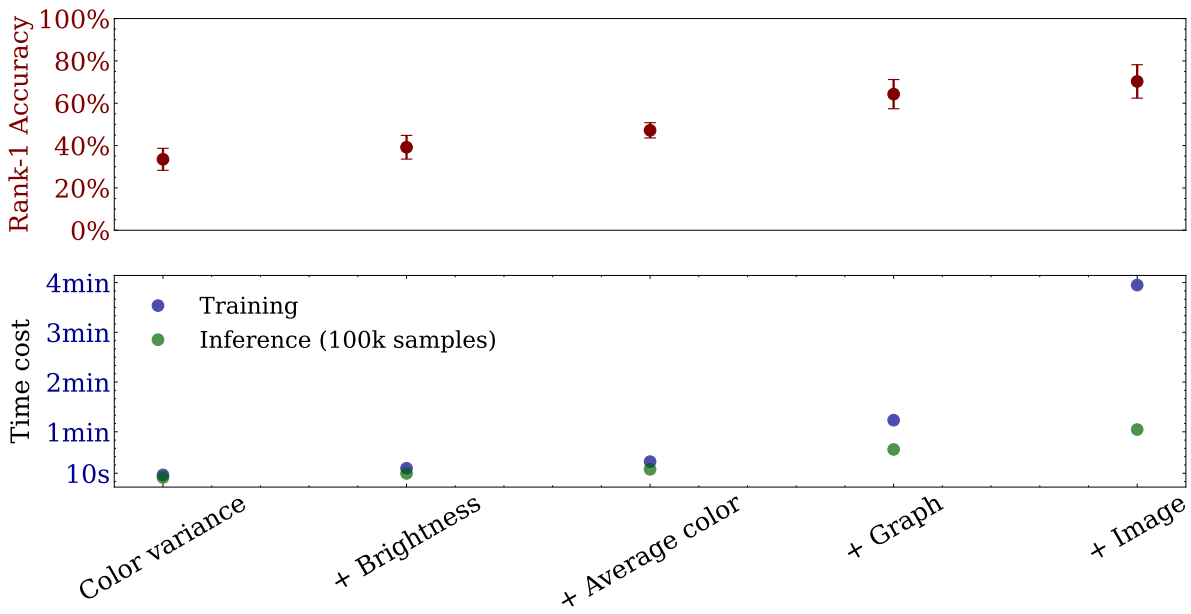


Figure B.1: Training duration, prediction cost and Rank-1 accuracy of different ensembles. The submodels are chosen to minimize the training duration. All our experiments were conducted using an NVIDIA A100 graphics card with 40GB of VRAM.

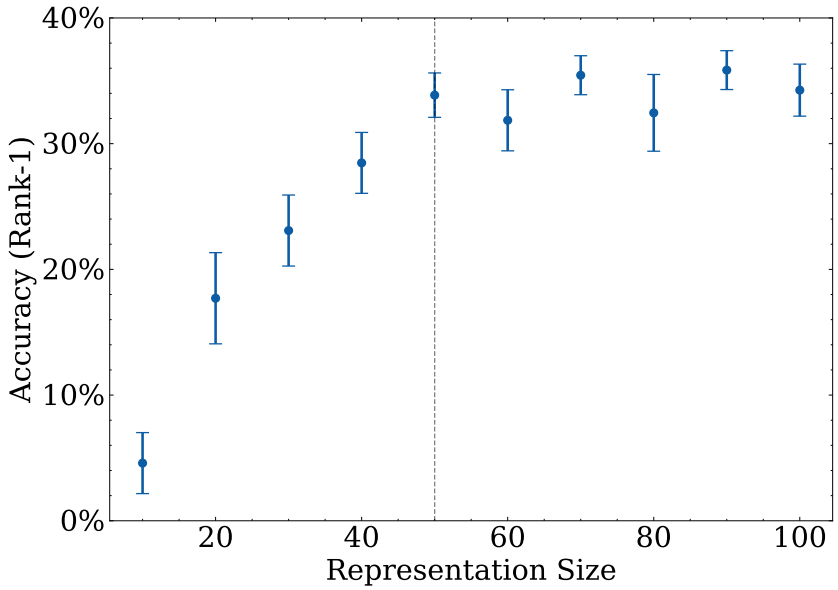


Figure B.2: Ablation study: Rank-1 accuracy of the color variance approach, as a function of the representation size used. The vertical line represents the representation size chosen in this chapter.

is why we decided to use it. Similarly we chose the same size for the rest of the submodels, except for the image-based network. Here, we chose a higher number (100), as the higher number of input features should relate to a higher number of output features.

B.3 Singular submodel impact

While we study the contribution of individual submodel pairs in Figure 11.7, another way to characterize these contributions would be to study ensembles with only 4 out of 5 submodels included. Thus, removing the most important submodel shows the lowest resulting performance. This is studied in Figure B.3.

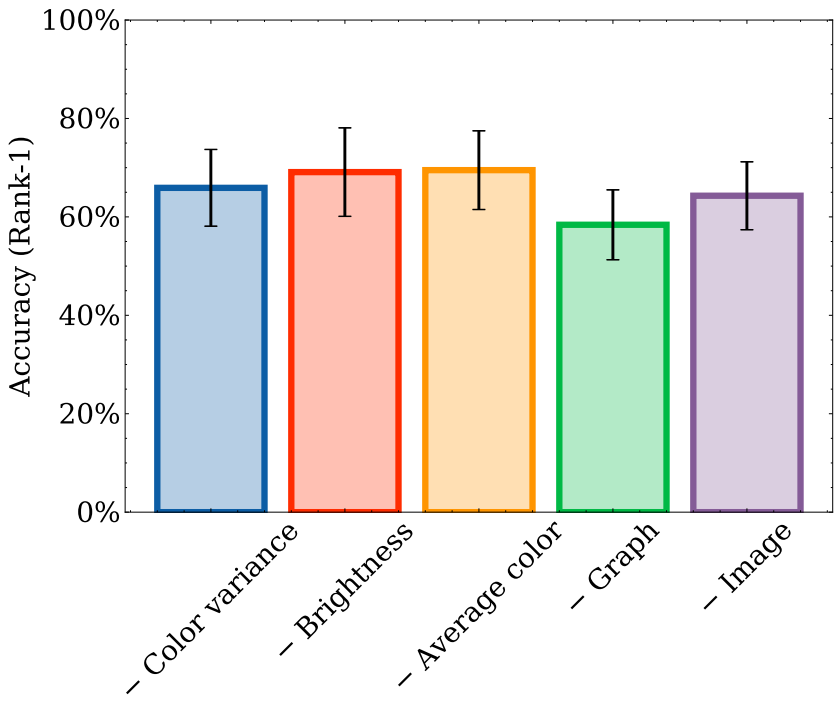


Figure B.3: Ablation study: Rank-1 accuracy for our ensemble approach when removing one of the submodel (the respective one noted on the x-axis).

While the plot shows that, as expected, the highest impact is achieved by the graph model, the differences are not significant, showing that which specific ensemble submodels to use might not be the most important choice to make.

B.4 Contribution Studies

To study our ensembles further, we present the Rank-1 accuracy improvement of an ensemble compared to its best individual model for all two-model ensembles in Figure B.4.

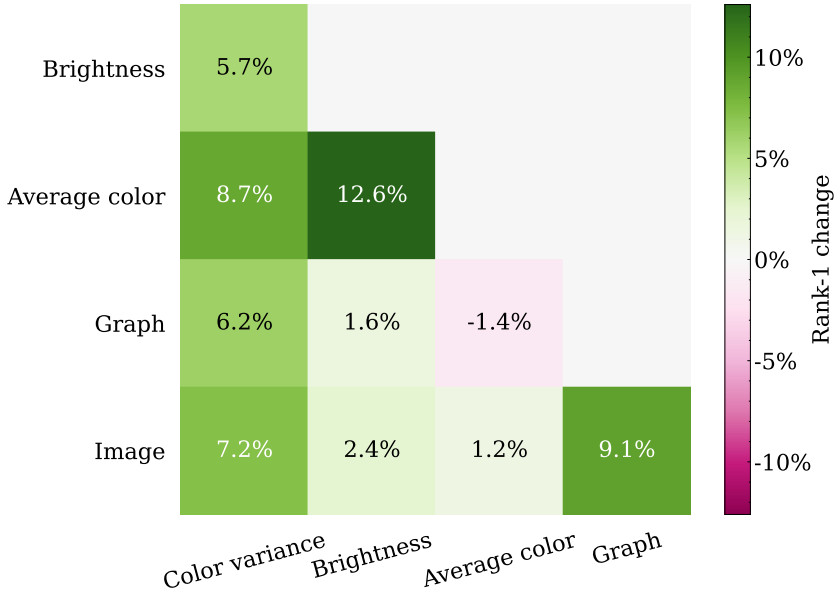


Figure B.4: Rank-1 accuracy change for all possible two-model ensembles compared to their best submodel performance.

