ADBench: Anomaly Detection Benchmark

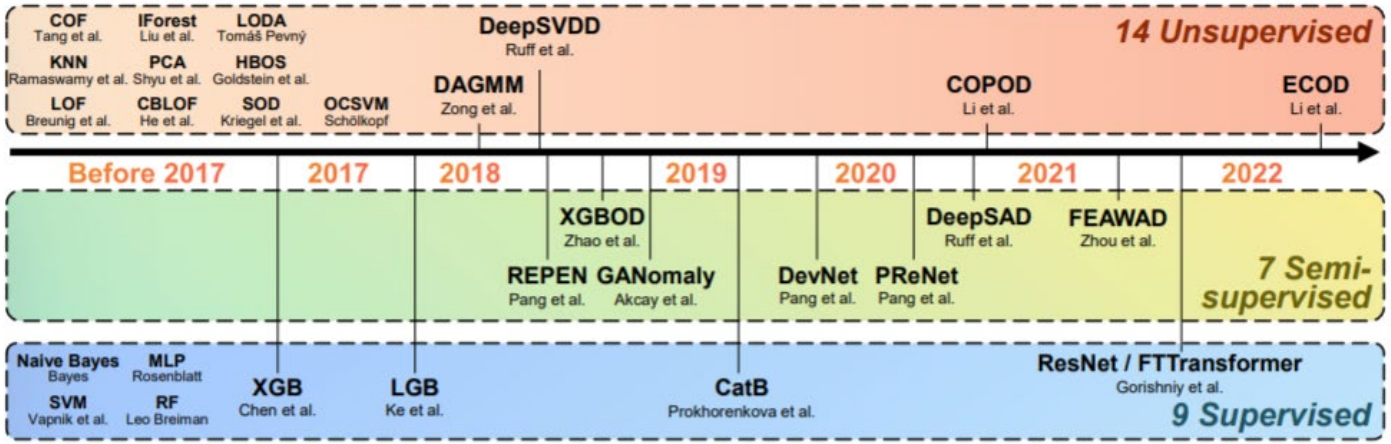
Songqiao Han[1], Xiyang Hu[2], Hailiang Huang[1], Minqi Jiang[1], Yue Zhao[2]
1 Shanghai University of Finance and Economics
2 Carnegie Mellon University

NEURAL INFORMATION PROCESSING SYSTEMS

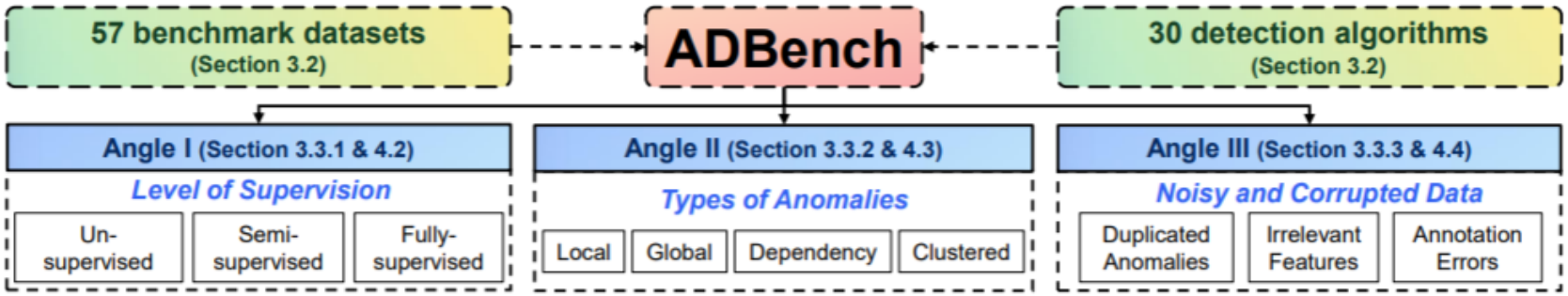
What did ADBench do?

Given **a long list of anomaly detection algorithms** developed in the last few decades, how do they perform with regard to:

(i) varying levels of supervision;
(ii) different types of anomalies;
(iii) noisy and corrupted data?



