LAMARR
Institute for Machine Learning and Artificial Intelligence

UA RUHR | CENTER FOR TRUSTWORTHY DATA SCIENCE AND SECURITY RESEARCH ALLIANCE

technische universität dortmund

Abstract Ensembles for Anomaly Detection

Simon Klüttermann

PhD Defense

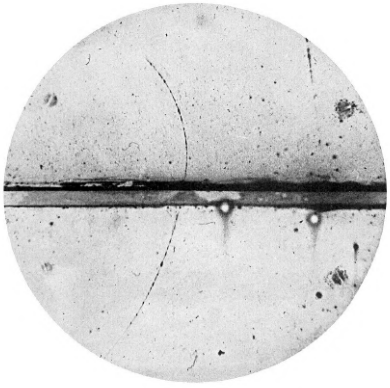
30.06.2025

Anomaly Detection

Anomaly Detection is the search for the **unusual**, **rare** or **exceptional**.

(AI generated)

Anomaly Detection



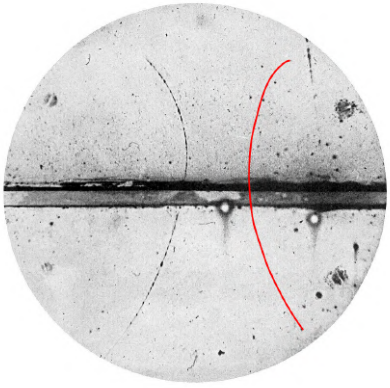
(Science Photo Library; Carl David Anderson, 1936)

1936 Photo by C. D. Anderson.

Track usually bends to the right

Led to discovery of anti-matter and a Nobel prize

Anomaly Detection



(Science Photo Library; Carl David Anderson, 1936; edited)

1936 Photo by C. D. Anderson.

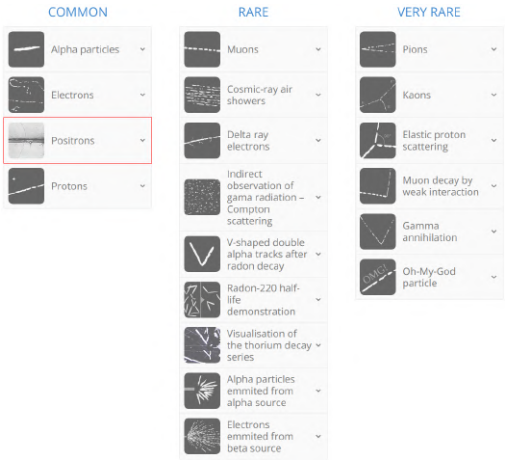
Track usually bends to the right

Led to discovery of anti-matter and a Nobel price

Anomaly Detection

Difficulty: We **don't know** how the anomaly looks like

⇒ Anomaly Detection is not easy

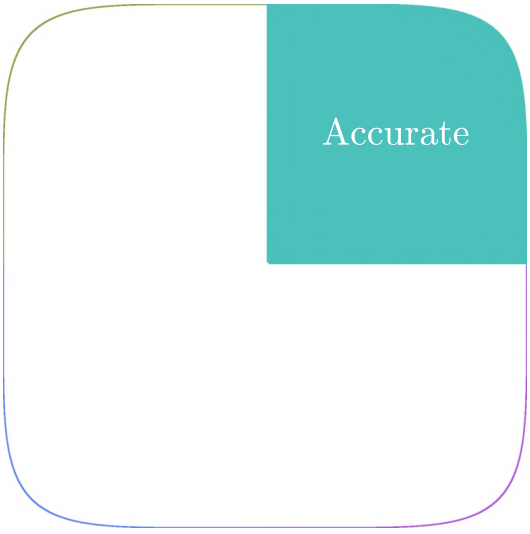


(Nuledo.com; edited)

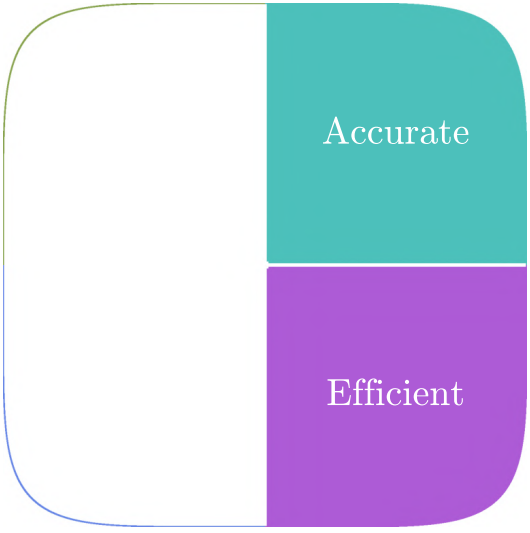
Anomaly Detection

(visitflorida.com)

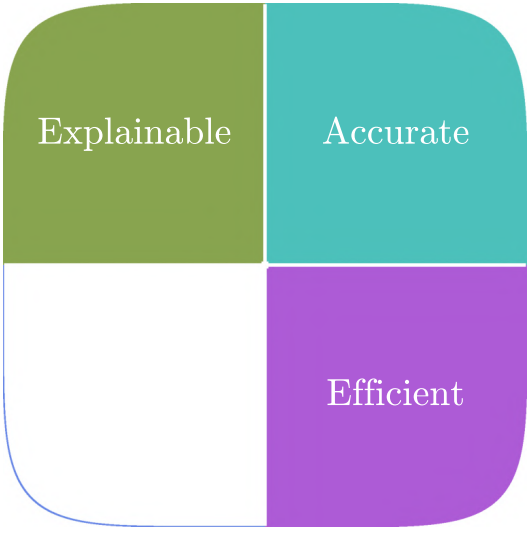
Effective Anomaly Detection



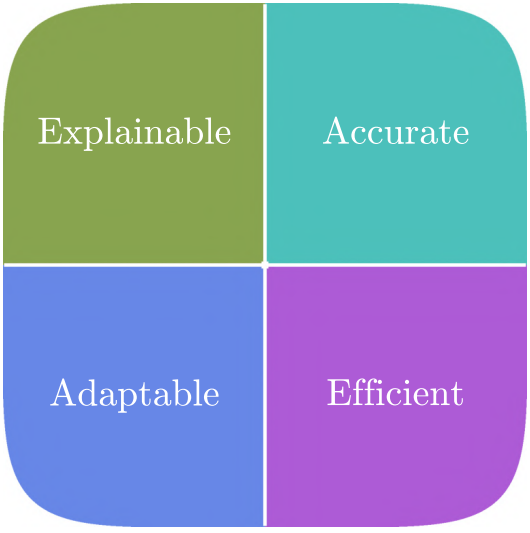
Effective Anomaly Detection



Effective Anomaly Detection



Effective Anomaly Detection



“三个臭皮匠，顶个诸葛亮”

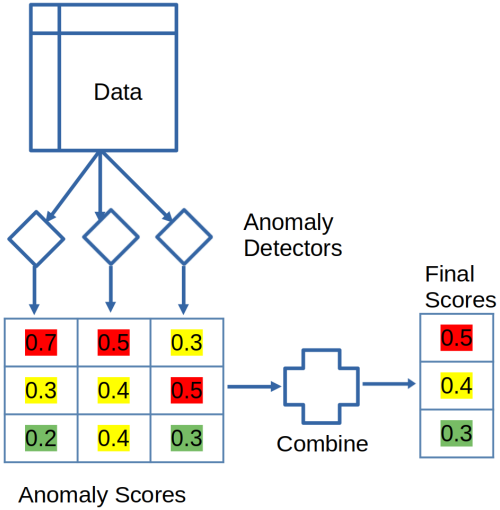
“Three stooges together are smarter than a legendary tactician alone”

(sohu.com/a/393228460_717847)

Ensembles for Anomaly Detection

Instead of one anomaly detection model, we combine multiple ones

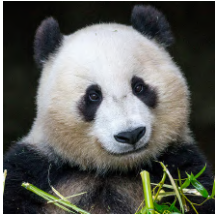
- 👍 Usually slightly more accurate
- 👎 Much slower



(Inspired by Chiang et. al. 2017; A study on anomaly detection ensembles)

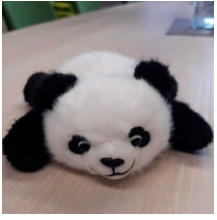
Abstraction

Abstraction is the process of **simplifying complex systems** by focusing on **high-level concepts**.



(San Diego Zoo)

(AI generated)



(Personal photo)

(Personal photo)

(Personal photo; Chengdu Metro)

Abstract Ensembles for Anomaly Detection

Instead of a **few complex** models, use **many simple**, high-level models

- 👍 Faster submodels
- 👍 Allows for more submodels
- 👍 Each submodel is more interpretable
- 👍 Can exploit the ensemble structure
- 👎 A submodel won't work on its own

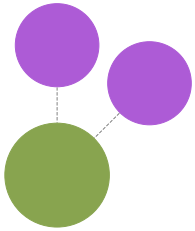
Example: Isolation Forests

(Personal photo)
(tinyurl.com/2f2dtu4k)
(tinyurl.com/eeetcxzu)

Contributions and Publications

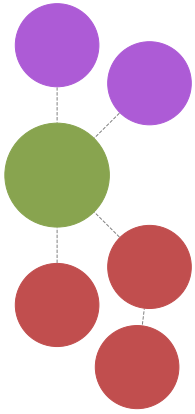
1) **Simon Klüttermann**, Tim Katzke, and Emmanuel Müller. “Unsupervised Surrogate Anomaly Detection”. ECML PKDD, 2025. (To appear)

Contributions and Publications



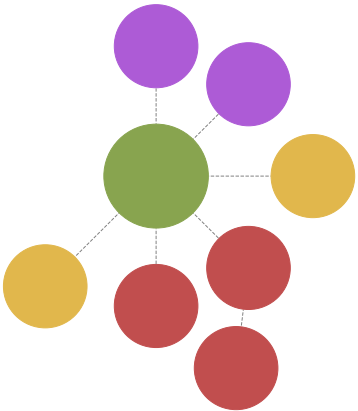
2) **Simon Klüttermann**, Chiara Balestra, and Emmanuel Müller. “On the efficient Explanation of Outlier Detection Ensembles through Shapley Values”. PAKDD, 2024.
3) Benedikt Boing*, **Simon Klüttermann***, and Emmanuel Müller. “Post-Robustifying Deep Anomaly Detection Ensembles by Model Selection”. ICDM, 2022.

Contributions and Publications



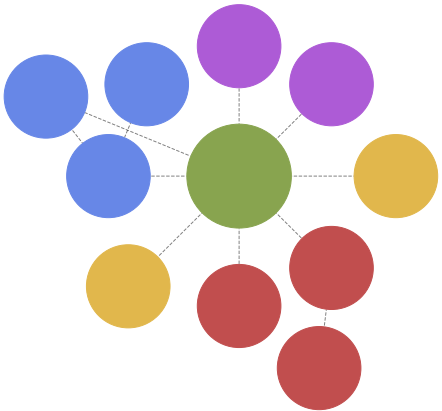
4) **Simon Klüttermann**, Vanlal Peka, Philipp Doebler, and Emmanuel Müller. “Towards Highly Efficent Anomaly Detection for Predictive Maintenance”. ICMLA, 2024.
5) **Simon Klüttermann** and Emmanuel Müller, “Near-Supervised Outlier Detection via Test-time Training”. BigData, 2025. (Under Review)
6) **Simon Klüttermann** and Emmanuel Müller. “Rare anomalies require large datasets: About proving the existence of anomalies”. ICONIP, 2025. (Under Review)

Contributions and Publications



7) **Simon Klüttermann** and Emmanuel Müller. “Evaluating and Comparing Heterogeneous Ensemble Methods for Unsupervised Anomaly Detection”. IJCNN, 2023.
8) **Simon Klüttermann**, Shubham Gupta, and Emmanuel Müller. “Evaluating Anomaly Detection Algorithms: The Role of Hyperparameters and Standardized Benchmarks”. DSAA, 2025. (Under Review)

Contributions and Publications

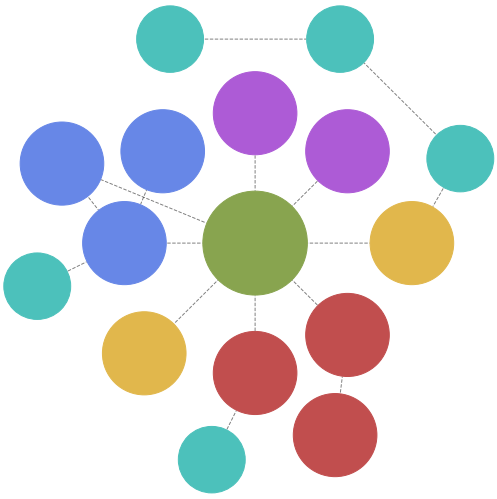


9) **Simon Klüttermann**, Jérôme Rutinowski, Christopher Reining, Moritz Roidl, and Emmanuel Müller. “Towards Graph Representation based Re-Identification of Chipwood Pallet Blocks”. ICMLA, 2022.

10) **Simon Klüttermann**, Jérôme Rutinowski, Frederik Polachowski, Anh Nguyen, Britta Grimme, Moritz Roidl, and Emmanuel Müller. “On the Effectiveness of Heterogeneous Ensemble Methods for Re-identification”. ICMLA, 2024.

11) **Simon Klüttermann**, Jérôme Rutinowski, and Emmanuel Müller. “The Phenomenon of Correlated Representations in Contrastive Learning”. IJCNN, 2024.

Contributions and Publications



12) Jérôme Rutinowski, **Simon Klüttermann**, Jan Endendyk, Christopher Reining, and Emmanuel Müller. “Benchmarking Trust: A Metric for Trustworthy Machine Learning”. XAI, 2024.

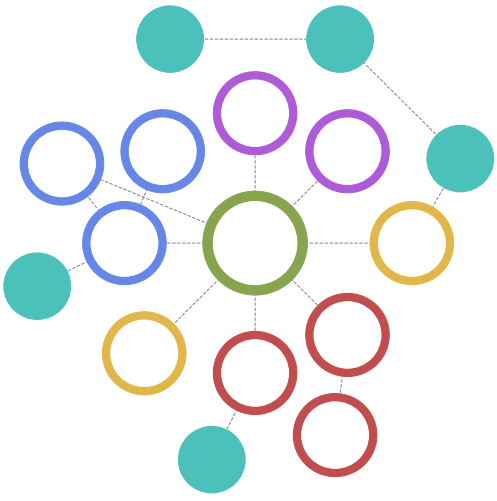
13) Minjae Ok, **Simon Klüttermann** and Emmanuel Müller. “Exploring the Impact of Outlier Variability on Anomaly Detection Evaluation Metrics”. Arxiv

14) Jérôme Rutinowski, Niklas Schrödter, **Simon Klüttermann**, Moritz Roidl, Christian Janiesch. “A Laser-based Volumetric Measurement Approach for Industrial Settings”. CASE, 2024.

15) Vikas Kumar, Vishesh Srivastava, Sadia Mahjabin, Arindam Pal, **Simon Klüttermann**, and Emmanuel Müller. “Autoencoder Optimization for Anomaly Detection: A Comparative Study with Shallow Algorithms”. IJCNN, 2024.

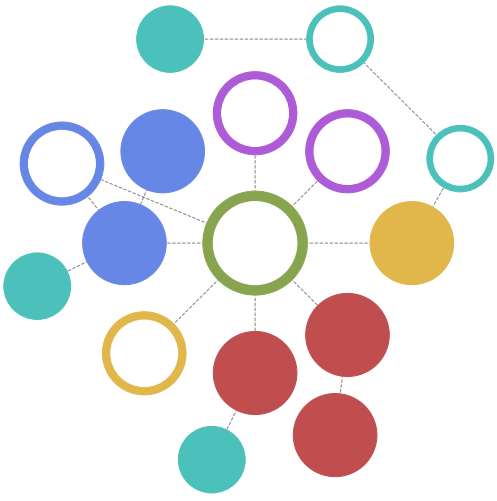
16) Mohammad Sahadat Hossain, Mohammad Sakhawat Hossain, **Simon Klüttermann**, and Emmanuel Müller. “Evaluating Anomaly Detection Algorithms: A Multi-Metric Analysis Across Variable Class Imbalances”. IJCNN, 2024.