First, it is apparent that almost all Rank-1 changes are positive. Only the combination of a *graph* representation with an *average color* leads to a negative change in Rank-1 accuracy. We hypothesize that this might result from the high uncertainty of the *average color* model and the drastic difference in the performance of both models. Most importantly, the sum of each row and column is positive, implying that adding another model always benefits the ensemble (see also Figure 2 in the supplementary material). This might also explain why we do not need any weighting factors to achieve higher performance; and it implies that even more ensemble submodels might help to improve the ensemble's performance further.

B.5 Submodel correlation

We want to characterize how far the predictions of different submodels differ from one another. Because we cannot simply use the Pearson Correlation for this, as the submodel predictions are vectors, we define triplet correlations in Alg. 3.

The correlation between two functions is calculated via the likelihood that said functions provide the same order for the distances between three samples. The results are nor-

malized so that the correlation follows the usual range of −1 to 1, with 0 representing a random chance.

Algorithm 3: Calculation of $corr_{triplet}$

Input: f, g, $x \in X$, $accuracy$

Output: $corr_{triplet}$

$count \leftarrow 0$;
$success \leftarrow 0$;
while $count < accuracy$ **do**
  Sample (A, B, C) from X ;
  $\Delta_f^{A,B} \leftarrow \|f(A) - f(B)\|_2$;
  $\Delta_f^{A,C} \leftarrow \|f(A) - f(C)\|_2$;
  $\Delta_g^{A,B} \leftarrow \|g(A) - g(B)\|_2$;
  $\Delta_g^{A,C} \leftarrow \|g(A) - g(C)\|_2$;
  if $\Delta_f^{A,B} < \Delta_f^{A,C} \wedge \Delta_g^{A,B} < \Delta_g^{A,C}$ **then**
    $success \leftarrow success + 1$;
  end
  if $\Delta_f^{A,B} > \Delta_f^{A,C} \wedge \Delta_g^{A,B} > \Delta_g^{A,C}$ **then**
    $success \leftarrow success + 1$;
  end
  $count \leftarrow count + 1$;
end
$corr_{triplet} \leftarrow 2 \cdot \frac{success}{count} - 1$;

While some correlations between our individual submodels can be perceived (see Figure B.5), there does not seem to be a significant difference between correlations nor an interesting pattern. This is interesting when comparing this plot with Figure B.4, as counterintuitively, the highest submodel correlation pair also produces the most effective ensembles. One explanation for this could be that the combination of strongly different submodels is harder, similar to the anomaly detection case (See Section 2.2). This implies that a homogenous ensemble approach with more similar submodels could improve the heterogenous case here. We show this in Chapter 12.

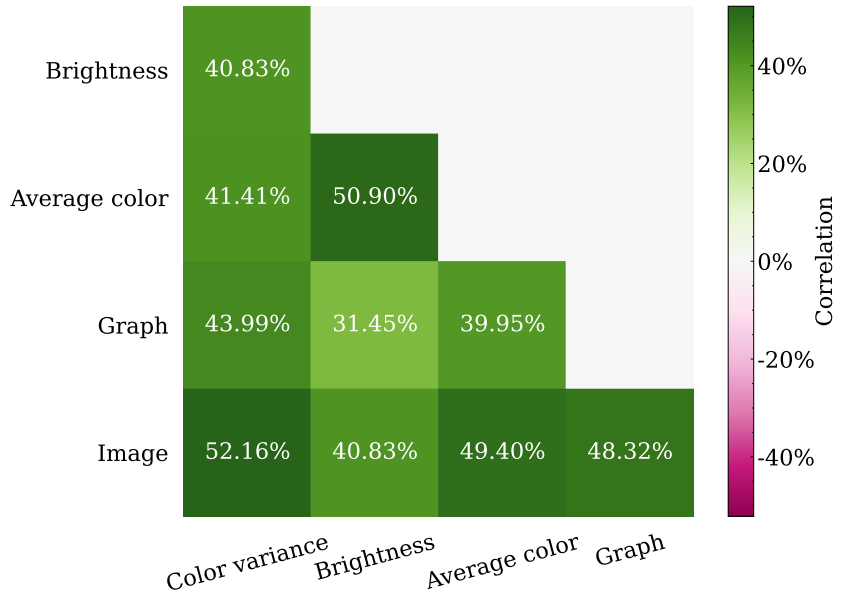


Figure B.5: Correlation (as defined in Alg. 3) between submodel predictions.

BIBLIOGRAPHY

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 34*(11), 2274–2282.

Aggarwal, C. (2012). Outlier ensembles: Position paper. *SIGKDD Explorations, 14*, 49–58.

Aggarwal, C. C., & Sathe, S. (2015). Theoretical foundations and algorithms for outlier ensembles. *SIGKDD Explor. Newsl., 17*(1), 24–47. https://doi.org/10.1145/2830544.2830549

Agosta, F., Sala, S., Valsasina, P., Meani, A., Canu, E., Magnani, G., Cappa, S. F., Scola, E., Quatto, P., Horsfield, M. A., Falini, A., Comi, G., & Filippi, M. (2013). Brain Network Connectivity Assessed Using Graph Theory in Frontotemporal Dementia. *Neurology, 81*(2), 134–143. https://doi.org/10.1212/WNL.0b013e31829a33f8

Alabdulatif, A., Kumarage, H., Khalil, I., & Yi, X. (2017). Privacy-preserving anomaly detection in cloud with lightweight homomorphic encryption. *Journal of Computer and System Sciences, 90*, 28–45. https://doi.org/https://doi.org/10.1016/j.jcss.2017.03.001

An, S., Lin, Z., Chen, B., Fu, Q., Zheng, N., & Lou, J. (2023). Does deep learning learn to abstract? a systematic probing framework. *Proceedings of the International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=QB1dMPEXau5

Anderson, T. W., & Darling, D. A. (1954). A test of goodness of fit. *Journal of the American Statistical Association, 49*(268), 765–769. https://doi.org/10.1080/01621459.1954.10501232

Archana, M., Student, N. M., Pawar, M. S. S., & Prof, A. (2015). Periodicity detection of outlier sequences using constraint based pattern tree with mad. *ArXiv*, *abs/1507.01685*. https://api.semanticscholar.org/CorpusID:1083169

Arning, A., Agrawal, R., & Raghavan, P. (1996). A linear method for deviation detection in large databases. *Knowledge Discovery and Data Mining*.

Ashok, P., Hashemi, V., Křetínský, J., & Mohr, S. (2020). Deepabstract: Neural network abstraction for accelerating verification. *Automated Technology for Verification and Analysis: 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19–23, 2020, Proceedings*, 92–107. https://doi.org/10.1007/978-3-030-59152-6_5

Athalye, A., Carlini, N., & Wagner, D. A. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *International Conference on Machine Learning, ICML*.

Avelar, P. H. C., Tavares, A. R., da Silveira, T. L. T., Jung, C. R., & Lamb, L. C. (2020). Superpixel Image Classification with Graph Attention Networks. *arXiv preprint: 2002.05544*.

Ayinde, B. O., Inanc, T., & Zurada, J. M. (2019). Redundant feature pruning for accelerated inference in deep neural networks. *Neural Networks*, *118*, 148–158.

Balestra, C., Li, B., & Müller, E. (2023). Slidshaps – sliding shapley values for correlation-based change detection in time series. *DSAA*.

Bandaragoda, T. R., Ting, K. M., Albrecht, D., Liu, F. T., Zhu, Y., & Wells, J. R. (2018). Isolation-based anomaly detection using nearest-neighbor ensembles. *Computational Intelligence*, *34*(4), 968–998. https://doi.org/https://doi.org/10.1111/coin.12156

Barghidarian, S. (2022). *Extending isolation forest to support non-numerical data* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/sina.pdf

Baudzus, A., Li, B., Jadid, A., & Müller, E. (2023). On model performance estimation in time series anomaly detection. *2023 6th International Conference on Computational Intelligence and Intelligent Systems*. https://doi.org/https://doi.org/10.1145/3638209.3638226

Bekker, J., & Davis, J. (2020). Learning from positive and unlabeled data: A survey. *Machine Learning, 109*(4), 719–760. https://doi.org/10.1007/s10994-020-05877-5

Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences, 116*(32), 15849–15854. https://doi.org/10.1073/pnas.1903070116

Bellman, R. (1966). Dynamic programming. *Science, 153*(3731), 34–37.

Benning, M. (2025). *Fairness enhancement methods in anomaly detection for dean* [Bachelor's thesis (ongoing)].

Bergman, L., & Hoshen, Y. (2020). Classification-based anomaly detection for general data. In *Iclr*. OpenReview.net. http://dblp.uni-trier.de/db/conf/iclr/iclr2020.html#BergmanH20

Bishop, C. M. (2007). *Pattern recognition and machine learning* (5th). Springer.

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., & Guttag, J. (2020). What is the state of neural network pruning? *Proceedings of Machine Learning and Systems*.

Böing, B., Klüttermann, S., & Müller, E. (2022). Post-robustifying deep anomaly detection ensembles by model selection. *2022 IEEE International Conference on Data Mining (ICDM)*, 861–866. https://doi.org/10.1109/ICDM54844.2022.00098

Böing, B., Roy, R., Müller, E., & Neider, D. (2020). Quality guarantees for autoencoders via unsupervised adversarial attacks. *Machine Learning and Knowledge Discovery in Databases - European Conference - ECML PKDD*.