Based on the above questions, we design the proposed ADBench via **3 Angles**:



Contribution of ADBench

Data	# Samples	# Features	# Anomaly	% Anomaly	Category	Reference
ALOI	49534	27	1508	3.04	Image	[42]
annthyroid	7200	6	534	7.42	Healthcare	[141]
backdoor	95329	196	2329	2.44	Network	[119]
breastw	683	9	239	34.99	Healthcare	[173]
campaign	41188	62	4640	11.27	Finance	[131]
cardio	1831	21	176	9.61	Healthcare	[12]
Cardiotocography	2114	21	466	22.04	Healthcare	[12]
celeba	202599	39	4547	2.24	Image	[131]
census	299285	500	18568	6.20	Sociology	[131]
cover	286048	10	2747	0.96	Botany	[18]
donors	619326	10	36710	5.93	Sociology	[131]
fault	1941	27	673	34.67	Physical	[42]
fraud	284807	29	492	0.17	Finance	[131]
glass	214	7	9	4.21	Forensic	[43]
Hepatitis	80	19	13	16.25	Healthcare	[36]
http	567498	3	2211	0.39	Web	[145]
InternetAds	1966	1555	368	18.72	Image	[25]
Ionosphere	351	33	126	35.90	Oryctognosy	[163]
landsat	6435	36	1333	20.71	Astronautics	[42]
letter	1600	32	100	6.25	Image	[48]
Lymphography	148	18	6	4.05	Healthcare	[26]
magic.gamma	19020	10	6688	35.16	Physical	[42]
mammography	11183	6	260	2.32	Healthcare	[176]
mnist	7603	100	700	9.21	Image	[90]
musk	3062	166	97	3.17	Chemistry	[37]
optdigits	5216	64	150	2.88	Image	[10]
PageBlocks	5393	10	510	9.46	Document	[113]
pendigits	6870	16	156	2.27	Image	[9]
Pima	768	8	268	34.90	Healthcare	[145]
satellite	6435	36	2036	31.64	Astronautics	[145]
satimage-2	5803	36	71	1.22	Astronautics	[145]
shuttle	49097	9	3511	7.15	Astronautics	[145]
skin	245057	3	50859	20.75	Image	[42]
smtp	95156	3	30	0.03	Web	[145]
SpamBase	4207	57	1679	39.91	Document	[25]
speech	3686	400	61	1.65	Linguistics	[23]
Stamps	340	9	31	9.12	Document	[25]
thyroid	3772	6	93	2.47	Healthcare	[142]
vertebral	240	6	30	12.50	Biology	[17]
vowels	1456	12	50	3.43	Linguistics	[82]
Waveform	3443	21	100	2.90	Physics	[107]
WBC	223	9	10	4.48	Healthcare	[114]
WDBC	367	30	10	2.72	Healthcare	[114]
Wilt	4819	5	257	5.33	Botany	[25]
wine	129	13	10	7.75	Chemistry	[2]
WPBC	198	33	47	23.74	Healthcare	[114]
yeast	1484	8	507	34.16	Biology	[66]
CIFAR10	5263	512	263	5.00	Image	[81]
FashionMNIST	6315	512	315	5.00	Image	[178]
MNIST-C	10000	512	500	5.00	Image	[120]
MVTec-AD	See Table B2.				Image	[16]
SVHN	5208	512	260	5.00	Image	[121]
Agnews	10000	768	500	5.00	NLP	[192]
Amazon	10000	768	500	5.00	NLP	[63]
Imdb	10000	768	500	5.00	NLP	[111]
Yelp	10000	768	500	5.00	NLP	[192]
20newsgroups	See Table B3.				NLP	[86]