Contributions and Publications



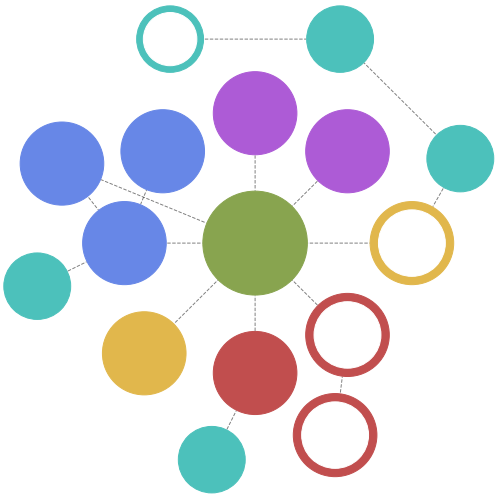
- Most papers as first author or equal contribution

Contributions and Publications



- Multiple papers in A/A* conferences

Contributions and Publications



- 4 Papers currently still under review/on arxiv

Structure of this thesis

III Benchmarking Ensemble Methods

IV DEAN: Deep Ensemble Anomaly Detection

V SEAN: Shallow Ensemble Anomaly Detection

VI Post-Robustifying DEAN by Model Selection

VII Efficient Explanations using Shapley Values

VIII Using Test Time training to improve Anomaly Detection Performance

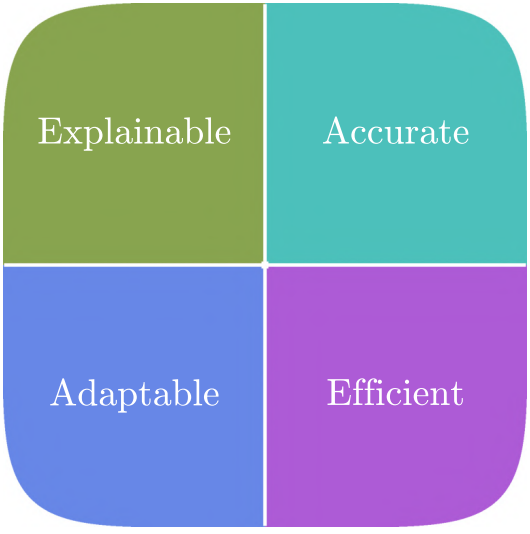
IX Understanding the limits of Test time training in Anomaly detection

X Hyperparameter-aware Benchmarking

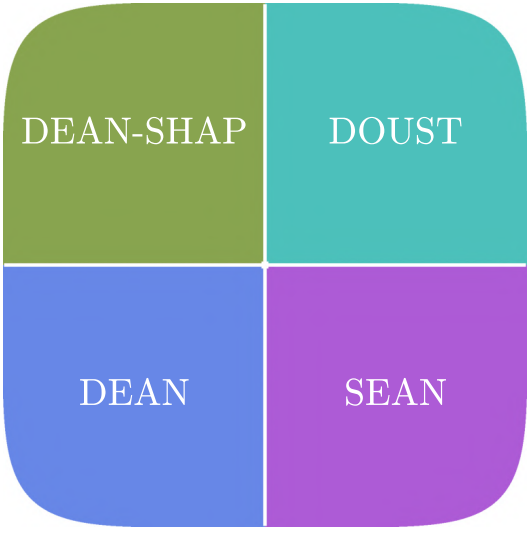
XI Abstract Re-Identificaton

XII The Problem of Correlated Representations in Contrastive Learning

Structure of this thesis



Structure of this thesis



DEAN

Deep Ensemble ANomaly detection

Simon Klüttermann, Tim Katzke, and Emmanuel Müller. “Unsupervised Surrogate Anomaly Detection”. ECML PKDD, 2025. (To appear)

DEAN — Methodology

Train **each submodel** $f(x)$ by minimizing

$$\mathcal{L}_{DEAN} = \sum_{x \in X_{train}} \|f_i(x) - 1\|, \ \forall f_i \in F$$

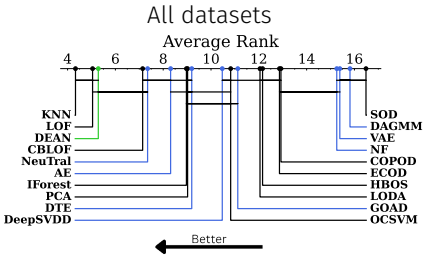
Results in MLPs with **roughly constant outputs** for normal data

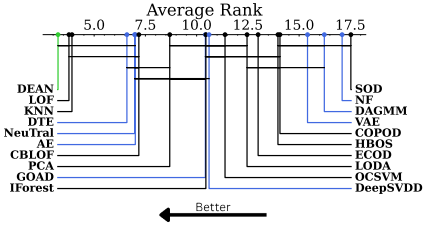
Average each submodel **error**

$$score(x) = \frac{1}{\|F\|} \sum_{f_i \in F} \|f_i(x) - q_i\|^{power}, \ q_i \approx 1$$

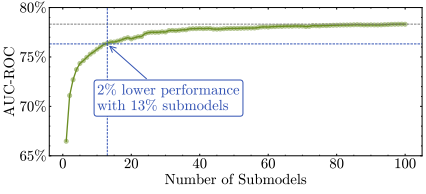
The higher this score, the more likely x is an anomaly

DEAN — Results



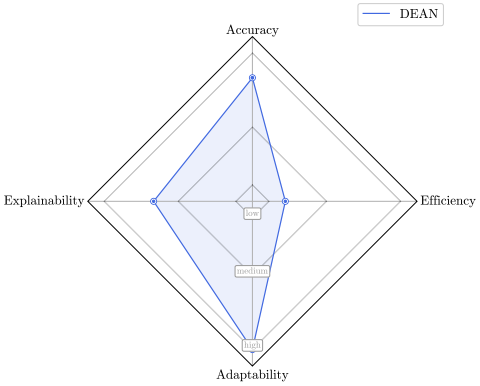


- 👍 Evaluated here on ADBench datasets + 19 competitors
- 👍 Highly competitive, and best on larger datasets
- 👍 Strong improvement with multiple submodels



DEAN — Conclusion

- 👍 Competitive performance
- 👍 Better on complex datasets
- 👍 Adaptable
- 👎 Slow



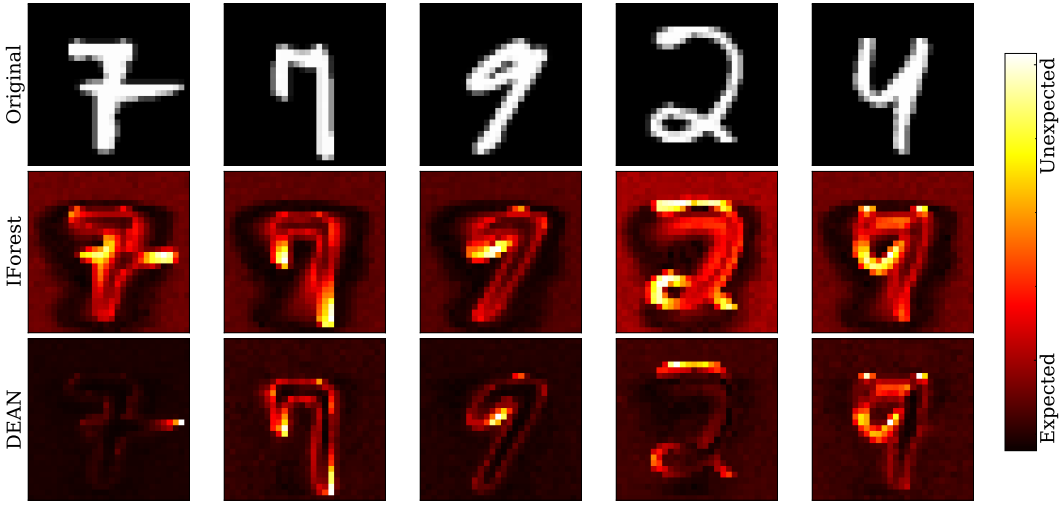
⇒ Lots more to do!

DEAN-SHAP

Simon Klüttermann, Chiara Balestra, and Emmanuel Müller. “On the Efficient Explanation of Outlier Detection Ensembles Through Shapley Values”. PAKDD 2024.

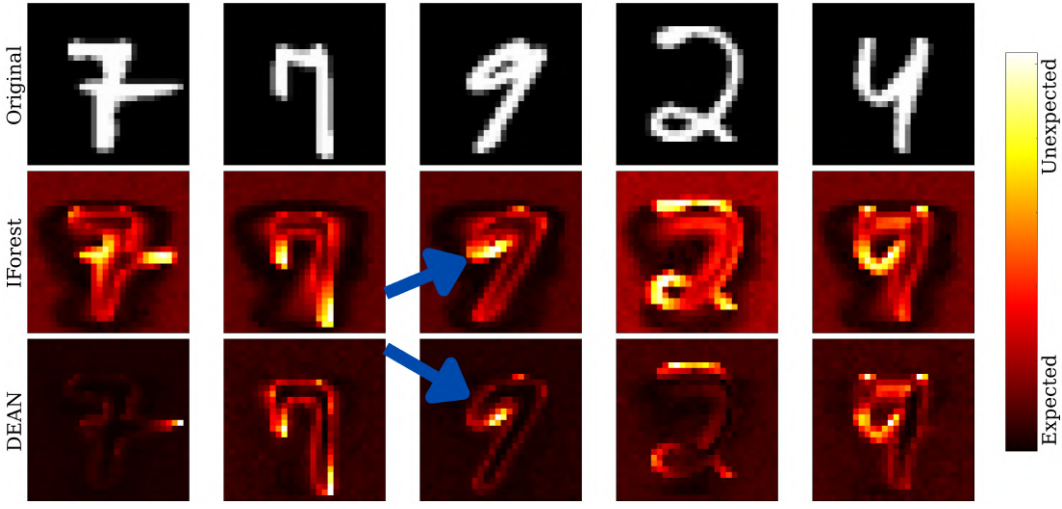
we propose a **polynomial time**
computation of Shapley values for
interpreting anomalies
for abstract ensemble methods

DEAN-SHAP



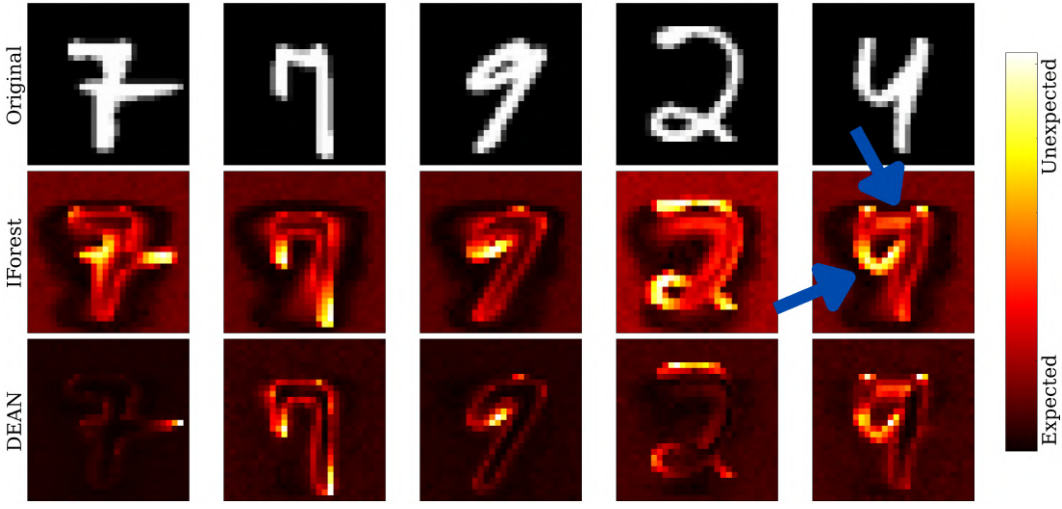
(contains MNIST images)

DEAN-SHAP



(contains MNIST images)

DEAN-SHAP



(contains MNIST images)

DEAN-SHAP

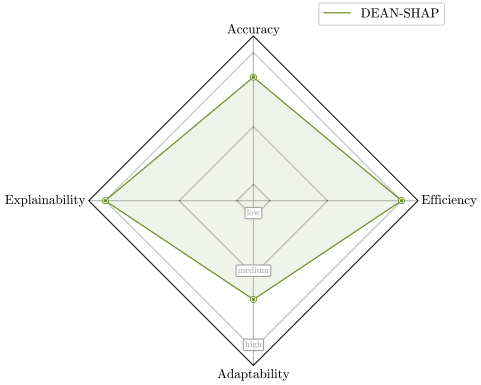
(contains CelebA images)

DEAN-SHAP

(contains CelebA images)

DEAN-SHAP — Conclusion

- 👍 Reasonable explanations
- 👍 Even on complex datasets
- 👍 Shows the limits of an anomaly detection method



SEAN

Shallow Ensemble ANomaly detection

Simon Klüttermann, Vanlal Peka, Philipp Doebler, and Emmanuel Müller. “Towards Highly Efficient Anomaly Detection for Predictive Maintenance”. ICMLA 2024.

Can we **simplify** DEAN submodels **further**?

(Personal photo)

(tinyurl.com/2f2dtu4k)

(tinyurl.com/eeetcxzu, edited)

(AI generated)

Simple MLPs ⇒ Linear regression

Instead MLPs use **feature vectors** $\boldsymbol{\alpha}$ so that

$$\min_{\boldsymbol{\alpha}_i} \sum_{X \in X_{normal,i}} (\boldsymbol{\alpha}_i^T \mathbf{x} - 1)^2$$

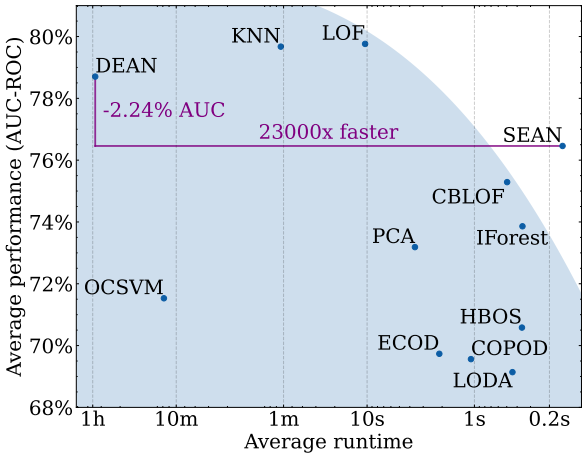
Results in roughly constant $\boldsymbol{\alpha}_i^T \mathbf{x}$ for normal data

Average each submodel **error**

$$score(x) = \frac{1}{||F||} \sum_i ||\boldsymbol{\alpha}_i^T \mathbf{x} - 1||^{power}$$

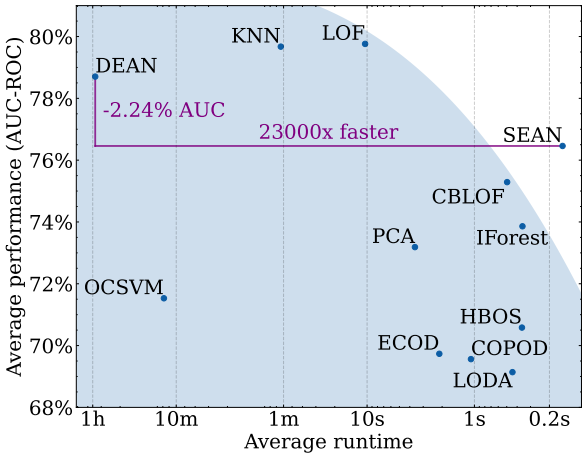
The higher this score, the more likely x is an anomaly

SEAN — Results

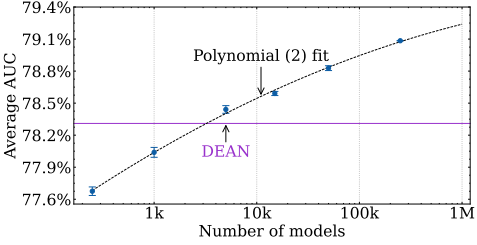


- 23000 times faster than DEAN with 50 submodels
- Slightly lower performance

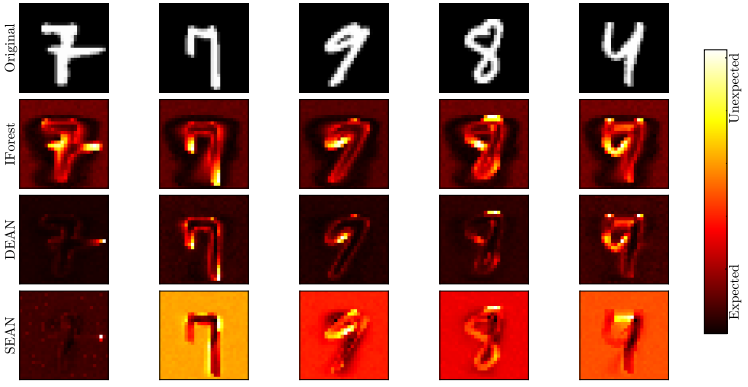
SEAN — Results



- 23000 times faster than DEAN with 50 submodels
- Slightly lower performance
- Can outperform DEAN in both performance and runtime with 10000 submodels



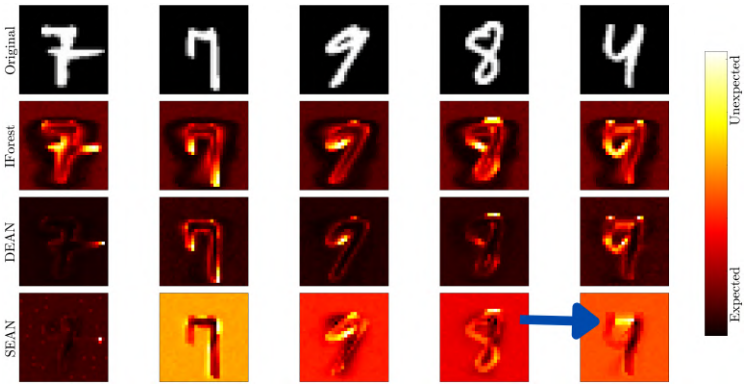
SEAN — Conclusion



(contains MNIST images)

- 👍 Much faster
- 👍 Can be more accurate
- 👍 Can run in 4 MB RAM
- 👎 Less expressive

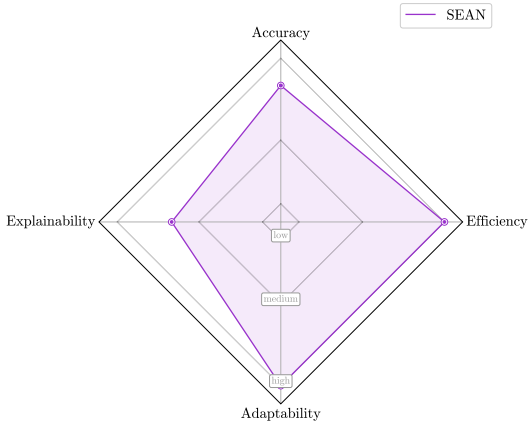
SEAN — Conclusion



(contains MNIST images)

- 👍 Much faster
- 👍 Can be more accurate
- 👍 Can run in 4 MB RAM
- 👎 Less expressive

SEAN — Conclusion



- 👍 Much faster
- 👍 Can be more accurate
- 👍 Can run in 4 MB RAM
- 👎 Slighly less expressive

DOUST

Deep OUtlier Selection using Test-time training

Simon Klüttermann, Emmanuel Müller. “About Test-time Training for Outlier Detection”. arXiv preprint, Under review at BigData 2025

Test-time training

(AI generated, edited)

- Consider a face recognition system

Test-time training

(AI generated)

- Consider a face recognition system
- What happens when something changes?
- Like for example, people wear corona masks.