Bounsiar, A., & Madden, M. G. (2014). One-class support vector machines revisited. *2014 International Conference on Information Science Applications (ICISA)*, 1–4. https://doi.org/10.1109/ICISA.2014.6847442

Boyd, K., Eng, K. H., & Page, C. D. (2013). Area under the precision-recall curve: Point estimates and confidence intervals. In H. Blockeel, K. Kersting, S. Nijssen, & F. Železný (Eds.), *Machine learning and knowledge discovery in databases* (pp. 451–466). Springer Berlin Heidelberg.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145–1159. https://doi.org/https://doi.org/10.1016/S0031-3203(96)00142-2

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*.

Breiman, L. (2001). Random forests. *Machine learning*, *45*.

Breunig, M., Kröger, P., Ng, R., & Sander, J. (2000). Lof: Identifying density-based local outliers. *ACM Sigmod Record*, *29*, 93–104. https://doi.org/10.1145/342009.335388

Brito, L. C., Susto, G. A., Brito, J. N., & Duarte, M. A. V. (2021). Fault detection of bearing: An unsupervised machine learning approach exploiting feature extraction and dimensionality reduction. *Informatics*. https://doi.org/10.3390/informatics8040085

Budynkov, A., & Masolkin, S. (2017). The problem of choosing the kernel for one-class support vector machines. *Autom Remote Control*, *78*, 138–145. https://doi.org/https://doi.org/10.1134/S000511791701011

Burgess, M. A., & Chapman, A. C. (2021). Approximating the shapley value using stratified empirical bernstein sampling. *IJCAI*.

Buschjäger, S., Honysz, P.-J., & Morik, K. (2022). Randomized outlier detection with trees. *International Journal of Data Science and Analytics*, *13*(2), 91–104. https://doi.org/10.1007/s41060-020-00238-w

Callegari, C., Gazzarrini, L., Giordano, S., Pagano, M., & Pepe, T. (2011). A novel pca-based network anomaly detection, 1–5. https://doi.org/10.1109/icc.2011.5962595

Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. d. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering, 137*, 106024.

Castro, C. M. I. P. M. (2014). Detecting falls as novelties in acceleration patterns acquired with smartphones. *PLoS ONE, 9*, None. https://doi.org/10.1371/journal.pone.0094811

Castro, J., Gómez, D., & Tejada, J. (2009). Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research, 36*(5), 1726–1730.

Cermak, V., Picek, L., Adam, L., & Papafitsoros, K. (2024). WildlifeDatasets: An open-source toolkit for animal re-identification. *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 5941–5951. https://doi.org/10.1109/WACV57701.2024.00585

Chen, D., Xu, D., Li, H., Sebe, N., & Wang, X. (2018). Group Consistent Similarity Learning via Deep CRF for Person Re-identification. *CVPR*.

Chen, J., Sathe, S., Aggarwal, C., & Turaga, D. (2017, June). Outlier detection with autoencoder ensembles. https://doi.org/10.1137/1.9781611974973.11

Chen, W., Li, H., Li, J., & Arshad, A. (2020). Autoencoder-based outlier detection for sparse, high dimensional data, 2735–2742. https://doi.org/10.1109/BigData50022.2020.9378325

Chen, Z., & Ahn, H. (2019). Item response theory based ensemble in machine learning.

Chiang, A., David, E., Lee, Y.-J., Leshem, G., & Yeh, Y.-R. (2016). A study on anomaly detection ensembles. *Journal of Applied Logic, 21*. https://doi.org/10.1016/j.jal.2016.12.002

Ciga, O., Xu, T., & Martel, A. L. (2022). Self supervised contrastive learning for digital histopathology. *Machine Learning with Applications, 7*, 100–198.

Çınar, Z. M., Abdussalam Nuhu, A., Zeeshan, Q., Korhan, O., Asmael, M., & Safaei, B. (2020). Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0. *Sustainability, 12*(19). https://doi.org/10.3390/su12198211

Colburn, T., & Shute, G. (2007). Abstraction in computer science. *Minds and Machines, 17*(2), 169–184. https://doi.org/10.1007/s11023-007-9061-7

Craig, N., Howard, J. N., & Li, H. (2024). Exploring Optimal Transport for Event-Level Anomaly Detection at the Large Hadron Collider.

Croce, F., Andriushchenko, M., & Hein, M. (2019). Provable robustness of relu networks via maximization of linear regions. *International Conference on Artificial Intelligence and Statistics - AISTATS*.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems, 2*(4), 303–314. https://doi.org/10.1007/BF02551274

Dalzochio, J., Kunst, R., Pignaton, E., Binotto, A., Sanyal, S., Favilla, J., & Barbosa, J. (2020). Machine learning and reasoning for predictive maintenance in industry 4.0: Current status and challenges. *Computers in Industry, 123*, 103298.

Decker, L., Leite, D., Giommi, L., & Bonacorsi, D. (2020). Real-time anomaly detection in data centers for log-based predictive maintenance using an evolving fuzzy-rule-based approach. *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. https://doi.org/10.1109/FUZZ48607.2020.9177762

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine, 29*.

Deviatkin, I., & Horttanainen, M. (2020). Carbon Footprint of an EUR-sized Wooden and a Plastic Pallet. *E3S Web of Conferences, 158*.

Dietterich, T. G., & Zemicheal, T. (2018). Anomaly detection in the presence of missing values. https://arxiv.org/abs/1809.01605

Dissanayake, T., Fernando, T., Denman, S., Sridharan, S., Ghaemmaghami, H., & Fookes, C. (2021). A robust interpretable deep learning classifier for heart anomaly detection without segmentation. *IEEE Journal of Biomedical and Health Informatics*, *25*.

Dodge, Y. (2008). Central limit theorem. In *The concise encyclopedia of statistics* (pp. 66–68). Springer New York. https://doi.org/10.1007/978-0-387-32833-1_50

Dong, L., Shulin, L., & Zhang, H. (2017). A method of anomaly detection and fault diagnosis with online adaptive learning under small training samples. *Pattern Recognition*, *64*.

Ehlers, R. (2017). Formal verification of piece-wise linear feed-forward neural networks. *Automated Technology for Verification and Analysis - ATVA*.

Fernando, T., Gammulle, H., Denman, S., Sridharan, S., & Fookes, C. (2022). Deep learning for medical anomaly detection - A survey. *ACM Computing Surveys*.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in Neural Information Processing Systems 28 (2015)*, 2962–2970.

Feynman, R. P. (1949). Space-Time Approach to Quantum Electrodynamics. *Physical Review Journals*, *76*, 769–789. https://doi.org/10.1103/PhysRev.76.769

Flusser, M., & Somol, P. (2022). Efficient anomaly detection through surrogate neural networks. *Neural Computing and Applications*, *34*, 1–15. https://doi.org/10.1007/s00521-022-07506-9

Fournier, Q., & Aloise, D. (2019). Empirical comparison between autoencoders and traditional dimensionality reduction methods. *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 211–214. https://doi.org/10.1109/AIKE.2019.00044

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *Science*, *315*(5814), 972–976. https://doi.org/10.1126/science.1136800

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, *11*, 86–92.

Fu, D., Chen, D., Bao, J., Yang, H., Yuan, L., Zhang, L., Li, H., & Chen, D. (2021). Unsupervised pre-training for person re-identification. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 14750–14759.

Gandelsman, Y., Sun, Y., Chen, X., & Efros, A. A. (2022). Test-time training with masked autoencoders. In A. H. Oh, A. Agarwal, D. Belgrave, & K. Cho (Eds.), *Advances in neural information processing systems*.
https://openreview.net/forum?id=SHMi1b7sjXk

Gao, J., & Tan, P.-N. (2006). Converting output scores from outlier detection algorithms into probability estimates. *Sixth International Conference on Data Mining (ICDM'06)*, 212–221.

Gigoni, L., Betti, A., Tucci, M., & Crisostomi, E. (2019). A scalable predictive maintenance model for detecting wind turbine component failures based on scada data. *2019 IEEE Power & Energy Society General Meeting (PESGM)*, 1–5.
https://doi.org/10.1109/PESGM40551.2019.8973898

Giunchiglia, F., & Walsh, T. (1992). A theory of abstraction. *Artificial Intelligence*, *57*(2), 323–389. https://doi.org/https://doi.org/10.1016/0004-3702(92)90021-O

Goldstein, M., & Dengel, A. R. (2012). Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. *Advances in Neural Information Processing Systems*, *27*.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. *International Conference on Learning Representations - ICLR*.

Gouk, H., Frank, E., Pfahringer, B., & Cree, M. J. (2021). Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, *110*(2), 393–416. https://doi.org/10.1007/s10994-020-05929-w

Goutte, C., & Gaussier, E. (2005). A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. *Lecture Notes in Computer Science*, *3408*, 345–359. https://doi.org/10.1007/978-3-540-31865-1_25

Grattarola, D., & Alippi, C. (2020). Graph neural networks in tensorflow and keras with spektral. https://arxiv.org/abs/2006.12138

Grimme, B. (2023). *Entwicklung eines algorithmus für die wiedererkennung von palettenklötzen basierend auf maschinellem lernen* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/britta.pdf

Gu, X., Akoglu, L., & Rinaldo, A. (2019). Statistical analysis of nearest neighbor methods for anomaly detection. In *Neurips* (pp. 10921–10931). http://dblp.uni-trier.de/db/conf/nips/nips2019.html#GuAR19

Gupta, N., Smith, J., Adlam, B., & Mariet, Z. E. (2022). Ensembles of classifiers: A bias-variance perspective. *Transactions on Machine Learning Research*.