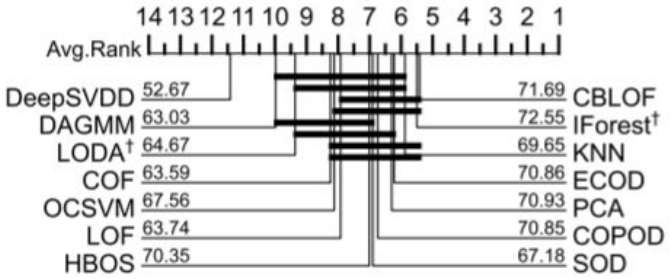
Benchmark	Coverage (§3.2)		Data Source		Algorithm Type		Comparison Angle (§3.3)		
	# datasets	# algo.	Real-world	Synthetic	Shallow	DL	Supervision	Types	Robustness
Ruff et al. [150]	3	9	✓	✓	✓	✓	✗	✓	✗
Goldstein et al. [53]	10	19	✓	✗	✓	✗	✗	✓	✗
Domingues et al. [38]	15	14	✓	✗	✓	✗	✗	✗	✓
Soenen et al. [164]	16	6	✓	✗	✓	✗	✗	✗	✗
Steinbuss et al. [166]	19	4	✗	✓	✓	✗	✗	✓	✗
Emmott et al. [42]	19	8	✓	✓	✓	✗	✗	✓	✓
Campos et al. [25]	23	12	✓	✗	✓	✗	✗	✗	✗
ADBench (ours)	57	30	✓	✓	✓	✓	✓	✓	✓

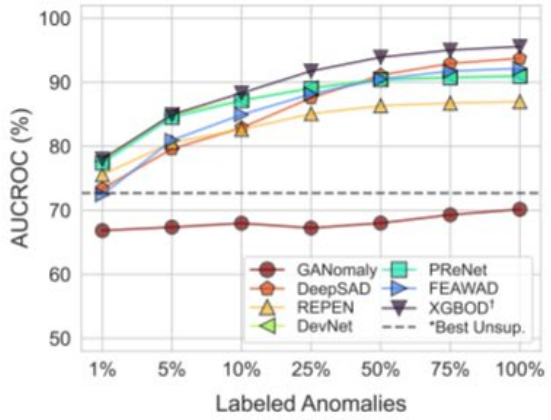
Compared to the existing AD benchmark, ADBench includes:

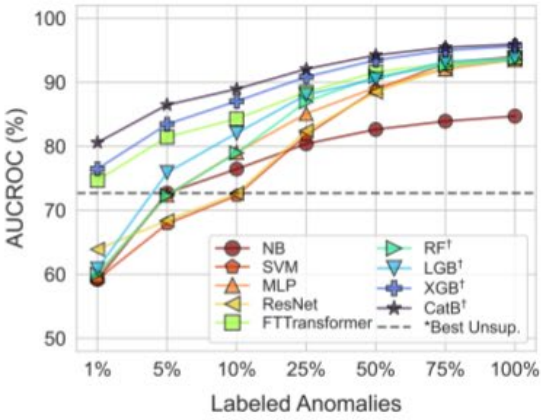
- Most datasets (57), including 10 **NLP** and **CV** datasets transformed by the pretrained model
- Most algorithms (30), including both **shallow** and **deep** learning algorithms
- Both **real-world** and **synthetic** datasets
- Multiple comparison angles
- Evaluation with both ML metrics and **statistical tests**
- Extensive experiments (**98, 436**)

Minqi Jiang (jiangmq95@163.com)

Angle I: Availability of Ground Truth Labels (Supervision)