Test-time training

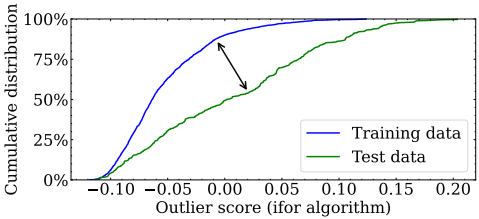
(AI generated)

- Consider a face recognition system
- What happens when something changes?
- Like for example, people wear corona masks.

Solution (by Yu Sun)

$\Rightarrow$ Use test data to adapt the system

Test-time training for Anomaly Detection



- Anomaly detection is very similar
- Training ⇔ Test-distribution
- We try to **maximize** this **difference**

⇒ DOUST algorithm

Two-step training process:

(1) Pretraining similar to DEAN

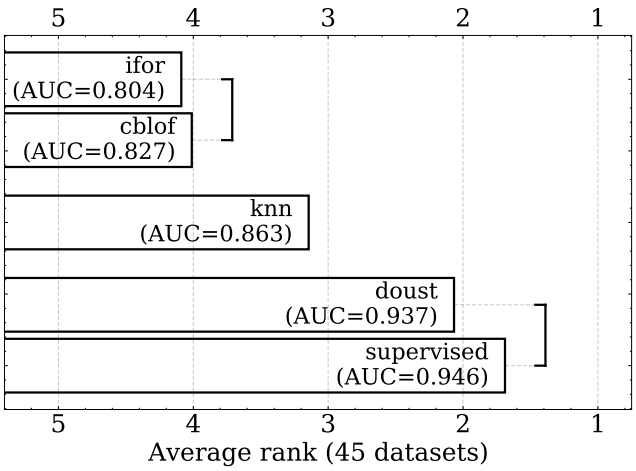
$$\mathcal{L}^1_{DOUST} = \sum_{x \in X_{train}} \left(f_i(x) - \frac{1}{2} \right)^2$$

(2) Maximize the difference between training and test

$$\mathcal{L}^2_{DOUST} = \sum_{x \in X_{train}} f_i(x)^2 + \sum_{x \in X_{test}} (1 - f_i(x))^2$$

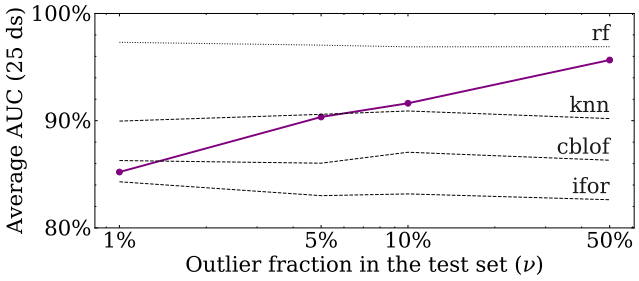
$$score(x) = \frac{1}{\|F\|} \sum_{f_i \in F} f_i(x)$$

DOUST — Results



- 👍 Outperforms the previously best algorithms
  - A supervised model is generally better than an unsupervised one
- 👍 DOUST is insignificantly worse than a supervised random forest

DOUST — Limitations



- 👎 The performance depends on ν (fraction of anomalies in the test set)
- 👎 Only seems to work when anomalies are common

DOUST — Limitations

$$N \gg \frac{1}{\nu^2}$$

N: Number of samples

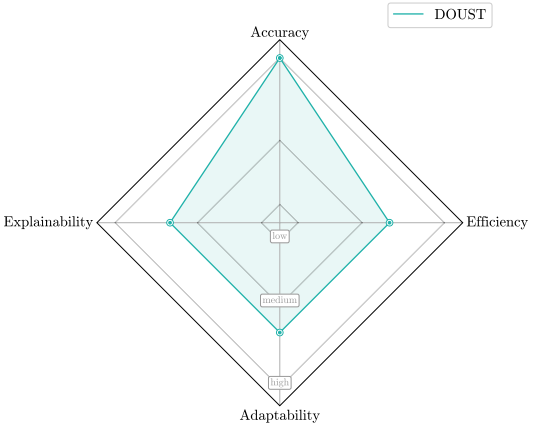
ν: Fraction of Anomalies

- 👎 The performance depends on ν (fraction of anomalies in the test set)
- 👎 Only seems to work when anomalies are common

- 👍 We found a limiting equation
- 👍 Anomalies may be rare when the dataset is large enough

DOUST — Conclusion

- 👍 Very good performance
- 👎 Slow
- 👎 Unreliable
- ⇒ Further research needed



DOUST — Conclusion

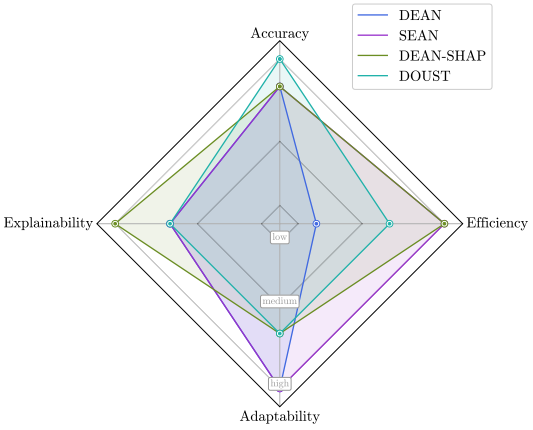
- 👍 Very good performance
- 👎 Slow
- 👎 Not reliable
- ⇒ Further research needed

Some of which has been done by thesis students

- Gautam Hariharan: *Anomaly Detection Model Selection Using Super-AUC Scores*
- Vikas Kumar: *Test-time training for anomaly detection in Time Series*
- Minjae Ok: *Enhancing Anomaly Detection through Test-Time Training*

Conclusion

- 👍 DEAN competitive and highly **adaptable**
- 👍 SEAN 23000 times **faster** with similar performance
- 👍 DEANSHAP allows **explaining** anomalies
- 👍 DOUST drastically more **accurate**, but unreliable



Future Work — Polyra

Can we **simplify** our submodels even **further**?

(Personal photo)

(tinyurl.com/2f2dtu4k)

(tinyurl.com/eeetcxzu, edited)

(AI generated)

(AI generated)

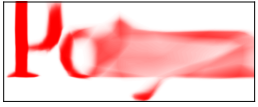
$\Rightarrow$ Polyra

Future Work — Polyra

With a NN

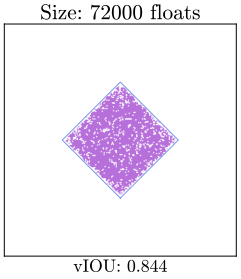
+reject option

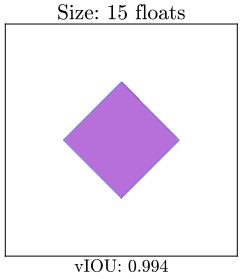
With Polyra



With a NN

With Polyra





(Under Review at NeurIPS)

Future Work — Statistical detection

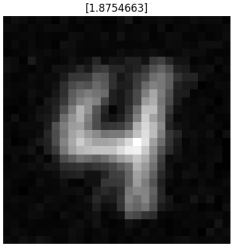
- We test whether we can **detect** the **existence** of anomalies using **statistical tests**
- Using the KS-Test we rediscover **DOUST's limit**ing equation

$$N \gg \frac{1}{\nu^2}$$

$$\Rightarrow$$

$$N > \frac{1}{\nu^2} \cdot \frac{1}{\gamma^2} \cdot c(\sigma)^2 \cdot (1 + \frac{N}{N_{max}})$$

Future Work



(Ben Ernst)

- Anomaly Detection left behind other disciplines
- Incapable of detecting arbitrarily complex signals

Thus, we need

$\Rightarrow$ More reliable evaluation approaches

$\Rightarrow$ Benchmark datasets for complex tasks

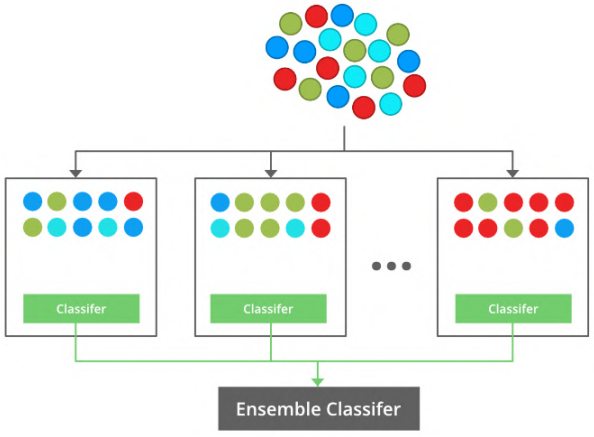
$\Rightarrow$ More modular anomaly detection setups

Thank you

Co-authors

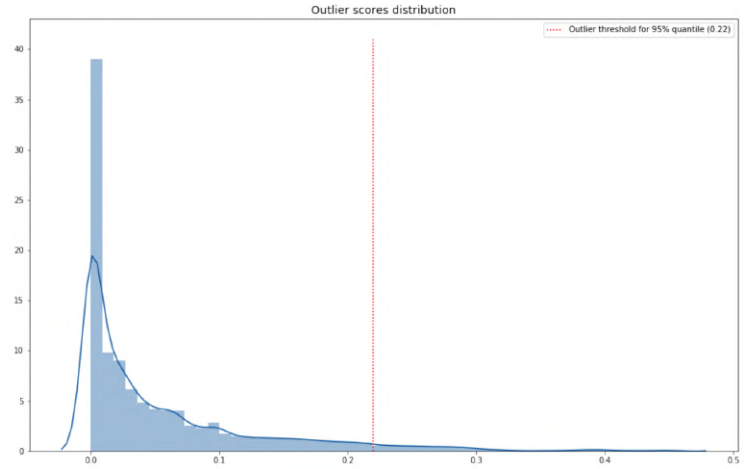
Students

Feature Bagging



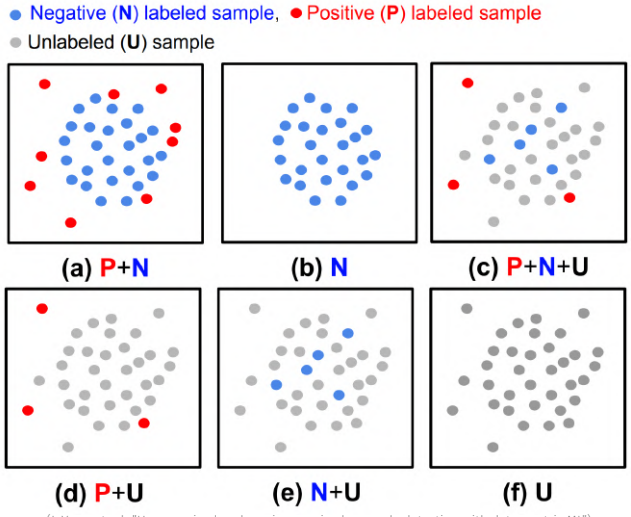
(www.geeksforgeeks.org/machine-learning/bagging-vs-boosting-in-machine-learning)

Thresholds and Anomaly Scores



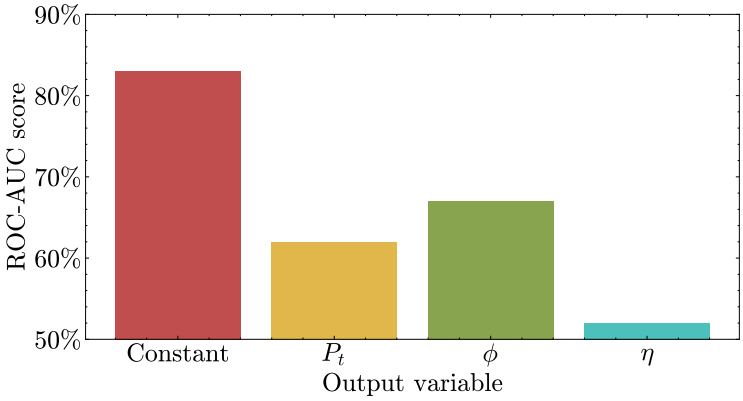
(A. Llamas et. al. "Flight Data Monitoring (FDM) Unknown Hazards detection during Approach Phase using Clustering Techniques and AutoEncoders")

Supervision



(J. Yoon et. al. "Unsupervised and semi-supervised anomaly detection with data-centric ML")

DEAN — Original



Autoencoder loss: $\mathcal{L}_{AE} = \sum_{feature} \|Input - Output\|^2$

1) 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 $\|g(x_1)\| \approx \|g(x_2)\|$.

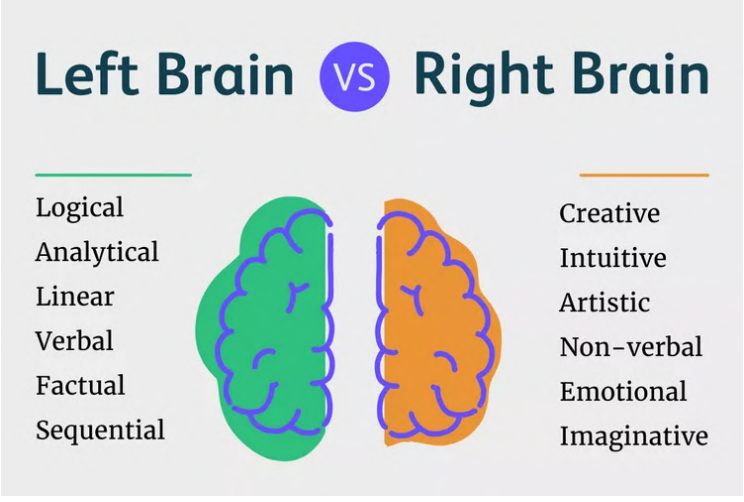
2) When learning to approximate g multiple times under identical training conditions, the variance in performance should be small. For learned instances $f_1, ..., f_n$ it holds that $Var(m(f_i)) \leq \delta^2$, where the constant $\delta > 0$ is as as small as possible.

3) There should be no trivial solution f_{trivial} such that $\nabla\mathcal{L}(f_{\text{trivial}}) = 0$ and $f_{\text{trivial}}(x) \approx c$ for all $x \in \mathbb{R}^d$ and some constant $c \in \mathbb{R}^k$.

4) For any two reasonable hyperparameter sets H_A and H_B, let f_{H_A} and f_{H_B} be the corresponding learned models. Then the performance difference should be bounded as $|m(f_{H_A}) - m(f_{H_B})| \leq \eta$, where $\eta > 0$ is chosen to be as small as possible.

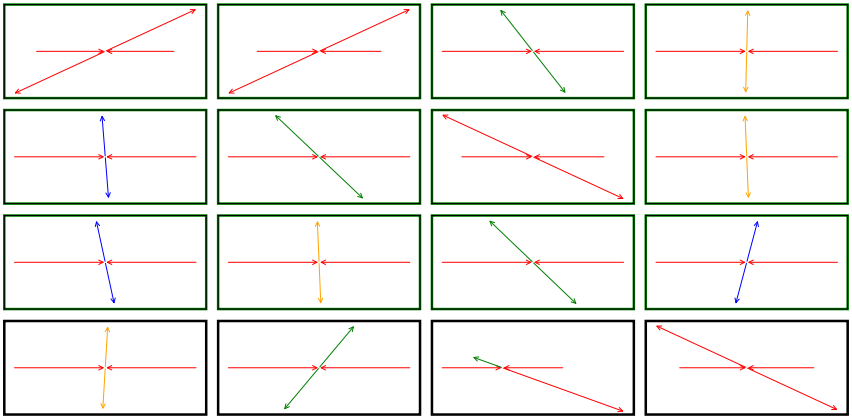
5) The learnable function f needs to be represented by a universal function approximator, capable of approximating any continuous function $g : \mathbb{R}^d \to \mathbb{R}^k$ to arbitrary precision on subsets of $\mathbb{R}^d$.