Halder, R. K. (2020). *Cardiovascular disease dataset*. https://doi.org/10.21227/7qm5-dz13

Han, S., Hu, X., Huang, H., Jiang, M., & Zhao, Y. (2022). Adbench: Anomaly detection benchmark. *NeurIPS*.

Hanley, J., & Mcneil, B. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, *143*, 29–36. https://doi.org/10.1148/radiology.143.1.7063747

Hariharan, G. D. (2024). *Anomaly detection model selection using super-auc scores* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/gautam.pdf

Hariri, S., Kind, M. C., & Brunner, R. J. (2018). Extended isolation forest. *CoRR*, *abs/1811.02141*.

Häusler, A. (2023). *Using large language models for anomaly detection on text datasets* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/anton.pdf

Hawkins, D. (1980). *Identification of outliers*. Chapman; Hall. https://www.bibsonomy.org/bibtex/2a18a993511eae14be1e4afcb6781ea58/fbw_hannover

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition, 770–778.

He, Z., Xu, X., & Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters*, *24*(9), 1641–1650. https://doi.org/https://doi.org/10.1016/S0167-8655(03)00003-5

Hendrycks, D., Mazeika, M., & Dietterich, T. (2019). Deep anomaly detection with outlier exposure. https://arxiv.org/abs/1812.04606

Hermans, A., Beye, L., & Leibe, B. (2017). In Defense of the Triplet Loss for Person Re-Identification. *arXiv preprint: 1703.07737*.

Hermanson, J. C., & Wiedenhoeft, A. C. (2011). A Brief Review of Machine Vision in the Context of Automated Wood Identification Systems. *IAWA Journal*, *32*(2), 233–250.

Hido, S., Tsuboi, Y., Kashima, H., Sugiyama, M., & Kanamori, T. (2011). Statistical outlier detection using direct density ratio estimation. *Knowledge and Information Systems*, *26*, 309–336. https://doi.org/10.1007/s10115-010-0283-2

Hilal, W., Gadsden, S. A., & Yawney, J. (2022). Financial fraud: A review of anomaly detection techniques and recent advances. *Expert systems With applications*, *193*.

Hoffer, E., & Ailon, N. (2015). Deep metric learning using triplet network. *Similarity-Based Pattern Recognition*.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, *6*(2), 65–70. Retrieved January 19, 2023, from http://www.jstor.org/stable/4615733

Hossain, M. S., Hossain, M. S., Klüttermann, S., & Müller, E. (2024). Evaluating anomaly detection algorithms: A multi-metric analysis across variable class imbalances. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN60899.2024.10650351

Hua, T., Wang, W., Xue, Z., Ren, S., Wang, Y., & Zhao, H. (2021). On feature decorrelation in self-supervised learning. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9598–9608.

Hudson, D. A., & Manning, C. D. (2019). Learning by abstraction: The neural state machine. In *Proceedings of the 33rd international conference on neural information processing systems*. Curran Associates Inc.

Islam, K. (2023). Deep learning for video-based person re-identification: A survey. *arXiv preprint arXiv:2303.11332*.

Jewell, Z., Alibhai, S., Weise, F., Munro, S., Vuuren, M., & Vuuren, R. (2016). Spotting Cheetahs: Identifying Individuals by their Footprints. *Journal of Visualized Experiments, 2016*. https://doi.org/10.3791/54034

Jing, L., Vincent, P., LeCun, Y., & Tian, Y. (2022). Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint: 2110.09348*.

Joby, J. E. (2024). *Understanding the impact of automated hyperparameter tuning on anomaly detection methods* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/jessia.pdf

Jovicic, E., Primorac, D., Cupic, M., & Jovic, A. (2023). Publicly available datasets for predictive maintenance in the energy sector: A review. *IEEE Access*, *11*, 73505–73520. https://doi.org/10.1109/ACCESS.2023.3295113

Kadir, T., & Brady, M. (2001). Saliency, scale and image description. *International Journal of Computer Vision, 45*(2), 83–105.

Kaggle [https://www.kaggle.com]. (n.d.).

Kantarcı, A., Ofli, F., Imran, M., & Ekenel, H. K. (2024). Bias-aware face mask detection dataset. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-024-20226-7

Karmaker, S. (, Hassan, M. M., Smith, M. J., Xu, L., Zhai, C., & Veeramachaneni, K. (2020). Automl to date and beyond: Challenges and opportunities. *ACM Computing Surveys (CSUR), 54*, 1–36. https://api.semanticscholar.org/CorpusID:235078786

Karr, N., Nachman, B., & Shih, D. (2022). One-class dense networks for anomaly detection. *Proceedings of the Machine Learning and the Physical Sciences Workshop at NeurIPS 2022*. https://ml4physicalsciences.github.io/2022/files/NeurIPS_ML4PS_2022_130.pdf

Katz, G., Barrett, C. W., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. *Computer Aided Verification - CAV*.

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljic, A., Dill, D. L., Kochenderfer, M. J., & Barrett, C. W. (2019). The marabou framework for verification and analysis of deep neural networks. *Computer Aided Verification - 31st International Conference, CAV*.

Katzke, T. (2023). *Dean-ts: Deep ensemble anomaly detection for time series* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/tim.pdf

Khaldi, K., Mantini, P., & Shah, S. K. (2022). Unsupervised Person Re-Identification Based on Skeleton Joints Using Graph Convolutional Networks. *Image Analysis and Processing – ICIAP 2022*, 135–146.

Khan, S. S., & Madden, M. G. (2010). A survey of recent trends in one class classification. In L. Coyle & J. Freyne (Eds.), *Artificial intelligence and cognitive science* (pp. 188–197). Springer Berlin Heidelberg.

Khan, S. D., & Ullah, H. (2019). A Survey of Advances in Vision-Based Vehicle Re-Identification. *Computer Vision and Image Understanding, 182*, 50–63. https://doi.org/https://doi.org/10.1016/j.cviu.2019.03.001

Kieu, T., Yang, B., Guo, C., Jensen, C. S., Zhao, Y., Huang, F., & Zheng, K. (2022). Robust and explainable autoencoders for unsupervised time series outlier detection. *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 3038–3050. https://doi.org/10.1109/ICDE53745.2022.00273

Kingma, D., & Welling, M. (2014). Auto-encoding variational bayes.

Kirichenko, P., Izmailov, P., & Wilson, A. G. (2020). Why normalizing flows fail to detect out-of-distribution data. *Proceedings of the 34th International Conference on Neural Information Processing Systems*.

Klar, A. (1995). *Identifizierung von Ladehilfsmitteln auf der Basis Stochastischer Oberflächeninformationen* [Doctoral dissertation, TU Dortmund University]. Verlag Praxiswissen.

Klüttermann, S. (2022, April). Deepsvdd-bias [https://github.com/yzhao062/pyod/issues/385].

Klüttermann, S. (2023, January). Yano. https://doi.org/10.5281/zenodo.7520448

Klüttermann, S., Balestra, C., & Müller, E. (2024). On the efficient explanation of outlier detection ensembles through shapley values. In D.-N. Yang, X. Xie, V. S. Tseng, J. Pei, J.-W. Huang, & J. C.-W. Lin (Eds.), , *pacific-asia conference on knowledge discovery and data mining (pakdd), advances in knowledge discovery and data mining* (pp. 43–55). Springer Nature Singapore. https://doi.org/https://doi.org/10.1007/978-981-97-2259-4_4

Klüttermann, S., Gupta, S., & Müller, E. (2025). Evaluating anomaly detection algorithms: The role of hyperparameters and standardized benchmarks [Under review]. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Klüttermann, S., Katzke, T., & Müller, E. (2025). Unsupervised surrogate anomaly detection [To appear]. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*. https://arxiv.org/abs/2504.20733

Klüttermann, S., & Müller, E. (2023). Evaluating and comparing heterogeneous ensemble methods for unsupervised anomaly detection. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN54540.2023.10191405

Klüttermann, S., & Müller, E. (2024). About test-time training for outlier detection [Under review]. *2025 IEEE International Conference on Big Data (BigData)*. https://arxiv.org/abs/2404.03495

Klüttermann, S., & Müller, E. (2025). Rare anomalies require large datasets: About proving the existence of anomalies [Under review]. *International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*.

Klüttermann, S., Peka, V., Doebler, P., & Müller, E. (2024). Towards highly efficient anomaly detection for predictive maintenance [Accepted and presented, to appear]. *Proceedings of the 23rd IEEE International Conference on Machine Learning and Applications (ICMLA), Special Session 3: Machine Learning for Predictive Models in Engineering Applications (MLPMEA)*.

Klüttermann, S., Rao, N., & Müller, E. (2025). Multi-conceptual autoencoder ensembles for anomaly detection [Under review]. *International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*.