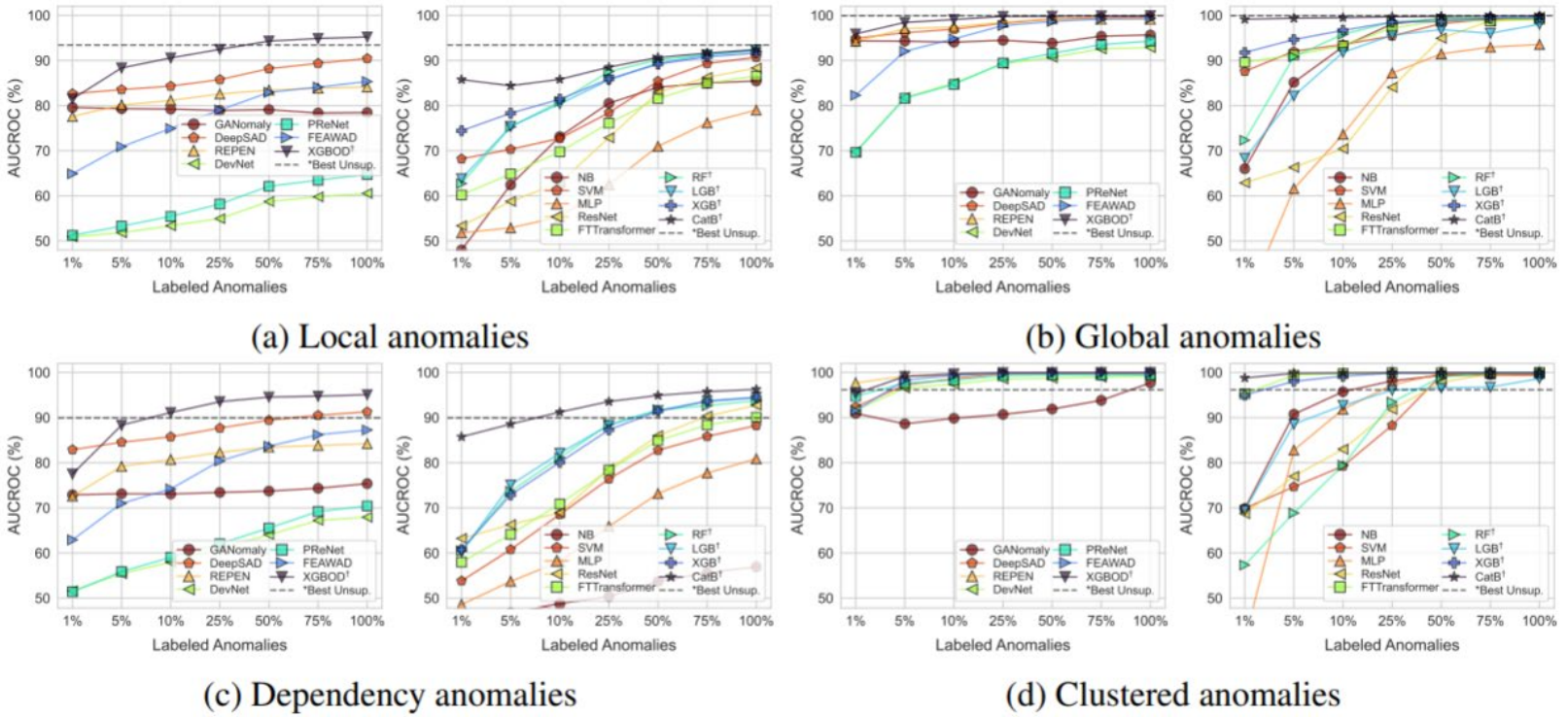




‼ Surprisingly **none** of the benchmarked unsupervised algorithms is statistically better than others, emphasizing the importance of **algorithm selection**;

‼ With merely 1% labeled anomalies, most semi-supervised methods can **outperform** the best unsupervised method, justifying the **importance of supervision**.