Left Brain <> Right Brain



(Michela Buttignol for Verywell Mind)

Collisions



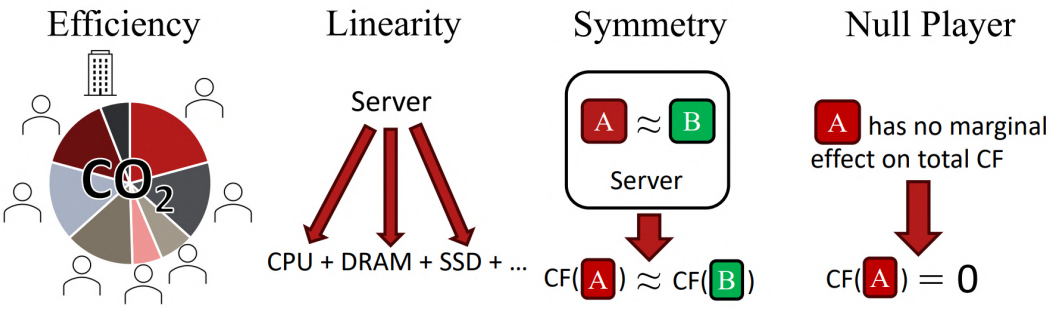
Shapley Values

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} \left[v(S \cup \{i\}) - v(S) \right]$$

- Proposed by Lloyd S. Shapley in 1953
- Calculates the **contribution of features** in a fair and theoretically grounded way

👎 Requires **exponentially many** models to be trained

Shapley Values — Attributes



(hotcarbon.org/assets/2024/pdf/slides-7.pdf

If $v(S)$ is an ensemble with feature bagging

$\Rightarrow$ we can calculate every step at the same time

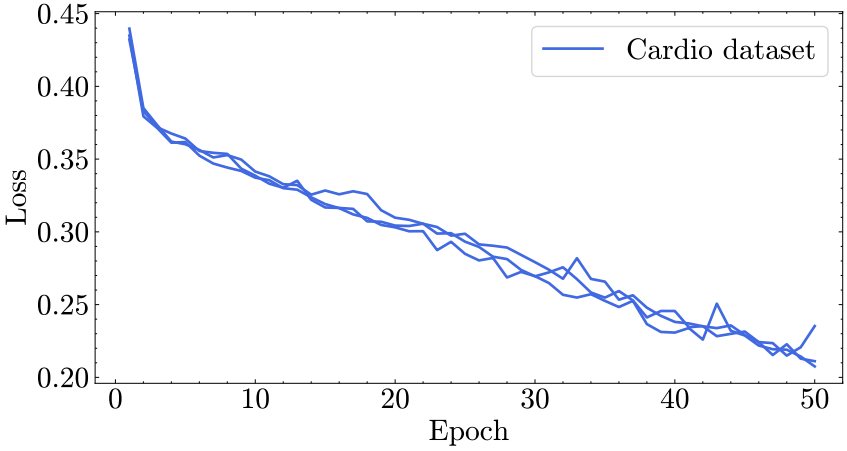
$\Rightarrow$ results in polynomial time cost

Let the model be
$v(S)(x) = \sum_{f_i \in F} score_{f_i}(x)$

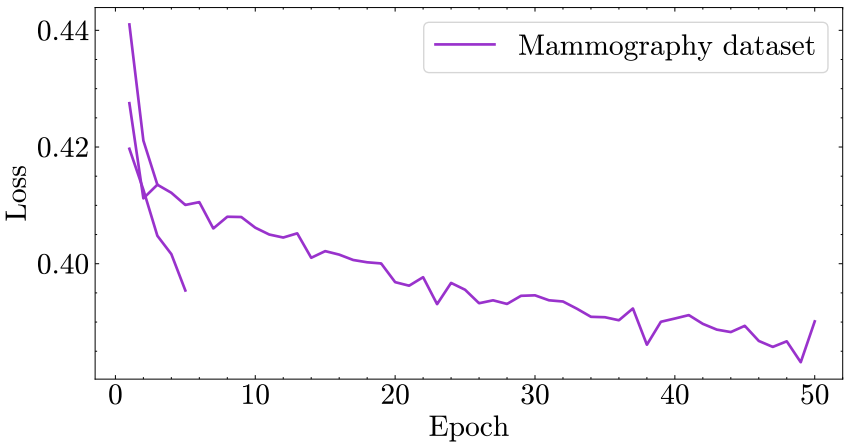
Then we calculate two sums
$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \sum_{f_i \in F} \cdots$

whose order we can reverse
$= \sum_{f_i \in F} \sum_{S \subseteq N \setminus \{i\}} \cdots$

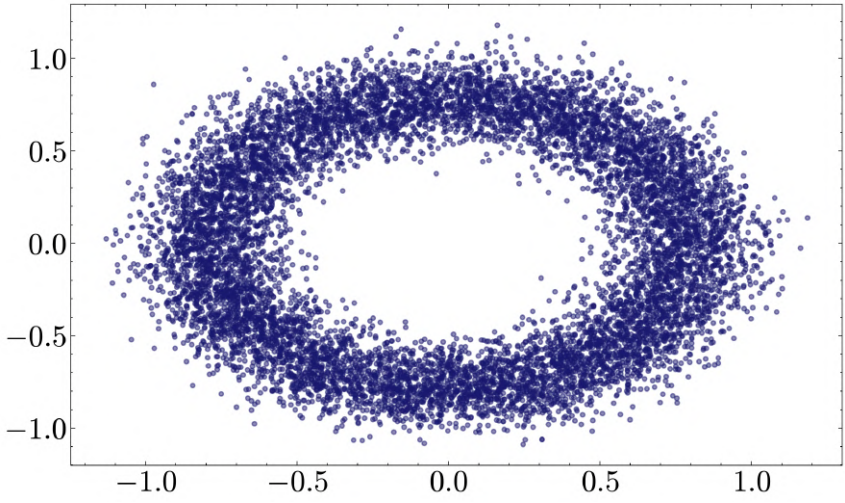
Training Curves — DOUST



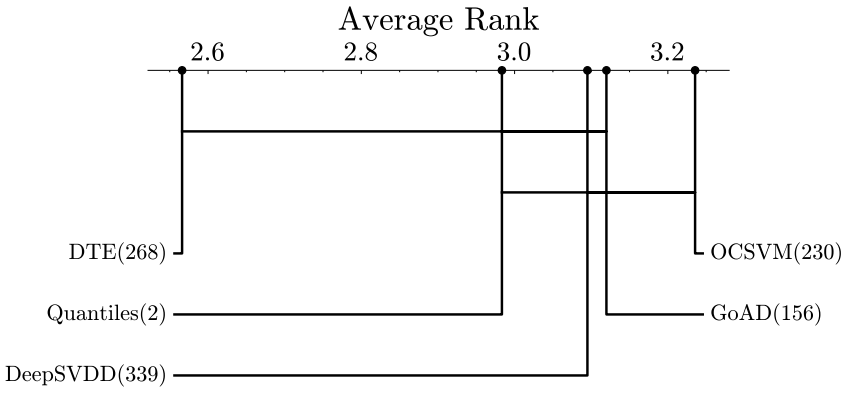
Training Curves — DOUST



Donut dataset



Quantiles Algorithm



III Benchmarking Ensemble Methods

Methods	Mean AUC	Bigger Than			ΔAUC to			ΔErr compared to		
		Minimum	*Mean*	*Maximum*	*Minimum*	*Mean*	*Maximum*	*Minimum*	*Mean*	*Maximum*
maximum (01)	**0.766**	**99.837%**	75.39%	4.445%	**0.28**	**0.036**	**-0.081**	**1.741**	**0.955**	**-1.386**
maximum (z)	0.758	99.248%	72.99%	3.258%	0.271	0.027	−0.09	1.601	0.814	−1.527
maximum (simple)	0.723	96.634%	64.153%	4.05%	0.237	−0.007	−0.124	1.37	0.584	−1.757
maximum (clipped 1)	0.66	91.431%	23.552%	0.282%	0.174	−0.07	−0.188	0.487	−0.299	−2.64
maximum (clipped 2)	0.712	95.002%	53.596%	0.739%	0.226	−0.018	−0.136	0.814	0.027	−2.314
mean (01)	0.754	96.794%	84.015%	2.64%	0.267	0.023	−0.094	1.599	0.813	−1.528
mean (z)	0.756	96.796%	84.857%	2.633%	0.27	0.026	−0.092	1.604	0.817	−1.524
mean (simple)	0.734	96.801%	69.156%	3.767%	0.248	0.004	−0.114	1.505	0.718	−1.623
mean (clipped 1)	0.747	96.8%	82.687%	1.308%	0.26	0.016	−0.101	1.199	0.412	−1.928
mean (clipped 2)	0.754	96.794%	**85.903%**	1.576%	0.267	0.023	−0.094	1.321	0.534	−1.807
median (01)	0.752	96.708%	82.115%	1.742%	0.265	0.021	−0.096	1.401	0.614	−1.727
median (z)	0.755	96.764%	84.038%	1.803%	0.268	0.024	−0.093	1.447	0.661	−1.68
median (simple)	0.744	96.659%	74.613%	2.805%	0.258	0.014	−0.104	1.381	0.595	−1.746
median (clipped 1)	0.743	96.761%	78.658%	0.475%	0.257	0.013	−0.104	1.056	0.269	−2.072
median (clipped 2)	0.753	96.759%	83.664%	1.065%	0.267	0.023	−0.095	1.295	0.509	−1.832
minimum (01)	0.673	95.942%	36.83%	0.902%	0.187	−0.057	−0.174	0.687	−0.1	−2.441
minimum (z)	0.671	96.439%	35.013%	0.682%	0.185	−0.059	−0.177	0.66	−0.126	−2.467
minimum (simple)	0.742	97.023%	71.487%	4.143%	0.255	0.011	−0.106	1.203	0.417	−1.924
minimum (clipped 1)	0.667	96.466%	29.391%	0.643%	0.181	−0.063	−0.181	0.629	−0.158	−2.499
minimum (clipped 2)	0.672	96.431%	34.76%	0.671%	0.186	−0.058	−0.175	0.66	−0.126	−2.467
l2mean (01)	0.753	97.075%	77.929%	3.079%	0.267	0.023	−0.095	1.649	0.862	−1.479
l2mean (z)	0.729	95.947%	52.75%	4.131%	0.243	−0.001	−0.119	1.311	0.524	−1.817
l2mean (simple)	0.735	97.753%	65.479%	2.892%	0.249	0.005	−0.113	1.458	0.671	−1.67
l2mean (clipped 1)	0.703	93.547%	37.37%	2.09%	0.216	−0.028	−0.145	0.803	0.016	−2.325
l2mean (clipped 2)	0.731	94.967%	53.955%	2.933%	0.244	0.0	−0.117	1.087	0.3	−2.041
l3mean (01)	0.751	97.333%	76.353%	3.242%	0.265	0.021	−0.097	1.637	0.851	−1.49
l3mean (z)	0.728	96.185%	52.526%	4.219%	0.242	−0.002	−0.12	1.314	0.528	−1.813
l3mean (simple)	0.734	97.618%	65.393%	3.095%	0.247	0.003	−0.114	1.444	0.657	−1.684
l3mean (clipped 1)	0.706	93.834%	39.472%	2.095%	0.22	−0.024	−0.142	0.827	0.04	−2.301
l3mean (clipped 2)	0.734	95.336%	56.223%	3.013%	0.247	0.003	−0.114	1.115	0.329	−2.012
l4mean (01)	0.75	97.537%	75.127%	3.361%	0.263	0.019	−0.098	1.604	0.817	−1.523
l4mean (z)	0.728	96.396%	52.426%	4.203%	0.241	−0.003	−0.12	1.316	0.53	−1.811
l4mean (simple)	0.733	97.599%	65.068%	3.15%	0.247	0.003	−0.114	1.436	0.649	−1.692
l4mean (clipped 1)	0.708	94.011%	40.809%	2.117%	0.222	−0.022	−0.14	0.844	0.058	−2.283
l4mean (clipped 2)	0.735	95.912%	57.487%	3.016%	0.249	0.005	−0.112	1.132	0.346	−1.995
Threshold ($t = -1$)	0.757	96.8%	85.036%	2.655%	0.271	0.027	−0.091	1.608	0.821	−1.52
Threshold ($t = -2$)	0.756	96.796%	84.934%	2.652%	0.27	0.026	−0.091	1.604	0.817	−1.524
Threshold ($t = 0$)	0.756	96.734%	81.793%	3.069%	0.27	0.026	−0.091	1.599	0.813	−1.528
Threshold ($t = 1$)	0.746	96.797%	70.396%	2.296%	0.26	0.016	−0.101	1.5	0.718	−1.628
Threshold ($t = 2$)	0.719	97.651%	47.855%	3.156%	0.233	−0.011	−0.129	1.295	0.508	−1.833
knn ($k = 1$)	0.757	97.326%	68.474%	4.434%	0.271	0.027	−0.091	1.543	0.757	−1.584
knn ($k = 3$)	0.757	97.532%	68.761%	4.434%	0.27	0.026	−0.091	1.537	0.751	−1.59
knn ($k = 5$)	0.756	97.664%	69.132%	4.074%	0.27	0.026	−0.092	1.523	0.736	−1.605
knn ($k = 10$)	0.752	97.068%	68.905%	3.22%	0.266	0.022	−0.096	1.48	0.693	−1.648
knn ($k = 100$)	0.664	78.755%	44.347%	3.768%	0.178	−0.066	−0.184	1.044	0.258	−2.083
Pcw	0.754	97.283%	69.757%	2.502%	0.268	0.024	−0.093	1.352	0.566	−1.775
Gaussian	0.754	97.807%	66.289%	**6.148%**	0.268	0.024	−0.094	1.397	0.611	−1.73
Clustered	0.752	96.74%	76.265%	3.991%	0.265	0.021	−0.096	1.694	0.907	−1.434
Greedy	0.751	96.73%	83.008%	2.167%	0.264	0.02	−0.097	1.339	0.552	−1.789
IRT	0.731	96.511%	70.052%	2.624%	0.245	0.001	−0.117	1.125	0.338	−2.003

Simpler methods are often better

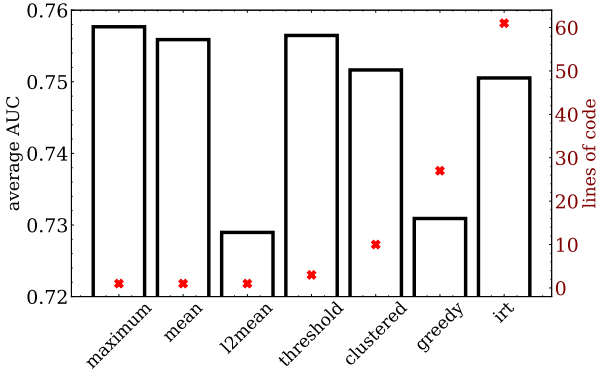
Some ensembles submodels usually outperform the ensemble

Which methods to use when

Which models to choose

Stacking for unsupervised ensembles

III Benchmarking Ensemble Methods



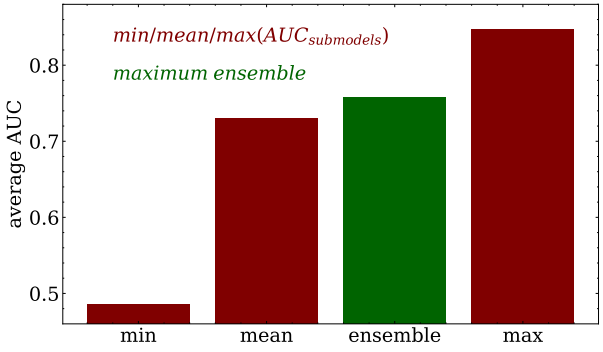
Many different ensemble functions suggested

No visible benefit through complexity

Best two ensemble functions are *maximum* and *mean*

→ Not a solved problem

III Benchmarking Ensemble Methods



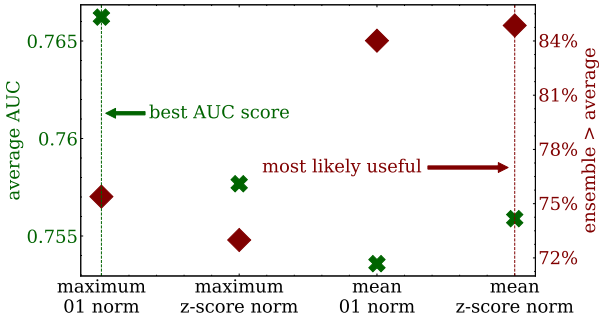
Unsupervised ensembles do not perform better than each submodel (4%)

But usually perform better than random submodel (75%)

Safely performs better than worst submodel (99.8%)