Klüttermann, S., Rutinowski, J., & Müller, E. (2024). The phenomenon of correlated representations in contrastive learning. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN60899.2024.10649913

Klüttermann, S., Rutinowski, J., Polachowski, F., Nguyen, A., Grimme, B., Roidl, M., & Müller, E. (2024). On the effectiveness of heterogeneous ensemble methods for re-identification [Accepted and presented, to appear]. *Proceedings of the 23rd IEEE International Conference on Machine Learning and Applications (ICMLA), Special Session 3: Machine Learning for Predictive Models in Engineering Applications (MLPMEA)*.

Klüttermann, S., Rutinowski, J., Reining, C., Roidl, M., & Müller, E. (2022). Towards graph representation based re-identification of chipwood pallet blocks. *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1543–1550. https://doi.org/10.1109/ICMLA55696.2022.00279

Koch, G. R. (2015). Siamese neural networks for one-shot image recognition.

Komati, H. (2024). *Enhancing anomaly detection with deep autoencoder network using a custom semi-supervised loss* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/hepsiba.pdf

Kortli, Y., Jridi, M., Al Falou, A., & Atri, M. (2020). Face Recognition Systems: A Survey. *Sensors*, *20*(2).

Koziarski, M., & Cyganek, B. (2018). Impact of low resolution on image recognition with deep neural networks: An experimental study. *International Journal of Applied Mathematics and Computer Science*, *28*(4), 735–744.

Koziel, S., Ciaurri, D. E., & Leifsson, L. (2011). Surrogate-based methods. In S. Koziel & X.-S. Yang (Eds.), *Computational optimization, methods and algorithms* (pp. 33–59). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-20859-1_3

Kozma, R., Ilin, R., & Siegelmann, H. T. (2018). Evolution of abstraction across layers in deep learning neural networks [INNS Conference on Big Data and Deep Learning]. *Procedia Computer Science*, *144*, 203–213. https://doi.org/https://doi.org/10.1016/j.procs.2018.10.520

Kriegel, H.-P., Kröger, P., Schubert, E., & Zimek, A. (2009). Outlier detection in axis-parallel subspaces of high dimensional data. In T. Theeramunkong, B. Kijsirikul, N. Cercone, & T.-B. Ho (Eds.), *Advances in knowledge discovery and data mining* (pp. 831–838). Springer Berlin Heidelberg.

Kriegel, H.-P., Schubert, M., & Zimek, A. (2008). Angle-based outlier detection in high-dimensional data. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 444–452. https://doi.org/10.1145/1401890.1401946

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images* (tech. rep.).

Kumar, V., Srivastava, V., Mahjabin, S., Pal, A., Klüttermann, S., & Müller, E. (2024). Autoencoder optimization for anomaly detection: A comparative study with shallow algorithms. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN60899.2024.10650057

Langone, R., Cuzzocrea, A., & Skantzos, N. (2020). Interpretable anomaly prediction: Predicting anomalous behavior in industry 4.0 settings via regularized logistic regression tools. *Data & Knowledge Engineering, 130*, 101850. https://doi.org/https://doi.org/10.1016/j.datak.2020.101850

Lazarevic, A., & Kumar, V. (2005). Feature bagging for outlier detection. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 21*. https://doi.org/10.1145/1081870.1081891

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

Leng, Q., Ye, M., & Tian, Q. (2019). A Survey of Open-World Person Re-Identification. *IEEE Transactions on Circuits and Systems for Video Technology, 30*(4), 1092–1108.

Levin, M. Z. A. I. (2019). Election forensics: Using machine learning and synthetic data for possible election anomaly detection. *PLoS ONE, 14*, None. https://doi.org/10.1371/journal.pone.0223950

Li, B. V., Alibhai, S., Jewell, Z., Li, D., & Zhang, H. (2018). Using Footprints to Identify and Sex Giant Pandas. *Biological Conservation*, *218*, 83–90. https://doi.org/https://doi.org/10.1016/j.biocon.2017.11.029

Li, G., Gu, Y., & Ding, J. (2021). L1 regularization in two-layer neural networks. *IEEE Signal Processing Letters*, *PP*, 1–1. https://doi.org/10.1109/LSP.2021.3129698

Li, X., Liu, B., & Ng, S.-K. (2007). Learning to identify unexpected instances in the test set. *International Joint Conference on Artificial Intelligence*. https://api.semanticscholar.org/CorpusID:14296672

Li, Z., Zhao, Y., Botta, N., Ionescu, C., & Hu, X. (2020). Copod: Copula-based outlier detection. *2020 IEEE International Conference on Data Mining (ICDM)*. https://doi.org/10.1109/ICDM50108.2020.00135

Li, Z., Zhao, Y., Hu, X., Botta, N., Ionescu, C., & Chen, G. (2022, January). *Ecod: Unsupervised outlier detection using empirical cumulative distribution functions*.

Li, Z., Zhu, Y., & Van Leeuwen, M. (2023). A survey on explainable anomaly detection. *ACM Transactions on Knowledge Discovery from Data*, *18*.

Liang, J., He, R., & Tan, T.-P. (2023). A comprehensive survey on test-time adaptation under distribution shifts. *ArXiv*, *abs/2303.15361*. https://api.semanticscholar.org/CorpusID:257767040

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. *ICDM*.

Liu, X., Yoo, C., Xing, F., Oh, H., Fakhri, G., Kang, J.-W., & Woo, J. (2022). Deep unsupervised domain adaptation: A review of recent advances and perspectives. *APSIPA Transactions on Signal and Information Processing*. https://doi.org/10.1561/116.00000192

Liu, X., Wang, H., Wu, Y., Yang, J., & Yang, M.-H. (2015). An ensemble color model for human re-identification. *IEEE Winter Conference on Applications of Computer Vision*.

Liu, X., Liu, W., Ma, H., & Fu, H. (2016). Large-Scale Vehicle Re-Identification in Urban Surveillance Videos. *ICME*.

Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. *ICCV*.

Livernoche, V., Jain, V., Hezaveh, Y., & Ravanbakhsh, S. (2024). On diffusion modeling for anomaly detection. *ICLR 2024*. https://openreview.net/forum?id=lR3rk7ysXz

Lochner, M., & Bassett, B. (2021). Astronomaly: Personalised active anomaly detection in astronomical data. *Astronomy and Computing, 36*, 100481. https://doi.org/https://doi.org/10.1016/j.ascom.2021.100481

Lokare, N., Ge, Q., Snyder, W., Jewell, Z., Allibhai, S., & Lobaton, E. (2014). Manifold Learning Approach to Curve Identification with Applications to Footprint Segmentation. *2014 IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP)*, 1–8.

Lu, L., Shin, Y., Su, Y., & Karniadakis, G. (2020). Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics, 28*(5), 1671–1706. https://doi.org/https://doi.org/10.4208/cicp.OA-2020-0165

Lu, Z., Pu, H., Wang, F., Hu, Z., & Wang, L. (2017). The expressive power of neural networks: A view from the width. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6232–6240.

Lundberg, S. M., & Lee, S.-I. (2017a). Consistent feature attribution for tree ensembles. *Proceedings of the ICML 2017 Workshop on Human Interpretability in Machine Learning (WHI)*.

Lundberg, S. M., & Lee, S.-I. (2017b). A unified approach to interpreting model predictions. *Advances in neural information processing systems, 30*.

Ma, M. Q., Zhao, Y., Zhang, X., & Akoglu, L. (2023). The need for unsupervised outlier model selection: A review and evaluation of internal evaluation strategies. *ACM SIGKDD Explorations Newsletter, 25*(1).

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations - ICLR*.

Maggipinto, M., Masiero, C., Beghi, A., & Susto, G. A. (2018). A convolutional autoencoder approach for feature extraction in virtual metrology [28th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2018), June 11-14, 2018, Columbus, OH, USAGlobal Integration of Intelligent Manufacturing and Smart Industry for Good of Humanity]. *Procedia Manufacturing*, *17*, 126–133. https://doi.org/https://doi.org/10.1016/j.promfg.2018.10.023

Malyi, V. (2017, July). Run or walk dataset. https://www.kaggle.com/datasets/vmalyi/run-or-walk

Massey, F. J. (1951). The kolmogorov-smirnov test for goodness of fit [Full publication date: Mar., 1951]. *Journal of the American Statistical Association*, *46*(253), 68–78. https://doi.org/10.2307/2280095

Mattia, F. D., Galeone, P., Simoni, M. D., & Ghelfi, E. (2021). A survey on gans for anomaly detection.

McKnight, P. E., & Najab, J. (2010). Mann-whitney u test. In *The corsini encyclopedia of psychology* (pp. 1–1). https://doi.org/https://doi.org/10.1002/9780470479216.corpsy0524

Mekki, L. (2024). *Understanding and mitigating training challenges for generative adversarial networks (gans)* [Master's thesis, TU Dortmund University].