Angle II: Types of Anomalies



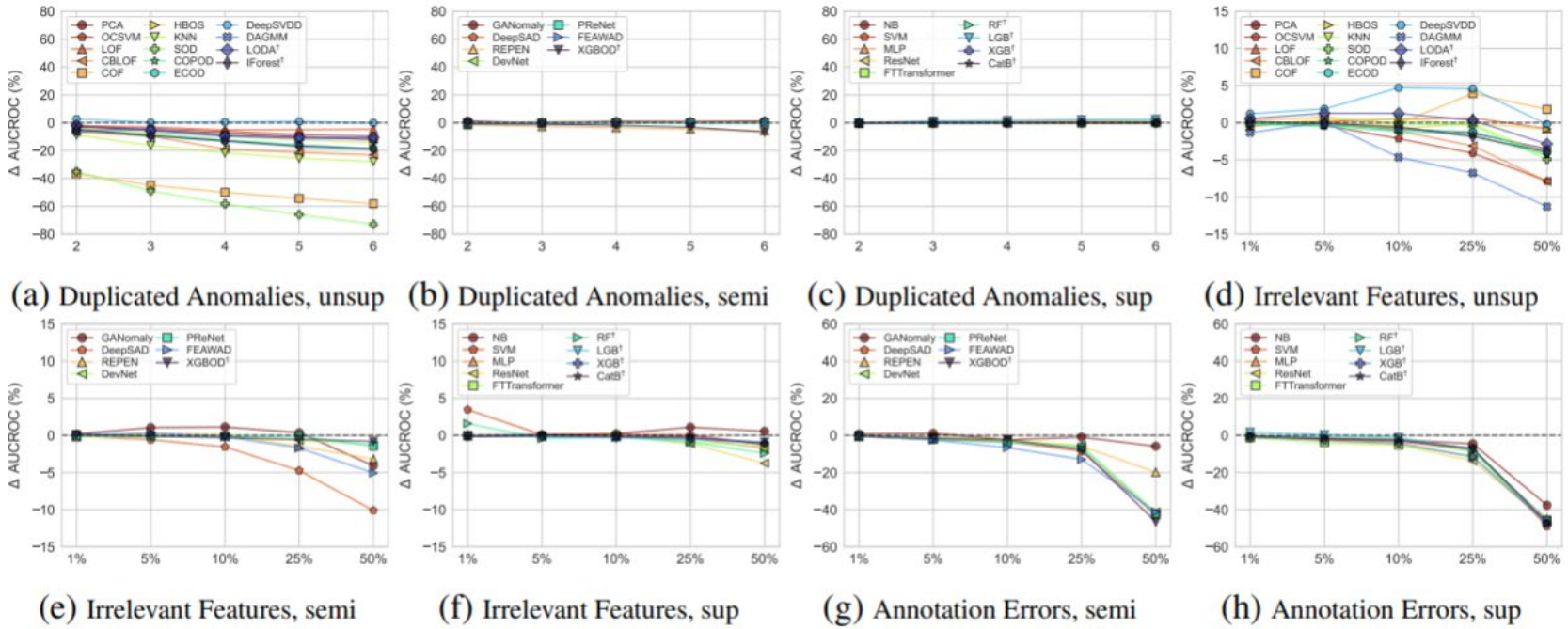
(a) Local anomalies (b) Global anomalies

(c) Dependency anomalies (d) Clustered anomalies

‼ We observe that best unsupervised methods for specific types of anomalies are **even better than** semi- and fully-supervised methods;

‼ That is to say, the **prior knowledge of data type** could be more valuable than that of **labeled anomalies**, revealing the necessity of understanding data characteristics.

Angle III: Model Robustness with Noisy and Corrupted Data



(a) Duplicated Anomalies, unsup (b) Duplicated Anomalies, semi (c) Duplicated Anomalies, sup (d) Irrelevant Features, unsup

(e) Irrelevant Features, semi (f) Irrelevant Features, sup (g) Annotation Errors, semi (h) Annotation Errors, sup

‼ **Semi-supervised** methods show potential in achieving robustness in noisy and corrupted data, possibly due to their efficiency in using labels and feature selection.

Future Direction

Based on the experimental results and analysis, we give the several possible future direction for AD community:

- Unsupervised Algorithm Evaluation, Selection, and Design;
- Semi-supervised Learning;
- Leveraging Anomaly Types as Valuable Prior Knowledge;
- Noise-resilient AD Algorithms.

ADGym: Design Choices for Deep Anomaly Detection

Minqi Jiang[1], Chaochuan Hou[1], Ao Zheng[1], Songqiao Han[1], Hailiang Huang[1],
Qingsong Wen[2], Xiyang Hu[3], Yue Zhao[4]

1 Shanghai University of Finance and Economics
2 DAMO Academy, Alibaba Group
3 Carnegie Mellon University
4 University of Southern California