→ Unsupervised ensembles work differently than supervised ones

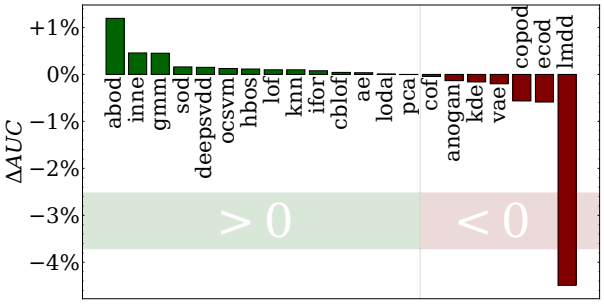
III Benchmarking Ensemble Methods



maximum on average the best

mean most often the best

$\rightarrow$ mean more reliable, maximum more effective



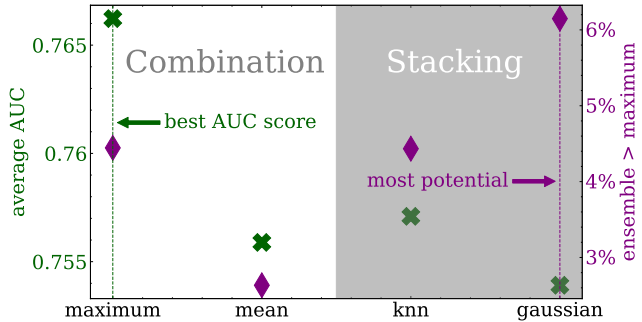
Not every algorithm works equally well in an ensemble

Similar algorithms ↔ similar improvement

Unique algorithm → high improvement

→ Algorithm performance less important than creativity

III Benchmarking Ensemble Methods



Stacking: Train a model on the output of other models

Not well studied in unsupervised AD

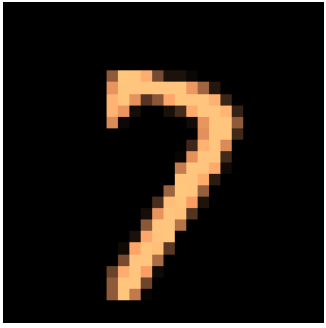
Cant outperform simple ensembles on average

But can most often improve each submodel

$\rightarrow$ Future work

VI Outlier Detection is susceptible to Adversarial Attacks

Original Sample



normal

Ensemble Pseudo-Adversarial

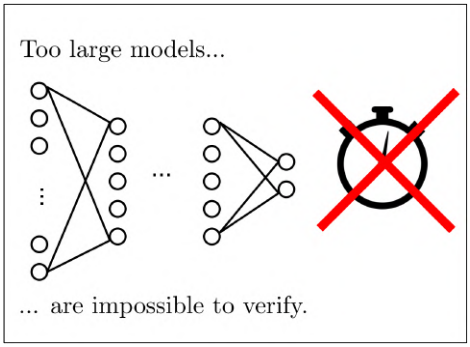
anomalous

- adversarial are designed to fool neural networks
- here: false positive anomalies
- robust $\Leftrightarrow$ no adversarial

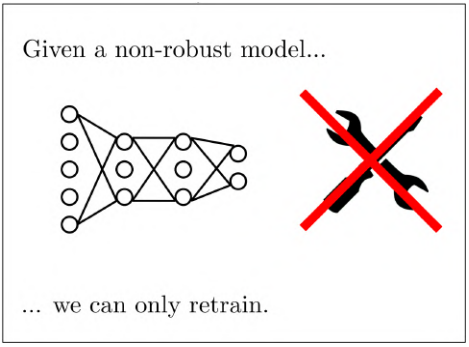
$\Rightarrow$ verify through SMT solvers

VI Currently, Verification has some Caveats

Exponential Runtime

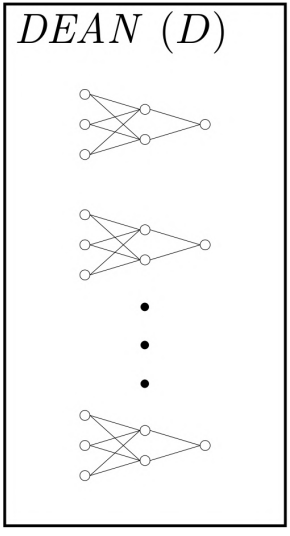


Limited Use



How to solve these problems?

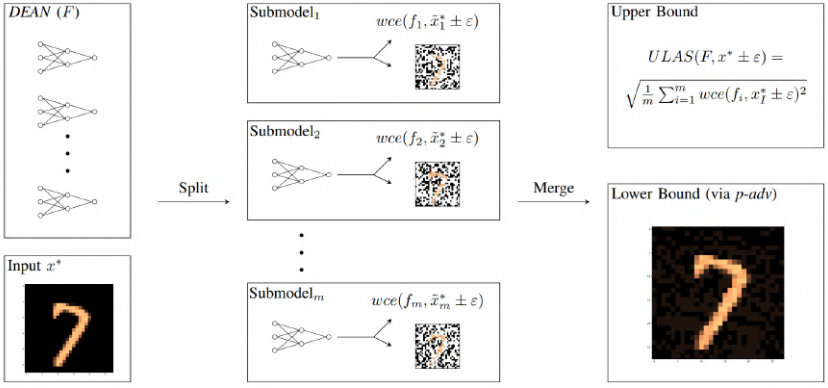
VI Ensembles can help solve this



- Not one complicated model but an **ensemble** of **simple models**
- constant model size through **feature bagging**
- Loss function: $L_i^{dean}(x) = (f_i(x) - q)^2$, $q = 1$
- Anomalousness: $Anom(x) = \sqrt{\frac{1}{N}\sum_i^N L_i^{dean}(x)}$
- $x = \begin{cases} \text{normal, if } Anom(x) < \tau \\ \text{abnormal else} \end{cases}$

VI Verification by Divide and Conquer

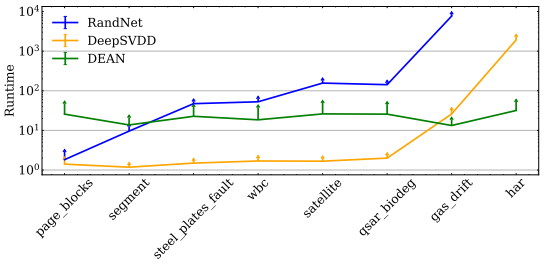
$wce(f_i, A) = \sup_{x \in A} \|q - f_i(x)\|_\infty$ Solve through SMT Solver



→ *Very good approximation,* ≈ 1% *uncertainty*

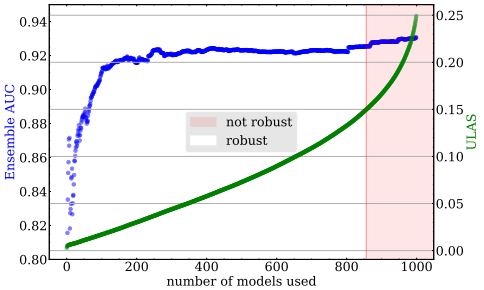
VI We Post-Robustify by leaving out Models

With larger dimension...



...verification time stays constant

Leaving our models with large *wce*...

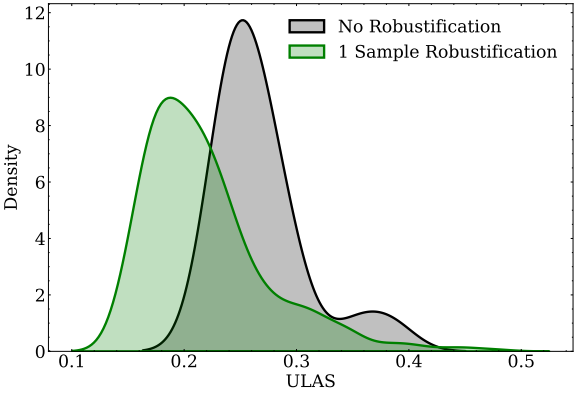


- makes the model robust
- keeps the predictive power

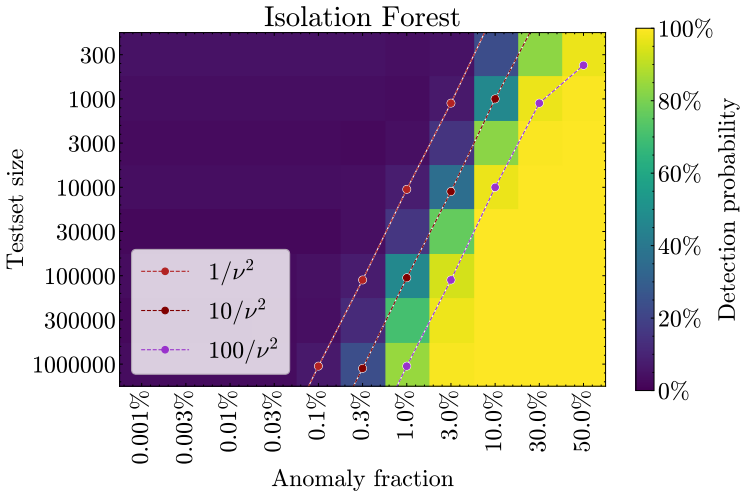
We overcome both challenges by network design!

VI Future Work: This can work for more than one sample

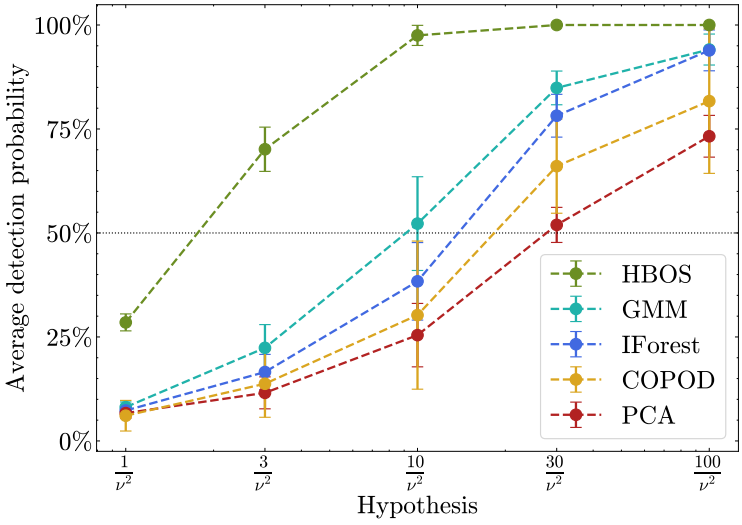
- Verification only locally possible
- Limits practical usability
- Here 1 sample robust models are also more robust on other samples
- Allows verifying multiple samples ($\approx$ 20 with $\approx$ 50% models remaining)



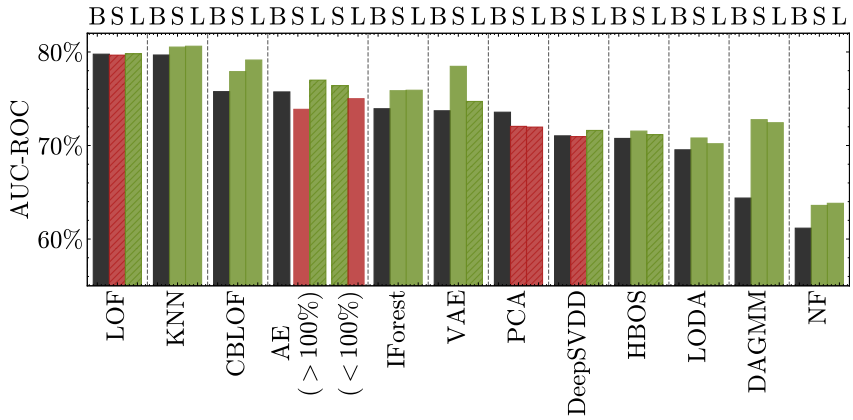
IX Understanding the limits of Test time training in Anomaly detection



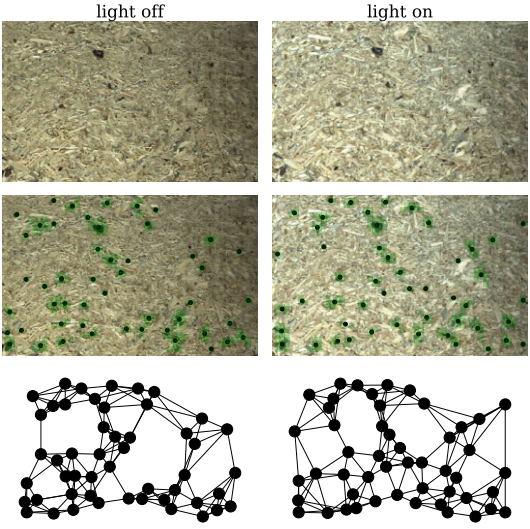
IX Understanding the limits of Test time training in Anomaly detection



X Hyperparameter-aware Benchmarking



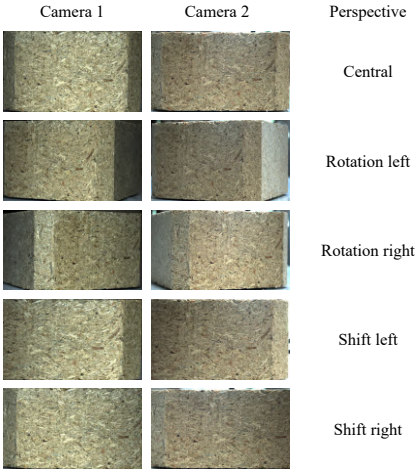
XI Abstract Re-identification



XI What is Re-identification?

- The task of distinguishing a single individual from a group of similar entities
- Pallet blocks very similar to each other
- No pretrained models

- Our dataset: 5,020 images of chipwood
- 10 images per ID



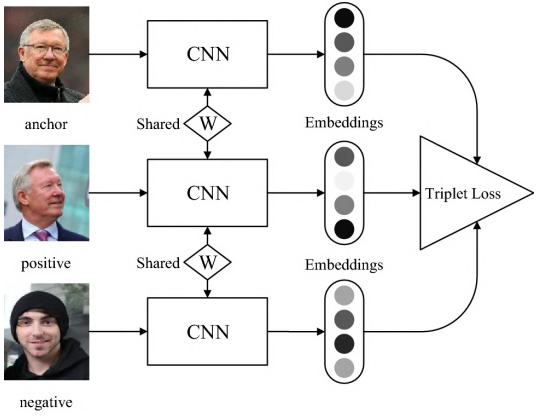
XI What are Siamese Neural Networks?

Learn representation, for example using *Triplet loss*

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

Close representation: Likely same object

Like classification, but ∞ classes

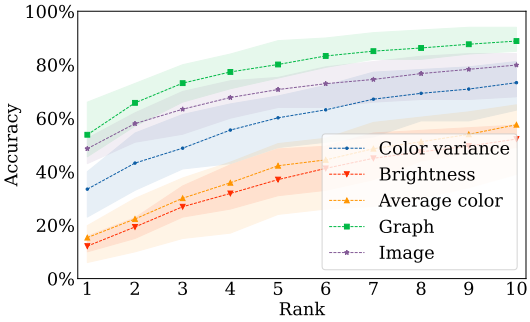


(Yu 2020)

XI Submodels

Our heterogeneous ensemble:

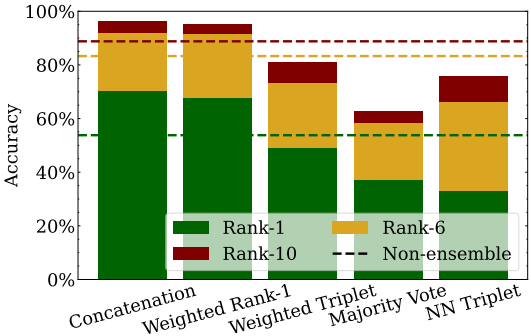
- Image-based model
- Dense model for averaged pixel values
- Dense model for averaged pixel and color values
- Dense model for pixel variance
- Graph-based model from prior work



XI Ensembling Functions

Our ensembling functions:

- Concatenating model representations
- Weighting the models
- Majority vote ensembling
- Neural Network ensembling



XI Performance

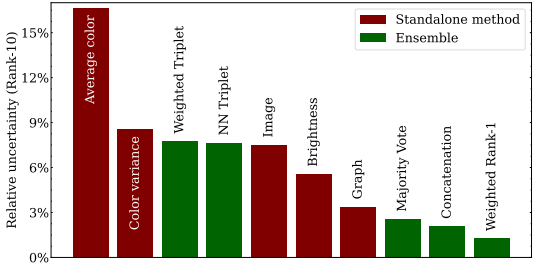
Dataset	Model	Rank-1	Rank-10	Runtime [s]
4*Wood	Ensemble	**0.703** $\pm$ 0.079	**0.964** $\pm$ 0.020	287
	Graph	0.526 $\pm$ 0.052	0.904 $\pm$ 0.045	50
	Image	0.486 $\pm$ 0.032	0.799 $\pm$ 0.060	213
	Inception	0.518 $\pm$ 0.05	0.918 $\pm$ 0.014	1,116
4*Metal	Ensemble	**0.777** $\pm$ 0.054	**0.992** $\pm$ 0.007	257
	Color Variance	0.549 $\pm$ 0.057	0.909 $\pm$ 0.022	8
	Image	0.360 $\pm$ 0.033	0.898 $\pm$ 0.047	224
	Inception	0.681 $\pm$ 0.059	0.951 $\pm$ 0.013	1,283

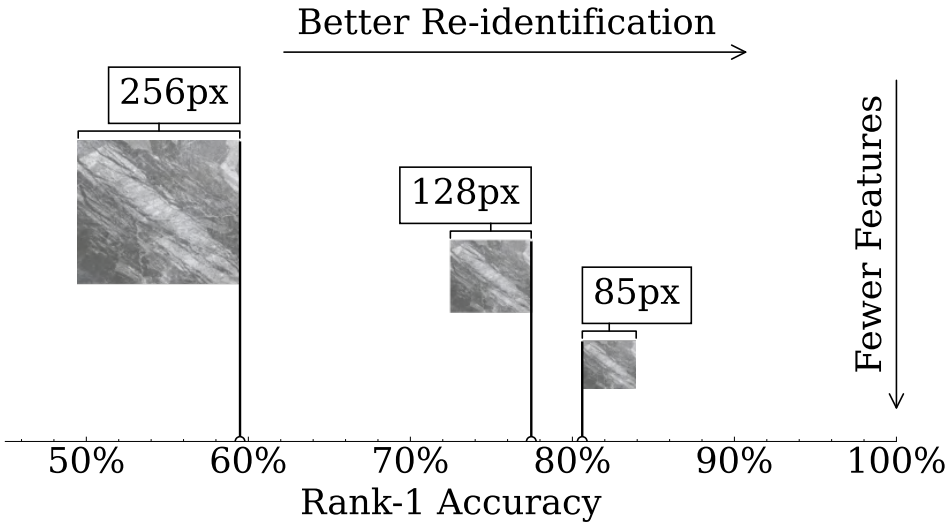
$\Rightarrow$ Ensembles work best!