Menges, F., Latzo, T., Vielberth, M., Sobola, S., Pöhls, H. C., Taubmann, B., Köstler, J., Puchta, A., Freiling, F., Reiser, H. P., & Pernul, G. (2021). Towards gdpr-compliant data processing in modern siem systems. *Computers & Security*, *103*, 102165. https://doi.org/https://doi.org/10.1016/j.cose.2020.102165

Mienye, D., & Sun, Y. (2022). A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, *PP*, 1–1. https://doi.org/10.1109/ACCESS.2022.3207287

Mihigo, I. N., Zennaro, M., Uwitonze, A., Rwigema, J., & Rovai, M. (2022). On-device iot-based predictive maintenance analytics model: Comparing tinylstm and tinymodel from edge impulse. *Sensors, 22*(14). https://doi.org/10.3390/s22145174

Miklautz, L., Mautz, D., Altinigneli, M. C., Böhm, C., & Plant, C. (2020). Deep embedded non-redundant clustering. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*, 5174–5181.

Mikuni, V., Nachman, B., & Shih, D. (2022). Online-compatible unsupervised nonresonant anomaly detection. *Phys. Rev. D, 105*, 055006. https://doi.org/10.1103/PhysRevD.105.055006

Miljković, D. (2011). Fault detection methods: A literature survey. *2011 Proceedings of the 34th International Convention MIPRO*, 750–755.

Ming, Z., Zhu, M., Wang, X., Zhu, J., Cheng, J., Gao, C., Yang, Y., & Wei, X. (2022). Deep Learning-Based Person Re-Identification Methods: A Survey and Outlook of Recent Works. *Image and Vision Computing, 119*, 104394.

Mitchelmore, M. C., & White, P. (2012). Abstraction in mathematics learning. In N. M. Seel (Ed.), *Encyclopedia of the sciences of learning* (pp. 31–33). Springer US. https://doi.org/10.1007/978-1-4419-1428-6_516

Modi, A., Kaur, J. N., Makar, M., Mallapragada, P., Sharma, A., Kiciman, E., & Swaminathan, A. (2023). Towards modular machine learning pipelines. *ICML Workshop on Localized Learning (LLW)*. https://openreview.net/forum?id=SiSID2Wo6j

Monteiro, W. R., & Reynoso-Meza, G. (2023). A multi-objective optimization design to generate surrogate machine learning models in explainable artificial intelligence applications. *EURO Journal on Decision Processes, 11*, 100040. https://doi.org/https://doi.org/10.1016/j.ejdp.2023.100040

Mora-Mariano, D., & Flores-Tlacuahuac, A. (2022). A machine learning approach for the surrogate modeling of uncertain distributed process engineering models.

Chemical Engineering Research and Design, 186, 433–450. https://doi.org/https://doi.org/10.1016/j.cherd.2022.07.050

Mu, H., Sun, R., Yuan, G., & Shi, G. (2021). Positive unlabeled learning-based anomaly detection in videos. *International Journal of Intelligent Systems, 36*. https://doi.org/10.1002/int.22437

Müller, E., Keller, F., Blanc, S., & Böhm, K. (2012). Outrules: A framework for outlier descriptions in multiple context spaces. *ECML PKDD*.

Müller, M., Dosovitskiy, A., Ghanem, B., & Koltun, V. (2018). Driving policy transfer via modularity and abstraction. *ArXiv, abs/1804.09364*. https://api.semanticscholar.org/CorpusID:13741394

Najari, N., Berlemont, S., Lefebvre, G., Duffner, S., & Garcia, C. (2021). Radon: Robust autoencoder for unsupervised anomaly detection. *International Conference on Security of Information and Networks - SIN*.

Nambiar, A., Bernardino, A., & Nascimento, J. C. (2019). Gait-Based Person Re-Identification: A Survey. *ACM Computing Surveys (CSUR), 52*(2), 1–34.

Nguyen, P. H. (2023). *Evaluating sequential autoencoder ensemble for anomaly detection* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/phuong.pdf

Nguyen, T. N. A. (2024). *Ensemble methods for outlier node detection on attributed static graphs* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/anh.pdf

Odegua, R. (2019, March). *An empirical study of ensemble techniques (bagging, boosting and stacking)*. https://doi.org/10.13140/RG.2.2.35180.10882

Ok, M. (2024). *Enhancing anomaly detection through test-time training* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/minjae.pdf

Ok, M., Klüttermann, S., & Müller, E. (2024). Exploring the impact of outlier variability on anomaly detection evaluation metrics. https://arxiv.org/abs/2409.15986

Olteanu, M., Rossi, F., & Yger, F. (2023). Meta-survey on outlier and anomaly detection. *Neurocomputing, 555*, 126634. https://doi.org/https://doi.org/10.1016/j.neucom.2023.126634

Oluwasegun, A., & Jung, J.-C. (2023). A multivariate gaussian mixture model for anomaly detection in transient current signature of control element drive mechanism. *Nuclear Engineering and Design, 402*, 112098.

Packaging, D. S. C. (2004). *Pallet Production Specification - Part 1: Construction Specification for 800 mm × 1200 mm Flat Wooden Pallets; German version EN 13698-1:2003* (tech. rep.). Beuth Verlag. https://doi.org/10.31030/9429856

Paisitkriangkrai, S., Shen, C., & van den Hengel, A. (2015). Learning to rank in person re-identification with metric ensembles. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Paisner, M., Cox, M. T., & Perlis, D. (2013). Symbolic anomaly detection and assessment using growing neural gas. *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, 175–181. https://doi.org/10.1109/ICTAI.2013.35

Papernot, N., McDaniel, P. D., Goodfellow, I. J., Jha, S., Celik, Z. B., & Swami, A. (2017). Practical black-box attacks against machine learning. *Proceedings of the ACM on Asia Conference on Computer and Communications Security, AsiaCCS*.

Park, C. H., & Kim, J. (2021). An explainable outlier detection method using region-partition trees. *The Journal of Supercomputing, 77*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research, 12*(Oct), 2825–2830.

Peka, V. (2024). *Anomaly detection using an ensemble with simple sub-models* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/peka.pdf

Pevný, T. (2016). Loda: Lightweight on-line detector of anomalies. *Mach. Learn.*, *102*(2). https://doi.org/10.1007/s10994-015-5521-0

Piperski, A. (2014). An Application of Graph Theory to Linguistic Complexity. *Yearbook of the Poznan Linguistic Meeting*, *1*. https://doi.org/10.1515/yplm-2015-0005

Prechelt, L. (2012). Early stopping — but when? In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade: Second edition* (pp. 53–67). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_5

Qiu, C., Li, A., Kloft, M., Rudolph, M. R., & Mandt, S. (2022). Latent outlier exposure for anomaly detection with contaminated data. *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:246867391

Qiu, C., Pfrommer, T., Kloft, M., Mandt, S., & Rudolph, M. (2022). Neural transformation learning for deep anomaly detection beyond images. https://arxiv.org/abs/2103.16440

Radaideh, M. I., Pappas, C., Wezensky, M., Ramuhalli, P., & Cousineau, S. (2022). Fault prognosis in particle accelerator power electronics using ensemble learning. *ArXiv*, *abs/2209.15570*. https://api.semanticscholar.org/CorpusID:252668772

Ramaswamy, S., Rastogi, R., & Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. *SIGMOD Rec.*, *29*(2), 427–438. https://doi.org/10.1145/335191.335437

Rao, N. (2022). *Improving autoencoder ensemble learning for unsupervised anomaly detection* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/nikitha.pdf

Rayana, S. (2016). Odds library. http://odds.cs.stonybrook.edu

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-validation. In L. LIU & M. T. ÖZSU (Eds.), *Encyclopedia of database systems* (pp. 532–538). Springer US. https://doi.org/10.1007/978-0-387-39940-9_565

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, 1530–1538.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.

Röchner, P., Marques, H. O., Campello, R. J. G. B., & Zimek, A. (2024). Evaluating outlier probabilities: Assessing sharpness, refinement, and calibration using stratified and weighted measures. *Data Mining and Knowledge Discovery*. https://doi.org/10.1007/s10618-024-01056-5

Rosenblatt, M. (1956). Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, *27*(3), 832–837. https://doi.org/10.1214/aoms/1177728190

Roßbach, D. P. (2018). Neural networks vs. random forests – does it always have to be deep learning? https://api.semanticscholar.org/CorpusID:221088499

Ruff, L., Kauffmann, J., Vandermeulen, R., Montavon, G., Samek, W., Kloft, M., Dietterich, T., & Müller, K.-R. (2021). A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE, PP*, 1–40.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., & Kloft, M. (2018). Deep one-class classification. *ICML*.

Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K., & Kloft, M. (2019). Deep semi-supervised anomaly detection. *CoRR, abs/1906.02694*. http://arxiv.org/abs/1906.02694

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. https://api.semanticscholar.org/CorpusID:62245742

Rutinowski, J., Chilla, T., Pionzewski, C., Reining, C., & ten Hompel, M. (2021a, September). *pallet-block-502 – A chipwood re-identification dataset*. Zenodo. https://doi.org/10.5281/zenodo.6353714

Rutinowski, J., Chilla, T., Pionzewski, C., Reining, C., & ten Hompel, M. (2021b). Towards Re-Identification for Warehousing Entities – A Work-in-Progress Study. *Proceedings of the IEEE Conference on Emerging Technologies In Factory Automation (ETFA)*, 501–504. https://doi.org/10.1109/ETFA45728.2021.9613250

Rutinowski, J., Endendyk, J., Reining, C., & Roidl, M. (2022, December). *galvanized-636 – A galvanized steel re- identification dataset*. Zenodo. https://doi.org/10.5281/zenodo.7386956

Rutinowski, J., Klüttermann, S., Endendyk, J., Reining, C., & Müller, E. (2024). Benchmarking trust: A metric for trustworthy machine learning. In L. Longo, S. Lapuschkin, & C. Seifert (Eds.), *Explainable artificial intelligence* (pp. 287–307). Springer Nature Switzerland. https://doi.org/https://doi.org/10.1007/978-3-031-63787-2_15

Rutinowski, J., Schrötler, N., Klüttermann, S., Roidl, M., & Janiesch, C. (2024). A laser-based volumetric measurement approach for industrial settings. *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 3595–3601. https://doi.org/10.1109/CASE59546.2024.10711535

Rutinowski, J., Vankayalapati, B., Schwenzfeier, N., Acosta, M., & Reining, C. (2022). On the applicability of synthetic data for re-identification. *arXiv preprint arXiv:2212.10105*.

Sabottke, C. F., & Spieler, B. M. (2020). The effect of image resolution on deep learning in radiography. *Radiology: Artificial Intelligence*, *2*(1).

Sadari, S. (2024). *Improving re-identification using specialized outlier detection algorithms* [Master's thesis, TU Dortmund University].

Sakurada, M., & Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, 4–11. https://doi.org/10.1145/2689746.2689747

Salem, A. (2018). Stimulation and detection of android repackaged malware with active learning. https://arxiv.org/abs/1808.01186

Sandim, M. O. (2017). *Using stacked generalization for anomaly detection* [Doctoral dissertation].

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks, 4510–4520.

Sarvari, H., Domeniconi, C., Prenkaj, B., & Stilo, G. (2021). Unsupervised boosting-based autoencoder ensembles for outlier detection. In K. Karlapalem, H. Cheng, N. Ramakrishnan, R. K. Agrawal, P. K. Reddy, J. Srivastava, & T. Chakraborty (Eds.), *Advances in knowledge discovery and data mining* (pp. 91–103). Springer International Publishing.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*(2), 197–227. https://doi.org/10.1007/BF00116037

Schapire, R. E., et al. (1999). A brief introduction to boosting. *IJCAI*.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 815–823.

Schubert, E., Wojdanowski, R., Zimek, A., & Kriegel, H. (2012). On evaluation of outlier rankings and outlier scores. *Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012*, 1047–1058. https://doi.org/10.1137/1.9781611972825.90

Sedjelmaci, H., Senouci, S. M., & Al-Bahri, M. (2016). A lightweight anomaly detection technique for low-resource iot devices: A game-theoretic methodology. *2016 IEEE*

International Conference on Communications (ICC), 1–6. https://doi.org/10.1109/ICC.2016.7510811

Serbetci, A., & Akgul, Y. S. (2020). End-to-end training of cnn ensembles for person re-identification. *Pattern Recognition*, *104*, 107319. https://doi.org/https://doi.org/10.1016/j.patcog.2020.107319

Shapley, L. S. (1953). A value for n-person games. *Contributions to the Theory of Games*.

Shen, Y., Li, H., Yi, S., Chen, D., & Wang, X. (2018). Person Re-Identification with Deep Similarity-Guided Graph Neural Network. *arXiv preprint: 1807.09975*.

Shukla, B., Fan, I.-S., & Jennions, I. (2020). Opportunities for explainable artificial intelligence in aerospace predictive maintenance. *PHM Society European Conference*, *5*(1), 11–11.

Singh, G., Gehr, T., Mirman, M., Püschel, M., & Vechev, M. T. (2018). Fast and effective robustness certification. *Advances in Neural Information Processing Systems - NeurIPS*.

Singh, S. K., & Roy, P. K. (2020). Detecting malicious dns over https traffic using machine learning. *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, 1–6. https://api.semanticscholar.org/CorpusID:231599709

Soleymani, S., Dabouei, A., Kazemi, H., Dawson, J. M., & Nasrabadi, N. M. (2018). Multi-level feature abstraction from convolutional neural networks for multimodal biometric identification. *2018 24th International Conference on Pattern Recognition (ICPR)*, 3469–3476. https://api.semanticscholar.org/CorpusID:49578081

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, *15*(1), 1929–1958.

Stocco, A., & Tonella, P. (2020). Towards anomaly detectors that learn continuously. *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 201–208. https://doi.org/10.1109/ISSREW51248.2020.00073

Strumbelj, E., & Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, *11*, 1–18.

Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, *41*(3), 647–665.

Student. (1908). The probable error of a mean. *Biometrika*, 1–25.

Sudret, B., Marelli, S., & Wiart, J. (2017). Surrogate models for uncertainty quantification: An overview. *2017 11th European Conference on Antennas and Propagation*. https://doi.org/10.23919/EuCAP.2017.7928679

Sun, H., & Guyon, I. (2023, August). Modularity in deep learning: A survey. https://doi.org/10.1007/978-3-031-37963-5_40

Sun, Y., Zheng, L., Yang, Y., Tian, Q., & Wang, S. (2018). Beyond Part Models: Person Retrieval with Refined Part Pooling (and a Strong Convolutional Baseline). *Proceedings of the European Conference on Computer Vision (ECCV)*, 480–496.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., & Hardt, M. (2020). Test-time training with self-supervision for generalization under distribution shifts. *Proceedings of the 37th International Conference on Machine Learning*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. *ICLR*.

Takahashi, T., & Ishiyama, R. (2014). FIBAR: Fingerprint Imaging by Binary Angular Reflection for Individual Identification of Metal Parts. *EST*.

Takahashi, T., Kudo, Y., & Ishiyama, R. (2017). Mass-Produced Parts Traceability System Based on Automated Scanning of "Fingerprint of Things". *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, 202–206.

Tallón-Ballesteros, A., & Chen, C. (2020). Explainable ai: Using shapley value to explain complex anomaly detection ml-based systems. *Machine learning and artificial intelligence, 332*, 152.

Tang, H. P. (2022). *Interpreting anomaly detection: Optimized preprocessing for ensemble methods* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/ping.pdf

Tang, J., Chen, Z., Fu, A. W.-c., & Cheung, D. W. (2002). Enhancing effectiveness of outlier detections for low density patterns. In M.-S. Chen, P. S. Yu, & B. Liu (Eds.), *Advances in knowledge discovery and data mining* (pp. 535–548). Springer Berlin Heidelberg.

Thambawita, V., Strümke, I., Hicks, S. A., Halvorsen, P., Parasa, S., & Riegler, M. A. (2021). Impact of image resolution on deep learning performance in endoscopy image classification: An experimental study using a large dataset of endoscopic images. *Diagnostics, 11*(12).

Thiyaagu, H. (2024). *Stacking ensemble methods for anomaly detection* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/haritha.pdf

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., & Li, L.-J. (2016). Yfcc100m: The new data in multimedia research. *Communications of the ACM, 59*(2), 64–73. https://doi.org/10.1145/2812802

Tian, K., Zhou, S., Fan, J., & Guan, J. (2019). Learning competitive and discriminative reconstructions for anomaly detection. *AAAI Conference on Artificial Intelligence*. https://api.semanticscholar.org/CorpusID:70022787

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological), 58*(1), 267–288.

Triguero, I., et al. (2017). Keel 3.0: An open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems, 10*.

van Campen, T., Hamers, H., Husslage, B., & Lindelauf, R. (2018). A new approximation method for the shapley value applied to the wtc 9/11 terrorist attack. *Social Network Analysis and Mining, 8*, 1–12.

Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2013). Openml: Networked science in machine learning. *SIGKDD Explorations*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Vaz, J. (2024). *To what extent does abstraction in physics hinder our ability to understand reality?* [Doctoral dissertation, -].

Vollert, S., Atzmueller, M., & Theissler, A. (2021). Interpretable machine learning: A brief survey from the predictive maintenance perspective. *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 01–08. https://doi.org/10.1109/ETFA45728.2021.9613467

Wang, C., Wu, Q., Weimer, M., & Zhu, E. ((2021). Flaml: A fast and lightweight automl library. *Fourth Conference on Machine Learning and Systems (MLSys 2021)*. https://www.microsoft.com/en-us/research/publication/flaml-a-fast-and-lightweight-automl-library/

Wang, J., Li, Y., & Miao, Z. (2019). Ensemble feature for person re-identification.

Wang, X., Wang, Y., Javaheri, Z., Almutairi, L., Moghadamnejad, N., & Younes, O. S. (2023). Federated deep learning for anomaly detection in the internet of things. *Computers and Electrical Engineering, 108*, 108651. https://doi.org/https://doi.org/10.1016/j.compeleceng.2023.108651

Wang, Y., & Tuerhong, G. (2022). A survey of interpretable machine learning methods. *2022 International Conference on Virtual Reality, Human-Computer Interaction and Artificial Intelligence (VRHCIAI)*, 232–237. https://doi.org/10.1109/VRHCIAI57205.2022.00047

Wang, Z., Muhammat, M., Yadikar, N., Aysa, A., & Ubul, K. (2023). Advances in offline handwritten signature recognition research: A review. *IEEE Access, 11*, 120222–120236. https://doi.org/10.1109/ACCESS.2023.3326471

Ward, M., & Rd, S. (1995). A definition of abstraction. *Journal of Software Maintenance: Research and Practice, 7*. https://doi.org/10.1002/smr.4360070606

Wei, L., Liu, X., Li, J., & Zhang, S. (2018). VP-ReID: Vehicle and Person Re-Identification System. *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, 501–504.