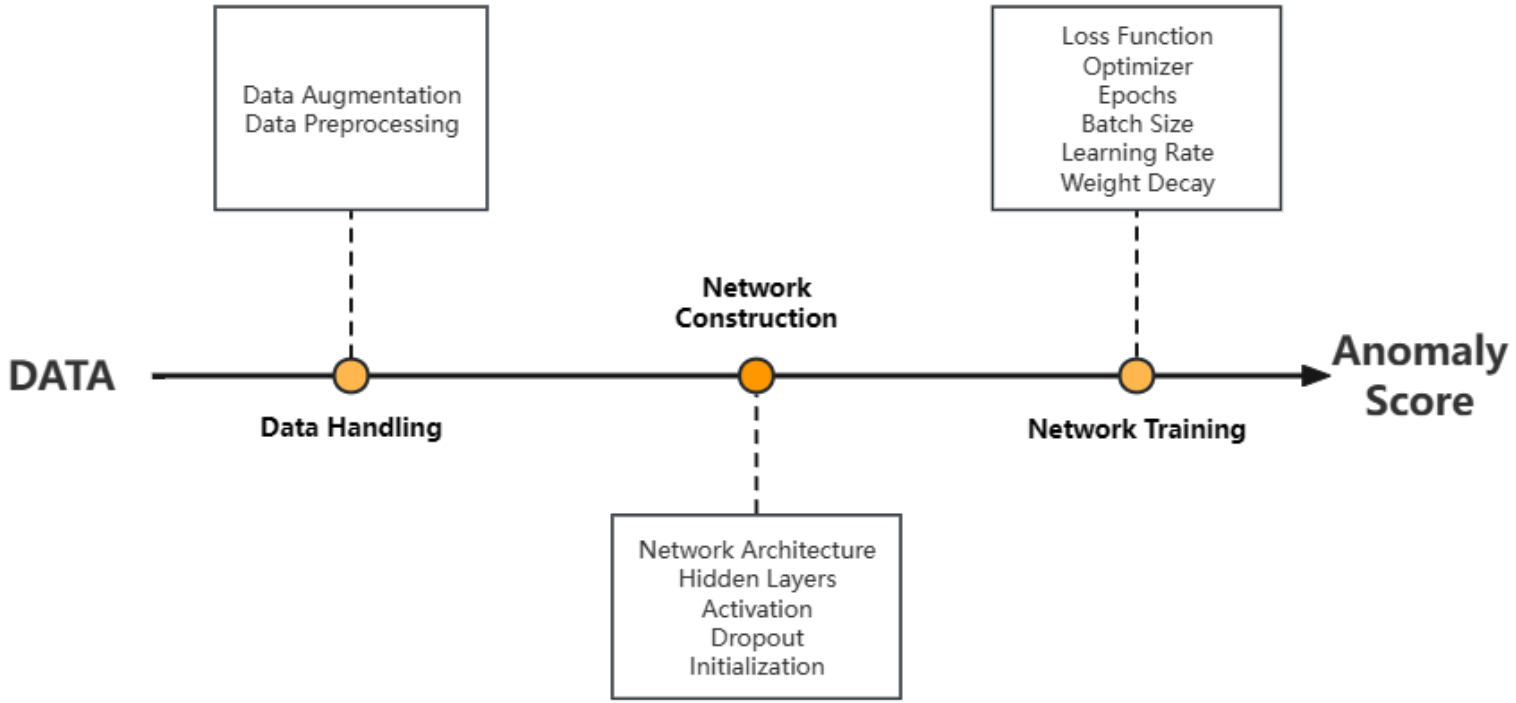
NEURAL INFORMATION PROCESSING SYSTEMS

What did ADGym do?

Given **a long list of anomaly detection algorithms** developed in the last few decades:

(i) Which components (i.e., design choices) of deep AD methods are pivotal in detecting anomalies?

(ii) How can we construct tailored AD algorithms for specific datasets by selecting the best design choices automatically, rather than relying on artificially generic solutions?



What did ADGym do?

Question1: Which components (i.e., design choices) of deep AD methods are pivotal in detecting anomalies?

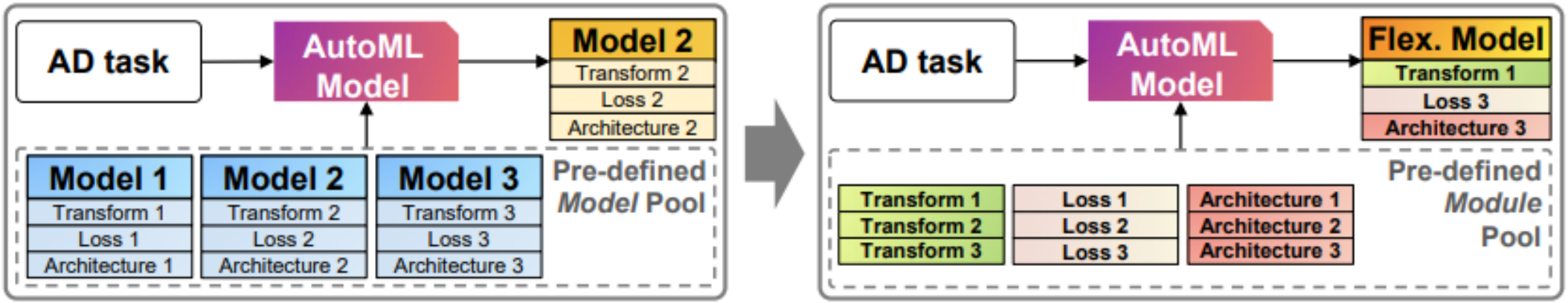
Based on this question, we provide an extensive list of specific **design choices** for deep AD models, and we examine various design combinations on large benchmark datasets.