XI Reliability

- Ensemble uncertainty generally lower than individual model uncertainty

⇒ Thus ensembles easier to train




Better Re-identification
256px
128px
85px
Fewer Features
50%
60%
70%
80%
90%
100%
Rank-1 Accuracy

XII Lets investigate

Lets start by looking at the representations learned

(Columns: Features, Rows: Samples)

23.92	42.37	35.80	0.00	0.00	0.00	0.00	0.00	20.09	0.00	0.00	4.32
17.52	26.86	25.60	0.00	0.00	0.00	0.00	0.00	16.45	0.00	0.00	1.28
11.75	32.19	22.43	0.00	0.00	0.00	0.00	0.00	16.54	0.00	0.00	2.89
30.68	43.65	45.56	0.00	0.00	0.00	0.00	0.00	18.81	0.00	0.00	0.00
37.24	4.28	43.45	0.00	0.00	0.00	0.00	0.00	10.17	0.00	0.00	0.00
13.95	30.84	21.15	0.00	0.00	0.00	0.00	0.00	16.17	0.00	0.00	5.46
24.25	33.45	31.71	0.00	0.00	0.00	0.00	0.00	18.60	0.00	0.00	0.00
15.28	28.16	29.38	0.00	0.00	0.00	0.00	0.00	13.92	0.00	0.00	0.00
20.23	36.47	26.49	0.00	0.00	0.00	0.00	0.00	21.21	0.00	0.00	6.59
14.32	33.69	28.84	0.00	0.00	0.00	0.00	0.00	14.29	0.00	0.00	0.18

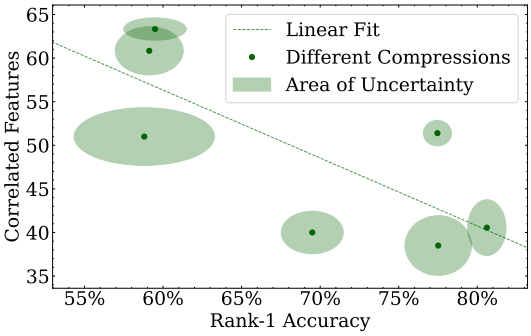
Lots of pointless features. Could this be the reason?

XII Lets check: Correlated Features

Define metric: Correlated Features

$cf = dim(R) - rank(R)$

75% correlation to accuracy



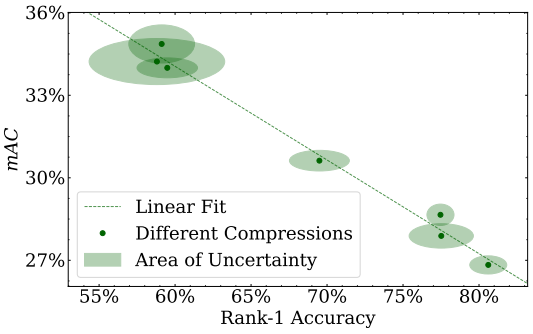
XII Lets check: Average Correlation

Better metric: Average Correlation

$mAC = \frac{2}{D \cdot (D-1)} \sum_{\nu}^{D} \sum_{\mu > \nu}^{D} \cdot \| corr^{\mu\nu} \|$

99.7% correlation to accuracy

p value $< 10^{-6} \rightarrow$ significant

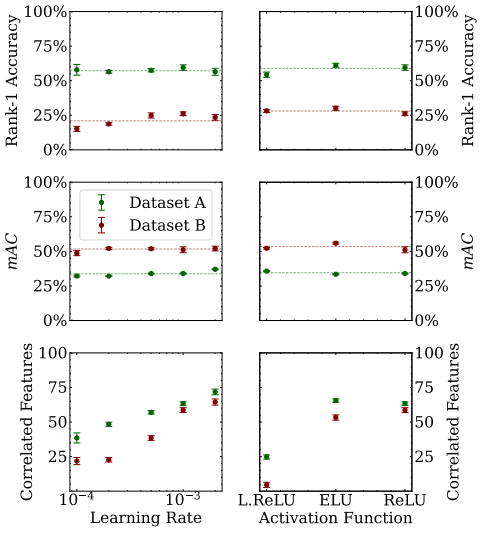


XII Hyperparameters

Let's see if we can change *cf* and *mac*

Here *Learning Rate* and *Activation Function*

cf easy to change, but *mAC* and accuracy stay constant



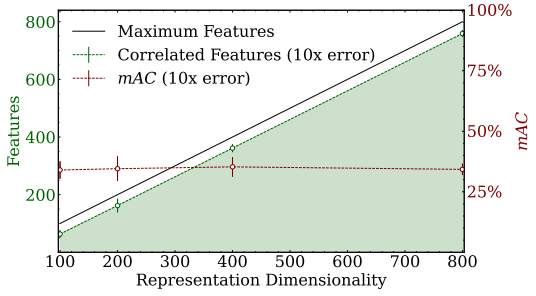
XII Some features are learned badly, can we just learn more?

Can we use more features

nontrivial features stays constant

$\rightarrow$ so no

$\exists$ maximum number of learnable features



XII Neural network size

Bigger NN have a higher limit

$\rightarrow$ Its a problem of NN not being able to fit every function! ($bias > 0$)

(remember: NN are only able to fit any function, when ∞ nodes)

most approaches use giant networks

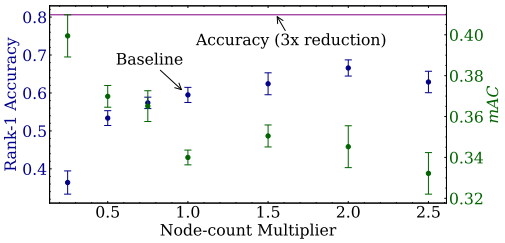


Figure 1: Accuracy and mAC when multiplying the amount of features after each hidden layer

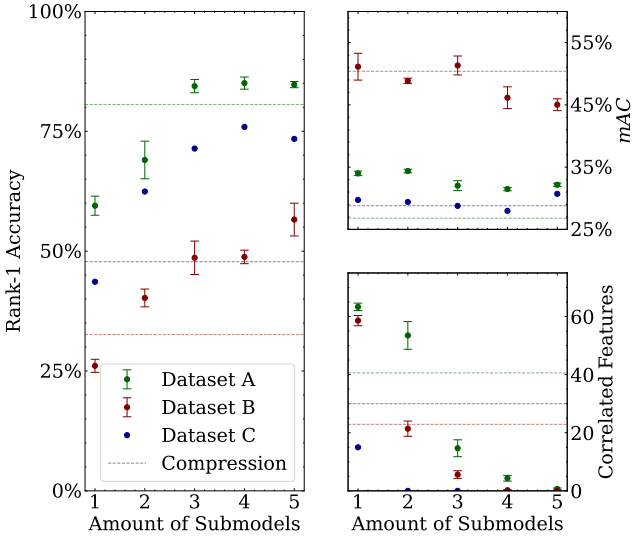
XII A better way

Ensembles can lower bias

same # features, much better reidentification

outperforms initial compression method

more effective than bigger NN



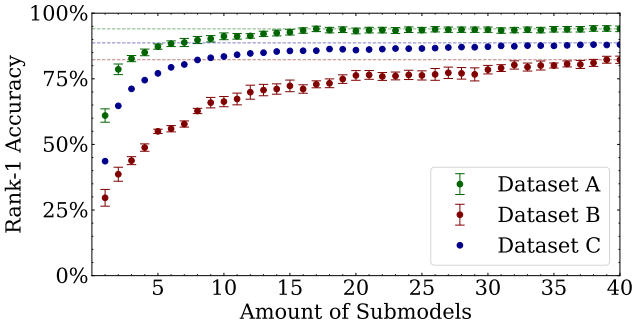
Dataset A:

- baseline: 60%
- compression: 80%
- with ensemble: 95%

Dataset C (Market 1501) almost competitive

SOTA about 95% – 98%, we have 90%

but without giant models and without pertaining



XII A better way

Usual way	Our approach
Larger and larger models	Simple Ensemble
Expensive training	Very fast training
Requires many GPUs	Can even be done on CPUs
Less effective	More effective
Limited by Computation Power	Limited by Creativity

+ Lots of potential to improve this further

Polyra — Motivation

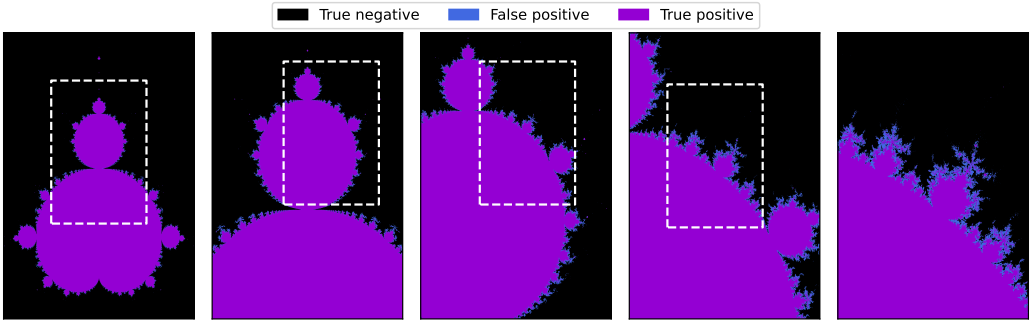
(AI generated)

Definition 5 (Polyra base shapes)**.** *We use base shapes F_i defined by the logical conjunction of two polytopes (Equation 4). If a sample lies in the condition polytope A_i, it also has to lie in the consequent polytope B_i.*

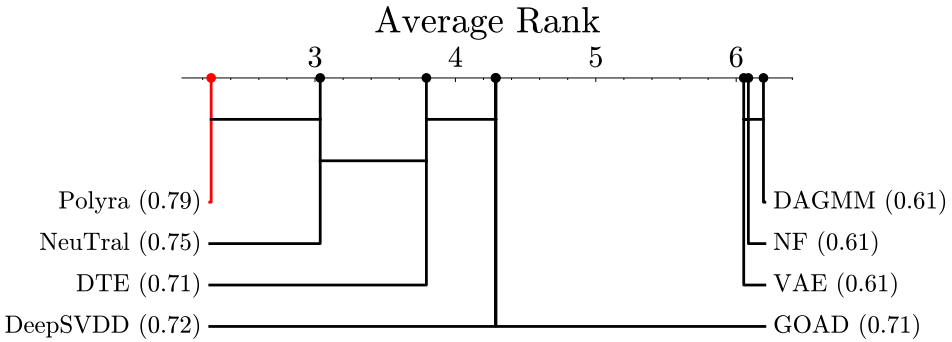
$$f_i(x) = (x \in A_i \Rightarrow x \in B_i) \Leftrightarrow f_i(x) = (x \notin A_i \vee x \in B_i) \Leftrightarrow F_i = A_i^{\complement} \cup B_i \tag{4}$$

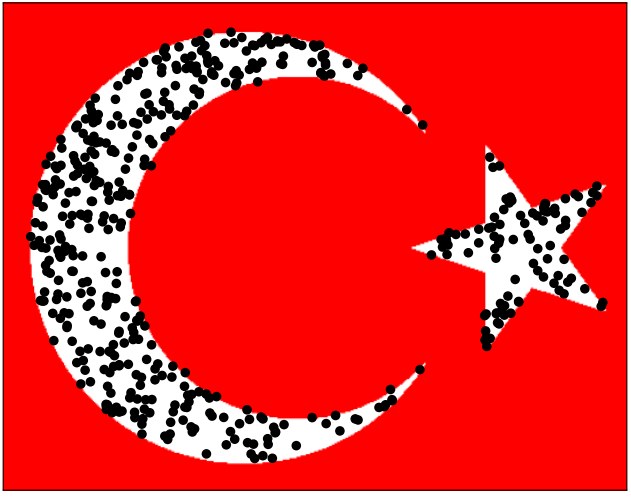
Here $A^{\complement}$ represents the complement shape to A ($a^{\complement}(x) = 1 - a(x)$). Effectively, each base shape excludes the shape $A_i \cap B_i^{\complement}$ from the shape approximated by the Polyra Swarm.

Polyra — Shape Approximation



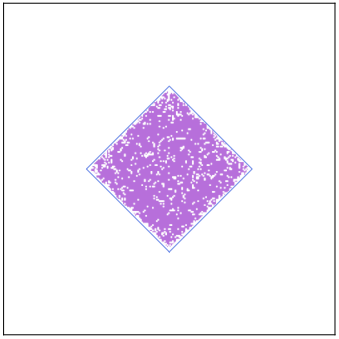
Polyra — Anomaly Detection





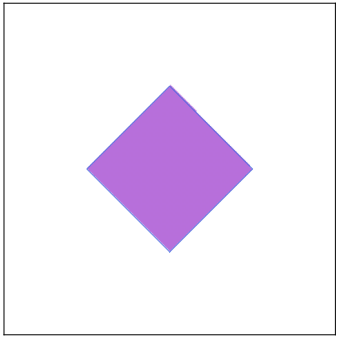
Polyra — Abstraction

Size: 72000 floats



vIOU: 0.844

Size: 15 floats

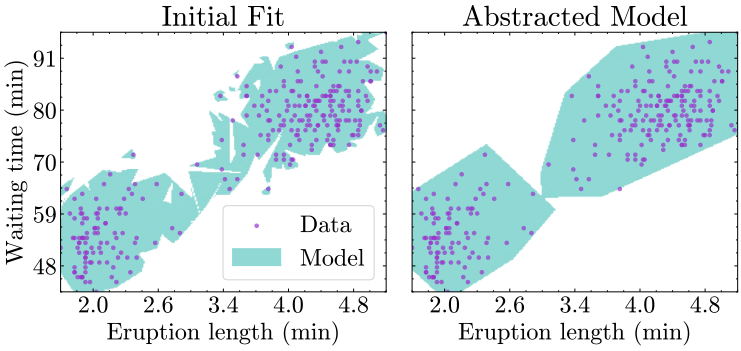


vIOU: 0.994

Polyra — Abstraction

$$\begin{bmatrix} -0.7071 & -0.7072 \\ -0.7081 & 0.7061 \\ 0.7086 & -0.7056 \\ 0.7155 & 0.6986 \\ 0.7853 & 0.6192 \end{bmatrix} \cdot x \leq \begin{bmatrix} -0.5303 \\ 0.1759 \\ 0.1784 \\ 0.8852 \\ 0.8967 \end{bmatrix}$$

Polyra — Abstraction

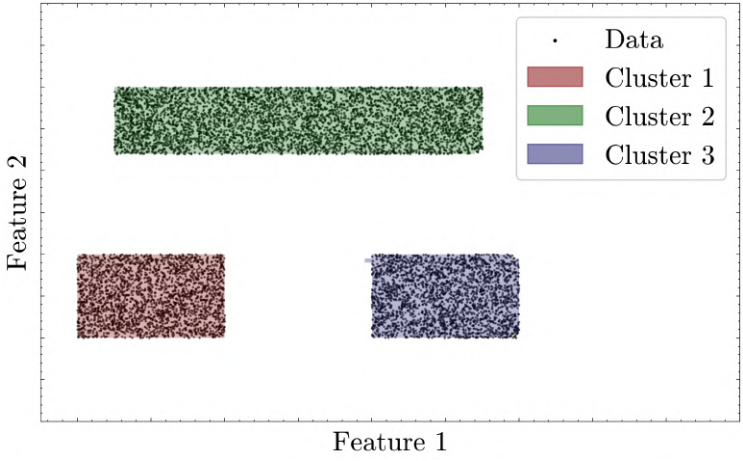


Polyra — Abstraction

$$\left(\begin{bmatrix} -0.756 & -0.6546 \\ 0.6207 & -0.784 \\ 0.8259 & -0.5639 \\ -0.6207 & 0.784 \\ 0.7643 & 0.6448 \end{bmatrix} \cdot x \leq \begin{bmatrix} -0.0522 \\ 0.095 \\ 0.1859 \\ 0.3054 \\ 0.5405 \end{bmatrix} \right)$$

$$\bigvee \left(\begin{bmatrix} 0.4978 & -0.8673 \\ -0.7498 & 0.6617 \\ -0.1934 & 0.9811 \\ -0.9716 & 0.2366 \\ 0.0267 & -0.9996 \\ -0.9982 & -0.0606 \end{bmatrix} \cdot x \leq \begin{bmatrix} -0.0259 \\ 0.1568 \\ 0.8063 \\ -0.2722 \\ -0.3493 \\ -0.4175 \end{bmatrix} \right)$$