Weng, T., Zhang, H., Chen, H., Song, Z., Hsieh, C., Daniel, L., Boning, D. S., & Dhillon, I. S. (2018). Towards fast computation of certified robustness for relu networks. *ICML*.

Wickes, J. (2024). *Surveying the effects of lipschitz continuity on the robustness of anomaly detection algorithms* [Master's thesis, TU Dortmund University]. https://psorus.github.io/thesis/justin.pdf

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin, 1*(6), 80–83.

Wu, D., Zheng, S.-J., Zhang, X.-P., Yuan, C.-A., Cheng, F., Zhao, Y., Lin, Y.-J., Zhao, Z.-Q., Jiang, Y.-L., & Huang, D.-S. (2019). Deep Learning-Based Methods for Person Re-Identification: A Comprehensive Review. *Neurocomputing, 337*, 354–371.

Wu, T., & Wang, Y. (2021). Locally interpretable one-class anomaly detection for credit card fraud detection. *International Conference on Technologies and Applications of Artificial Intelligence - TAAI,* https://doi.org/10.1109/TAAI54685.2021.00014

Xiao, K. Y., Tjeng, V., Shafiullah, N. M. (, & Madry, A. (2019). Training for faster adversarial robustness verification via inducing relu stability. *International Conference on Learning Representations - ICLR*.

Xu, H., Pang, G., Wang, Y., & Wang, Y. (2023). Deep isolation forest for anomaly detection. *IEEE Trans. on Knowl. and Data Eng.*, *35*(12), 12591–12604. https://doi.org/10.1109/TKDE.2023.3270293

Xu, J., Li, Z., Du, B., Zhang, M., & Liu, J. (2020). *2020 IEEE Symposium on Computers and Communications (ISCC)*, 1–7. https://doi.org/10.1109/ISCC50000.2020.9219587

Xu, S., Shijia, E., & Xiang, Y. (2018). An ensemble model based on siamese neural networks for the question pairs matching task. *CCKS Tasks*.

Xu, W., Jang-Jaccard, J., Liu, T., Sabrina, F., & Kwak, J. (2022). Improved bidirectional gan-based approach for network intrusion detection using one-class classifier. *Computers*, *11*(6). https://doi.org/10.3390/computers11060085

Yamada, Y. I. S. (2019). Neural network-based anomaly detection for high-resolution x-ray spectroscopy. *Monthly Notices of the Royal Astronomical Society*, *None*, None. https://doi.org/10.1093/mnras/stz1528

Yamanaka, Y., Iwata, T., Takahashi, H., Yamada, M., & Kanai, S. (2019). Autoencoding binary classifiers for supervised anomaly detection. In A. C. Nayak & A. Sharma (Eds.), *Pricai 2019: Trends in artificial intelligence* (pp. 647–659). Springer International Publishing.

Yang, C., Zhu, Y., Lu, W., Wang, Y., Chen, Q., Gao, C., Yan, B., & Chen, Y. (2024). Survey on knowledge distillation for large language models: Methods, evaluation, and application [Just Accepted]. *ACM Trans. Intell. Syst. Technol.* https://doi.org/10.1145/3699518

Yang, Y., Liu, X., Ye, Q., & Tao, D. (2018). Ensemble learning-based person re-identification with multiple feature representations. *Complexity*, *2018*, 5940181.

Ye, F., Zheng, H., Huang, C., & Zhang, Y. (2021). Deep unsupervised image anomaly detection: An information theoretic framework. *2021 IEEE International Conference on Image Processing (ICIP)*, 1609–1613. https://doi.org/10.1109/ICIP42928.2021.9506079

Ye, H., Liu, Z., Shen, X., Cao, W., Zheng, S., Gui, X., Zhang, H., Chang, Y., & Bian, J. (2023). Uadb: Unsupervised anomaly detection booster. *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 2593–2606. https://doi.org/10.1109/ICDE55515.2023.00199

Ye, M., Lan, X., Leng, Q., & Shen, J. (2020). Cross-modality person re-identification via modality-aware collaborative ensemble learning. *IEEE Transactions on Image Processing*, *29*, 9387–9399. https://doi.org/10.1109/TIP.2020.2998275

Ye, M., Shen, J., Lin, G., Xiang, T., Shao, L., & Hoi, S. C. H. (2022). Deep Learning for Person Re-Identification: A Survey and Outlook. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *44*(06), 2872–2893. https://doi.org/10.1109/TPAMI.2021.3054775

Yee, E. (2019). Abstraction and concepts: When, how, where, what and why? *Language, Cognition and Neuroscience*, *34*(10), 1257–1265. https://doi.org/10.1080/23273798.2019.1660797

Yong, B. X., & Brintrup, A. (2022). Do autoencoders need a bottleneck for anomaly detection? *IEEE Access*, *10*, 1–1. https://doi.org/10.1109/ACCESS.2022.3192134

Yuan, S., & Wu, X. (2022). Trustworthy anomaly detection: A survey. https://arxiv.org/abs/2202.07787

Zbontar, J., Jing, L., Misra, I., LeCun, Y., & Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. *Proceedings of the International Conference on Machine Learning*, 12310–12320.

Zhang, H., & Vargas, D. (2023). A survey on reservoir computing and its interdisciplinary applications beyond traditional machine learning. *IEEE Access*, *PP*, 1–1. https://doi.org/10.1109/ACCESS.2023.3299296

Zhang, H., & Davidson, I. (2021). Towards fair deep anomaly detection. *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 138–148. https://doi.org/10.1145/3442188.3445878

Zhang, H., Cheng, M., & Hsieh, C. (2019). Enhancing certifiable robustness via a deep model ensemble. *CoRR, abs/1910.14655*.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., & Daniel, L. (2018). Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems - NeurIPS*.

Zhang, J., Xie, Y., Li, Y., Shen, C., & Xia, Y. (2020). Covid-19 screening on chest x-ray images using deep learning based anomaly detection. *ArXiv, abs/2003.12338*. https://api.semanticscholar.org/CorpusID:214693235

Zhang, J., Wang, Z., Yuan, J., & Tan, Y.-P. (2017). Positive and unlabeled learning for anomaly detection with multi-features, 854–862. https://doi.org/10.1145/3123266.3123304

Zhang, W., Yang, D., & Wang, H. (2019). Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE systems journal*, *13*(3), 2213–2227.

Zhang, Y. (2024). Causal abstraction in model interpretability: A compact survey. https://arxiv.org/abs/2410.20161

Zhao, Y., & Hryniewicki, M. K. (2018). Xgbod: Improving supervised outlier detection with unsupervised representation learning. *2018 International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/ijcnn.2018.8489605

Zhao, Y., Nasrullah, Z., & Li, Z. (2019). Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, *20*(96), 1–7. http://jmlr.org/papers/v20/19-011.html

Zhao, Y., Rossi, R., & Akoglu, L. (2021). Automatic unsupervised outlier model selection. *Advances in Neural Information Processing Systems (NeurIPS)*, *34*.

Zhao, Y., Rossi, R. A., & Akoglu, L. (2021). Automating outlier detection via meta-learning. https://arxiv.org/abs/2009.10606

Zhao, Z. (2017). *Ensemble methods for anomaly detection* [Doctoral dissertation].

Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., & Tian, Q. (2015). Scalable person re-identification: A benchmark. *Proceedings of the IEEE International Conference on Computer Vision*.

Zheng, L., Zhang, H., Sun, S., Chandraker, M., Yang, Y., & Tian, Q. (2016). Person Re-Identification in the Wild. *arXiv preprint: 1604.02531*. https://doi.org/10.48550/ARXIV.1604.02531

Zheng, S., Song, Y., Leung, T., & Goodfellow, I. (2016). Improving the robustness of deep neural networks via stability training. *IEEE Conference on Computer Vision and Pattern Recognition - CVPR*.

Zhou, C., & Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. *International Conference on Knowledge Discovery and Data Mining, KDD*.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2018). Graph Neural Networks: A Review of Methods and Applications. *arXiv preprint: 1812.08434*. https://doi.org/10.48550/ARXIV.1812.08434

Zhu, T., Ran, Y., Zhou, X., & Wen, Y. (2024). A survey of predictive maintenance: Systems, purposes and approaches. https://arxiv.org/abs/1912.07383

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE, PP*, 1–34. https://doi.org/10.1109/JPROC.2020.3004555

Zimek, A., Campello, R. J., & Sander, J. (2014). Ensembles for unsupervised outlier detection: Challenges and research questions a position paper. *SIGKDD Explor. Newsl., 15*(1), 11–22.

Zimek, A., Gaudet, M., Campello, R. J., & Sander, J. (2013). Subsampling for efficient and effective unsupervised outlier detection ensembles. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 428–436. https://doi.org/10.1145/2487575.2487676

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D.-k., & Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. *International Conference on Learning Representations*. https://api.semanticscholar.org/CorpusID:51805340