Pipeline	Design Dimensions	Design Choices
Data Handling	Data Augmentation	[Oversampling, SMOTE, Mixup, GAN]
	Data Preprocessing	[MinMax, Normalization]
Network Construction	Network Architecture	[MLP, AutoEncoder, ResNet, FTTransformer]
	Hidden Layers	[[20], [100, 20], [100, 50, 20]]
	Activation	[Tanh, ReLU, LeakyReLU]
	Dropout	[0.0, 0.1, 0.3]
	Initialization	[default, Xavier (normal), Kaiming (normal)]
Network Training	Loss Function	[BCE, Focal, Minus, Inverse, Hinge, Deviation, Ordinal]
	Optimizer	[SGD, Adam, RMSprop]
	Epochs	[20, 50, 100]
	Batch Size	[16, 64, 256]
	Learning Rate	[1e-2, 1e-3]
	Weight Decay	[1e-2, 1e-4]

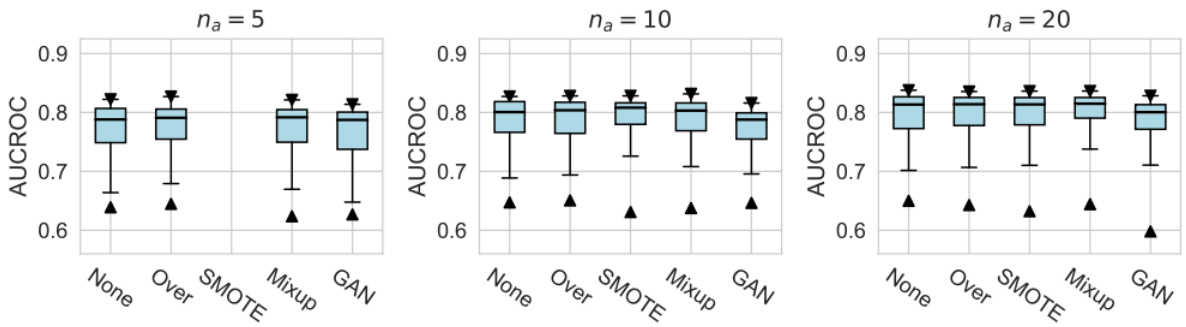
What did ADGym do?

Question2: How can we construct tailored AD algorithms for specific datasets by selecting the best design choices automatically, rather than relying on artificially generic solutions?

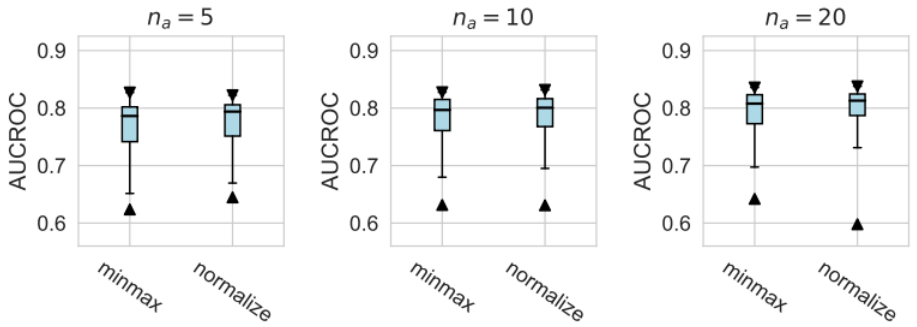
Based on this question, we propose the first automatic pipeline for design choice selection for weakly-supervised AD, where very few labels are available in model building. We find models crafted using ADGym markedly surpass current state-of-the-art techniques.