Polyra — Clustering



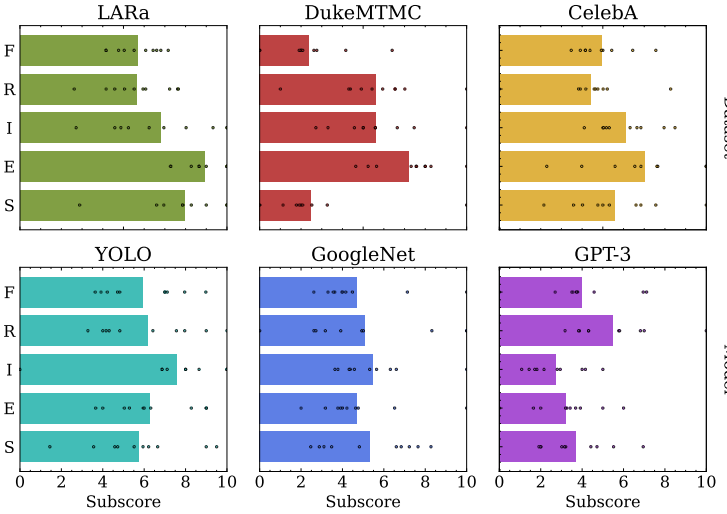
Benchmarking Trust: A Metric for Trustworthy Machine Learning

Occurrence (O)		**Significance (S)**		**Detection (D)**	
Probability		Impact		Probability	
Impossible	10	Negligible	10	Certain	10
Unlikely	9	Barely perceptible	9	High	9
Very low	7-8	Insignificant	7-8	Moderate	7-8
Low	4-6	Moderate	4-6	Low	4-6
Moderate	2-3	Severe	2-3	Very low	2-3
High	1	Extremely severe	1	Unlikely	1
Certain	0	Unacceptable	0	Impossible	0

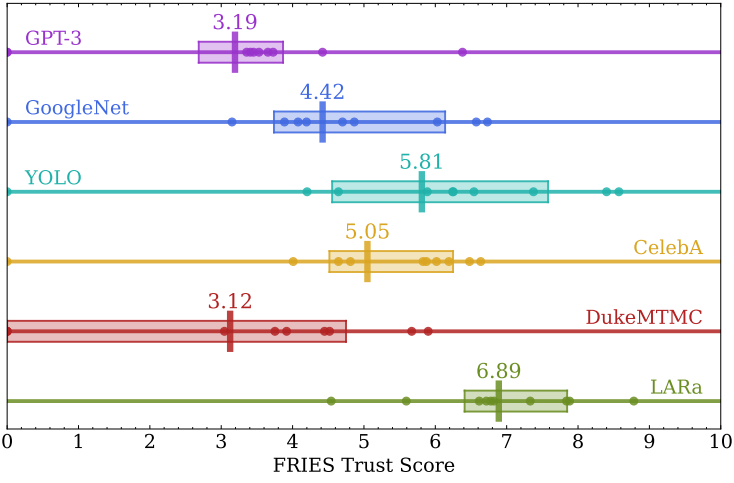
Benchmarking Trust: A Metric for Trustworthy Machine Learning

Aspect	Risk	O	S	D	Π	$\bar{\Pi}$	ω	T_ω
Fairness	Inputs requested in a biased manner	4	4	8	5.04	5.04	0.2	1.01
Robustness	Risk of model inversion attacks	4	8	9	6.6	5.89	0.2	1.18
	Risk of adversarial attacks	7	4	5	5.19			
Integrity	The model is not open source	3	9	2	3.78	3.78	0.2	0.76
Explainability	Illusion of Explanatory Depth	8	4	5	5.43	5.43	0.3	1.63
Safety	Decisions reveal sensitive information	6	3	6	4.76	4.76	0.1	0.48
							T	5.06

Benchmarking Trust: A Metric for Trustworthy Machine Learning



Benchmarking Trust: A Metric for Trustworthy Machine Learning

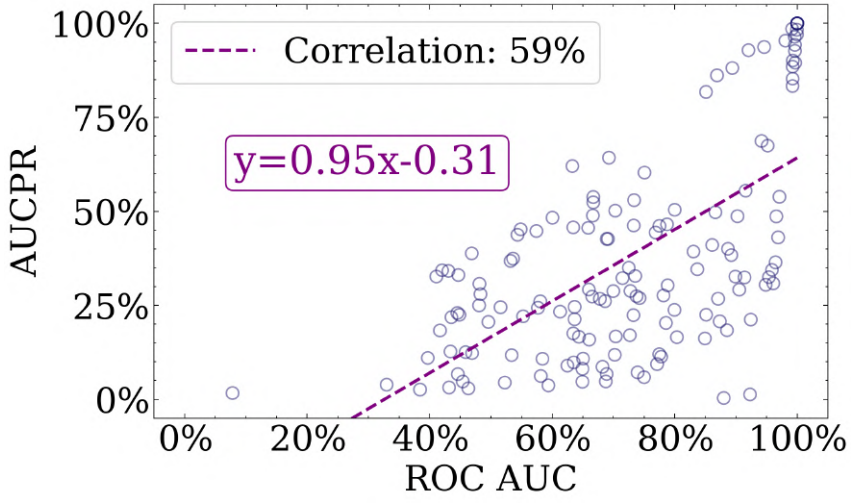


A Laser-based Volumetric Measurement Approach for Industrial Settings

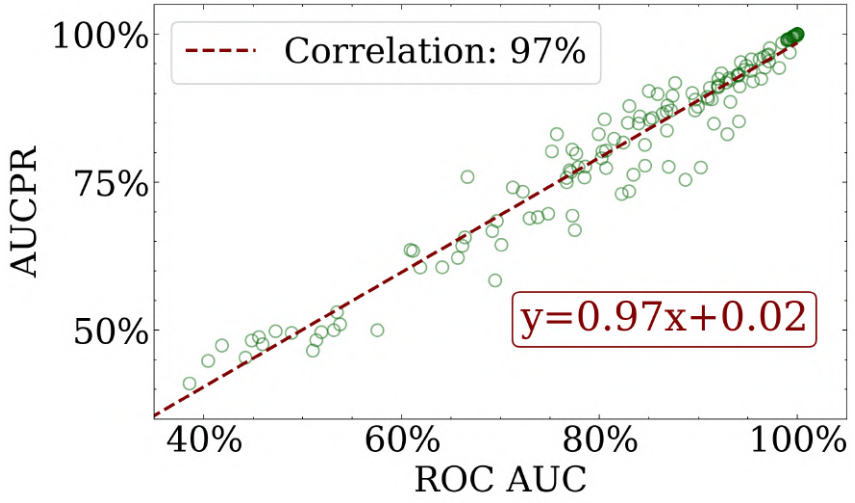
A Laser-based Volumetric Measurement Approach for Industrial Settings

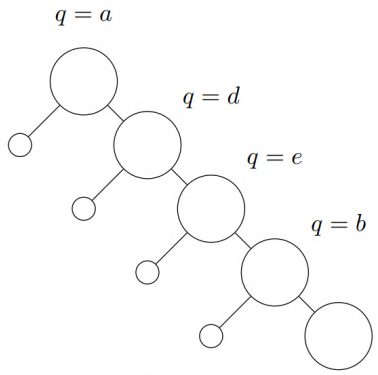
Obj.	GT [cm^3]	Min. [cm^3]	Max. [cm^3]	MPM [cm^3]	Deviation [%]
(a)	3,510	15,396	19,854	17,735	405.28
(b)	26,400	21,739	30,805	24,924	-5.59
(c)	26,400	23,973	30,464	26,840	1.67
(d)	34,810	18,023	181,974	31,225	-10.3
(e)	52,800	24,646	280,001	51,974	-1.56
(f)	233,824	51,875	1,564,753	232,170	0.71
(g)	748,800	182,211	927,067	438,988	-41.37

Exploring the Impact of Outlier Variability on Anomaly Detection Evaluation Metrics

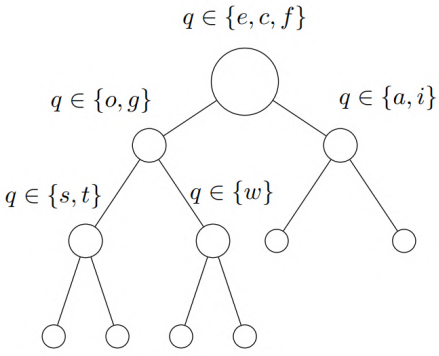


Exploring the Impact of Outlier Variability on Anomaly Detection Evaluation Metrics

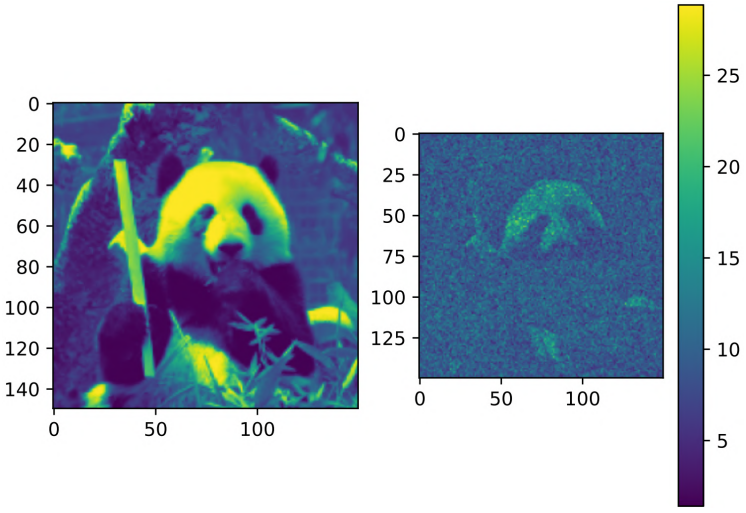


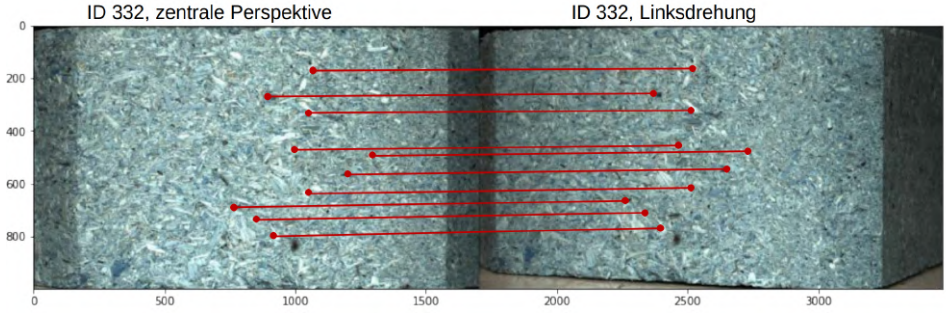


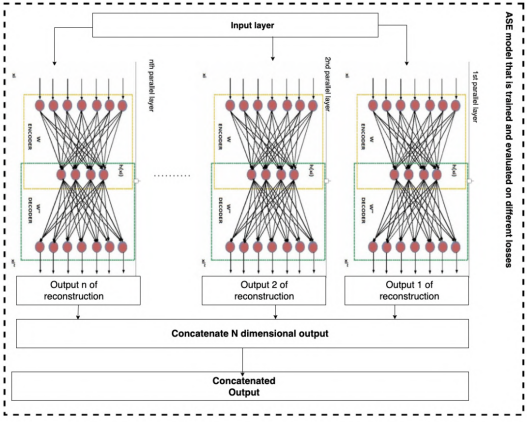
(a) Unbalanced tree

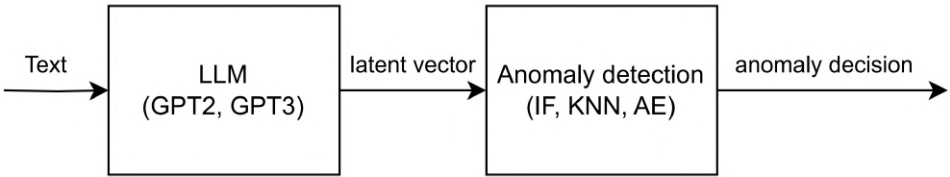


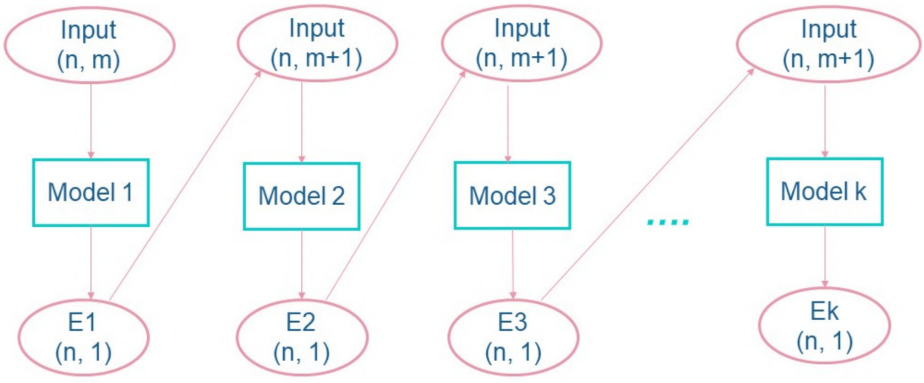
(b) Balanced tree


0
20
40
60
80
100
120
140
0
50
100
0
25
50
75
100
125
0
50
100
25
20
15
10
5


ID 332, zentrale Perspektive
ID 332, Linksdrehung
0
200
400
600
800
0
500
1000
1500
2000
2500
3000

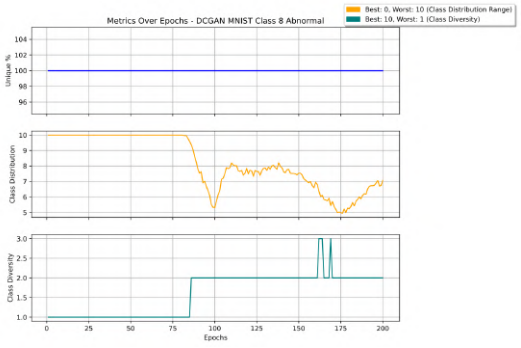

ASE model that is trained and evaluated on different losses
Input layer
nth parallel layer
2nd parallel layer
1st parallel layer
Output n of reconstruction
Output 2 of reconstruction
Output 1 of reconstruction
Concatenate N dimensional output
Concatenated Output

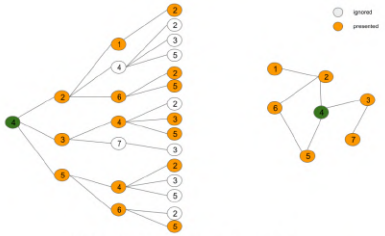

Text
LLM
(GPT2, GPT3)
latent vector
Anomaly detection
(IF, KNN, AE)
anomaly decision


Input
(n, m)
Input
(n, m+1)
Input
(n, m+1)
Input
(n, m+1)
Model 1
Model 2
Model 3
....
Model k
E1
(n, 1)
E2
(n, 1)
E3
(n, 1)
Ek
(n, 1)

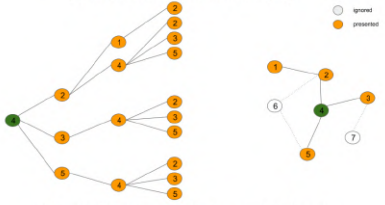
$$\text{Loss} = \frac{1}{N_{\text{unlabeled}}} \sum_{i=1}^{N_{\text{unlabeled}}} \left(\max\left(1 - y_{\text{true},i} \cdot y_{\text{pred},i}, 0\right)\right)^2 + \frac{1}{N_{\text{normal}}} \sum_{i=1}^{N_{\text{normal}}} \left(\max\left(1 - y_{\text{true},i} \cdot y_{\text{pred},i}, 0\right)\right)^2 - \frac{1}{N_{\text{anomaly}}} \sum_{i=1}^{N_{\text{anomaly}}} \left(\max(1 - y_{\text{true},i} \cdot y_{\text{pred},i}, 0)\right)^2,$$

Logien Mekki

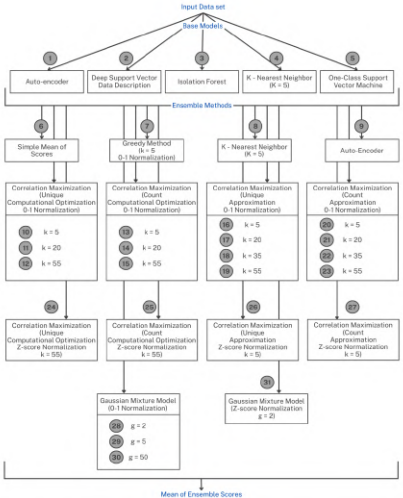


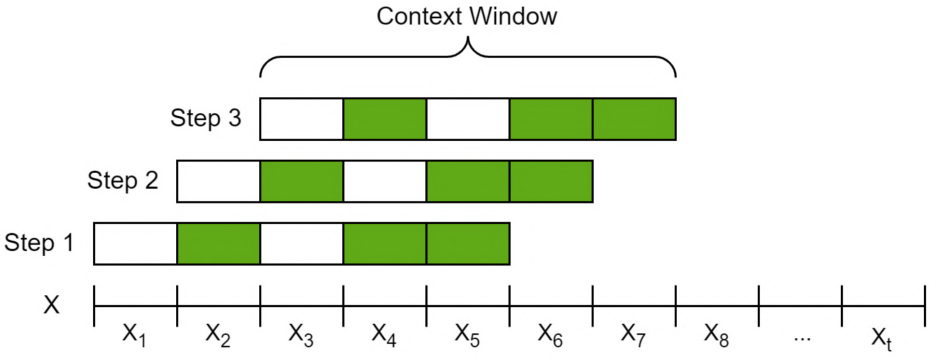


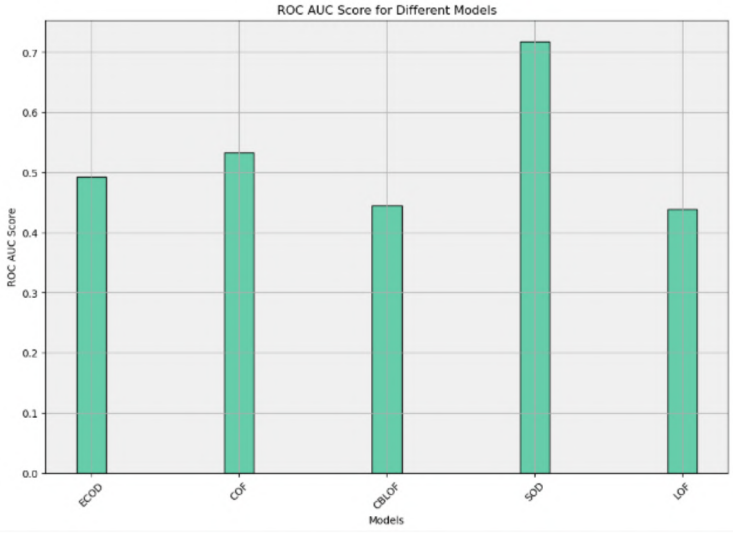
(b) GraphSAGE: layer-wise nodes sampling.



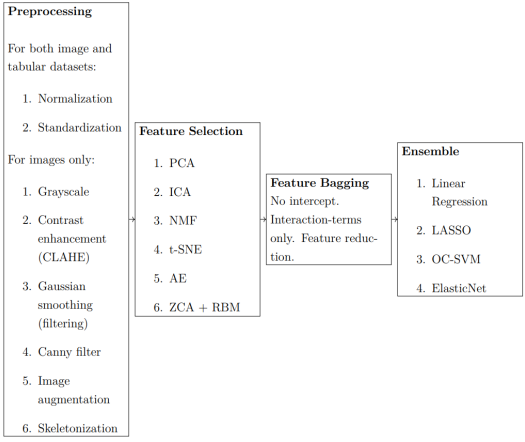
(c) GraphSAINT: full propagation in a sampled subgraph.


Input Data set
Base Models
1
2
3
4
5
Auto-encoder
Deep Support Vector Data Description
Isolation Forest
K - Nearest Neighbor (K = 5)
One-Class Support Vector Machine
Ensemble Methods
6
7
8
9
Simple Mean of Scores
Greedy Method (k = 5 0-1 Normalization)
K - Nearest Neighbor (K = 5)
Auto-Encoder
Correlation Maximization (Unique Computational Optimization 0-1 Normalization)
10 k = 5
11 k = 20
12 k = 55
Correlation Maximization (Count Computational Optimization 0-1 Normalization)
13 k = 5
14 k = 20
15 k = 55
Correlation Maximization (Unique Approximation 0-1 Normalization)
16 k = 5
17 k = 20
18 k = 35
19 k = 55
Correlation Maximization (Count Approximation 0-1 Normalization)
20 k = 5
21 k = 20
22 k = 35
23 k = 55
24
25
26
27
Correlation Maximization (Unique Computational Optimization Z-score Normalization k = 55)
Correlation Maximization (Count Computational Optimization Z-score Normalization k = 55)
Correlation Maximization (Unique Approximation Z-score Normalization k = 5)
Correlation Maximization (Count Approximation Z-score Normalization k = 5)
31
Gaussian Mixture Model (0-1 Normalization)
28 g = 2
29 g = 5
30 g = 50
Gaussian Mixture Model (Z-score Normalization g = 2)
Mean of Ensemble Scores


Context Window
Step 3
Step 2
Step 1
X
X_1
X_2
X_3
X_4
X_5
X_6
X_7
X_8
...
X_t


ROC AUC Score for Different Models
ROC AUC Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
ECOD
COF
CBLOF
SOD
LOF
Models

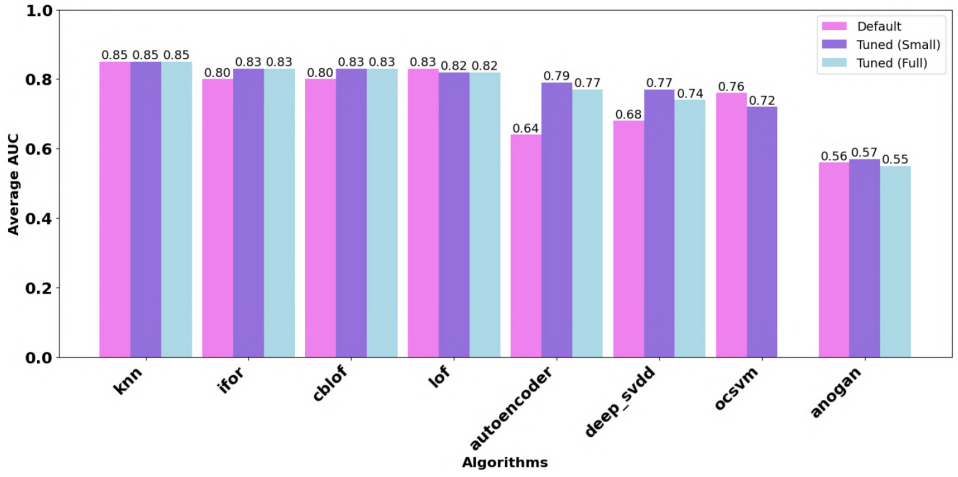
Vanlal Peka

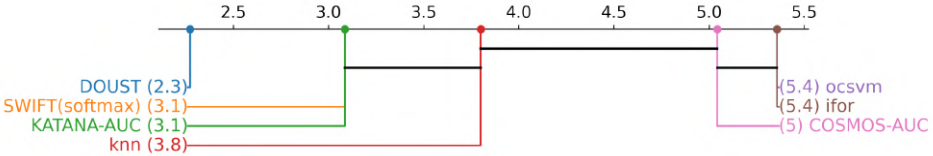

Preprocessing
For both image and tabular datasets:
1. Normalization
2. Standardization
For images only:
1. Grayscale
2. Contrast enhancement (CLAHE)
3. Gaussian smoothing (filtering)
4. Canny filter
5. Image augmentation
6. Skeletonization
Feature Selection
1. PCA
2. ICA
3. NMF
4. t-SNE
5. AE
6. ZCA + RBM
Feature Bagging
No intercept. Interaction-terms only. Feature reduction.
Ensemble
1. Linear Regression
2. LASSO
3. OC-SVM
4. ElasticNet

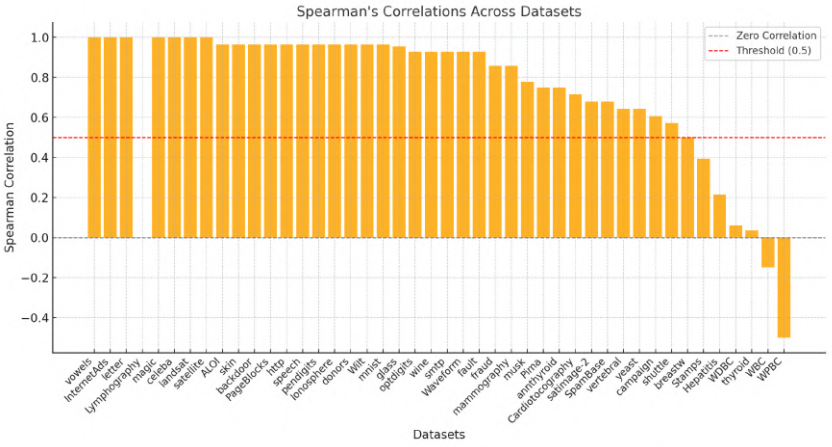
Here is a table detailing how many times higher the robustness metrics of the standard models were when compared to the Lipschitz models. For clarity's sake, we give the example of 8.24 which indicates that the standard DeepSVDD One-Class measured a lower bound on the Lipschitz constant that was 8.24 times higher than the Lipschitz variant.

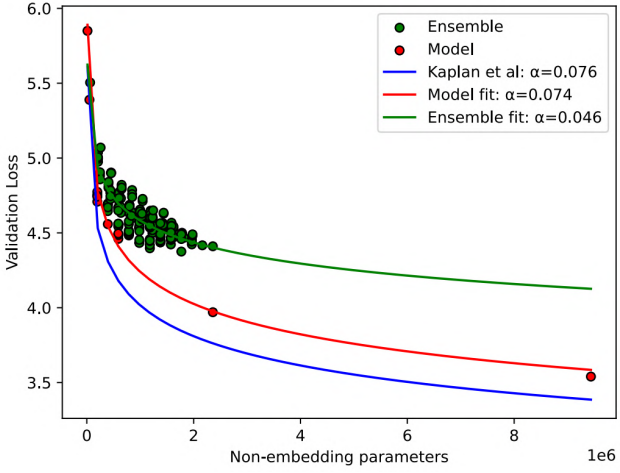
Data Set	DCAE	DeepSVDD SB	DeepSVDD OC	AnoGAN
MNIST	9.88	2.49	2.27	57.08
CIFAR10	2.26	6.52	8.24	21.80

Jessia Erappakkal Joby

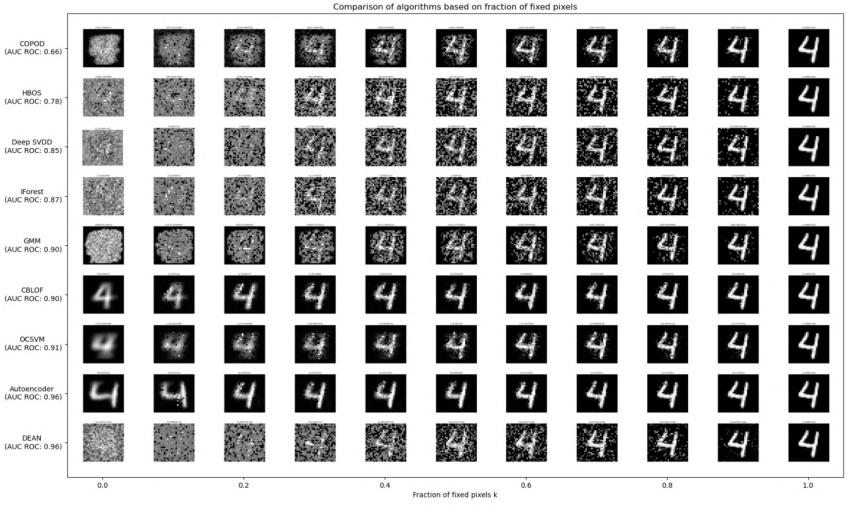



2.5
3.0
3.5
4.0
4.5
5.0
5.5
DOUST (2.3)
SWIFT(softmax) (3.1)
KATANA-AUC (3.1)
knn (3.8)
(5.4) ocsvm
(5.4) ifor
(5) COSMOS-AUC


Spearman's Correlations Across Datasets
Zero Correlation
Threshold (0.5)
Spearman Correlation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
vowels
InternetAds
letter
Lymphography
magic
celeba
landsat
satellite
ALOI
skin
backdoor
PageBlocks
http
speech
pendigits
Ionosphere
donors
Wilt
mnist
glass
optdigits
wine
smtp
Waveform
fault
fraud
mammography
musk
Pima
annthyroid
Cardiotocography
satimage-2
SpamBase
vertebral
yeast
campaign
shuttle
breastw
Stamps
Hepatitis
WDBC
thyroid
WBC
WPBC
Datasets


Ensemble
Model
Kaplan et al: α=0.076
Model fit: α=0.074
Ensemble fit: α=0.046
Validation Loss
Non-embedding parameters
1e6
6.0
5.5
5.0
4.5
4.0
3.5
0
2
4
6
8

Ben Ernst


Comparison of algorithms based on fraction of fixed pixels
COPOD
(AUC ROC: 0.66)
HBOS
(AUC ROC: 0.78)
Deep SVDD
(AUC ROC: 0.85)
IForest
(AUC ROC: 0.87)
GMM
(AUC ROC: 0.90)
CBLOF
(AUC ROC: 0.90)
OCSVM
(AUC ROC: 0.91)
Autoencoder
(AUC ROC: 0.96)
DEAN
(AUC ROC: 0.96)
0.0
0.2
0.4
0.6
0.8
1.0
Fraction of fixed pixels k

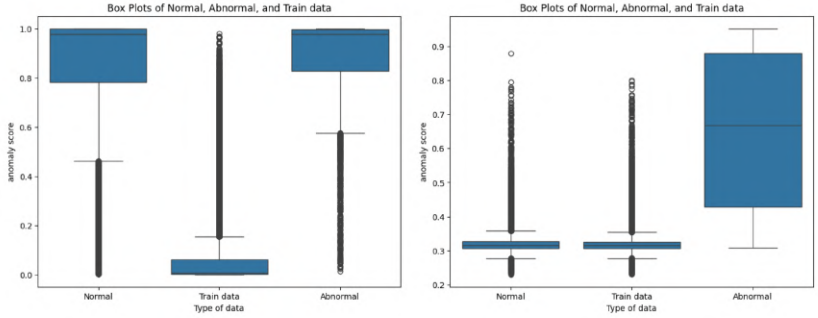
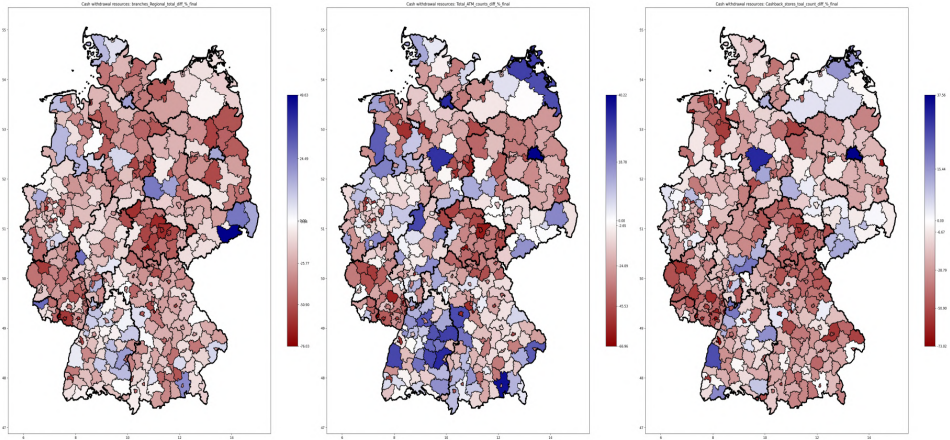
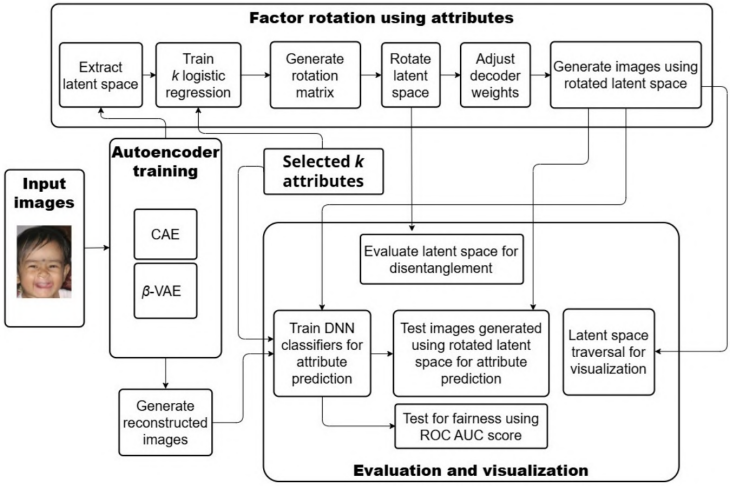


Figure 4.3: Box plots of anomaly scores for D2 dataset. The left plot represents scores before shuffling (D2), while the right plot corresponds to shuffled data ($D2_{\text{shuff}}$).



(a) Bank branches count (b) ATM counts (c) Cashback stores count


Factor rotation using attributes
Extract latent space
Train k logistic regression
Generate rotation matrix
Rotate latent space
Adjust decoder weights
Generate images using rotated latent space
Input images
Autoencoder training
CAE
β-VAE
Selected k attributes
Evaluate latent space for disentanglement
Train DNN classifiers for attribute prediction
Test images generated using rotated latent space for attribute prediction
Latent space traversal for visualization
Generate reconstructed images
Test for fairness using ROC AUC score
Evaluation and visualization