Goal I: Understanding Design Choices of Deep Anomaly Detection



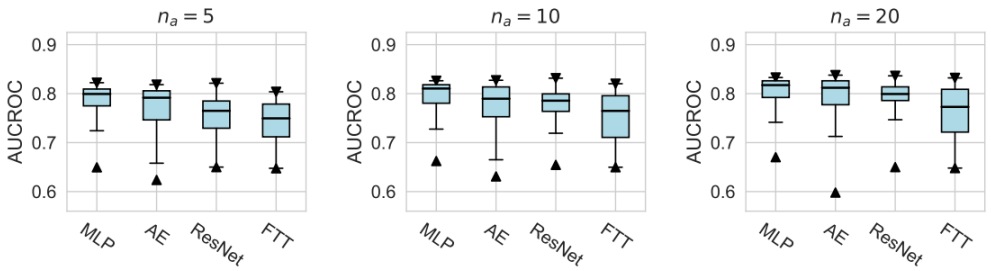
(a) Data augmentation



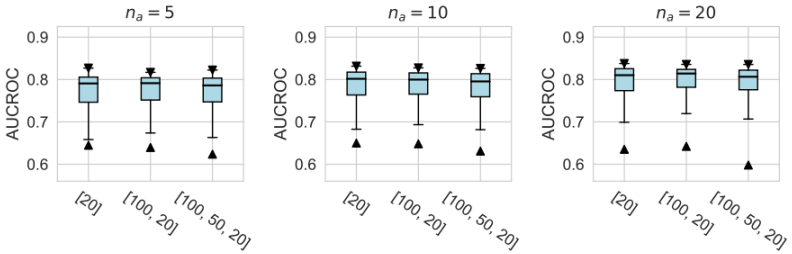
(b) Data preprocessing

- We find that almost no data augmentation method has brought significant performance improvements;
- For data preprocessing methods, We do not observe a significant difference between the minmax scaling and normalizing methods.

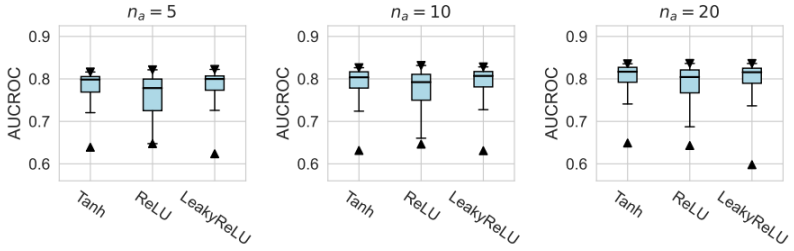
Goal I: Understanding Design Choices of Deep Anomaly Detection



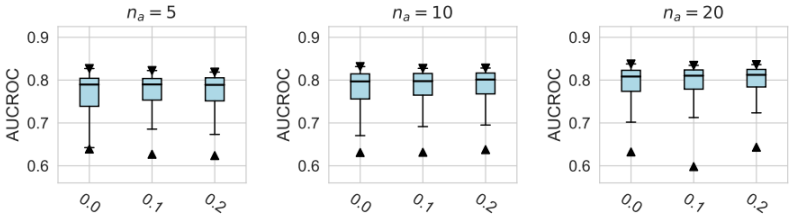
(a) Network architecture



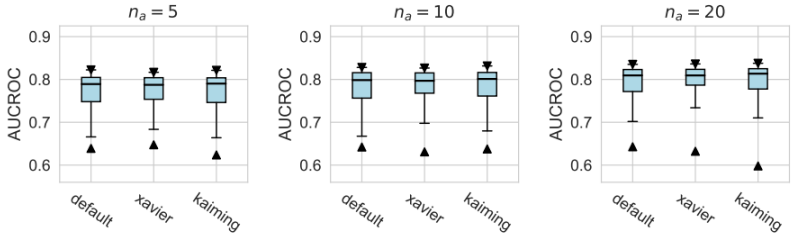
(b) Hidden layers



(c) Activation function



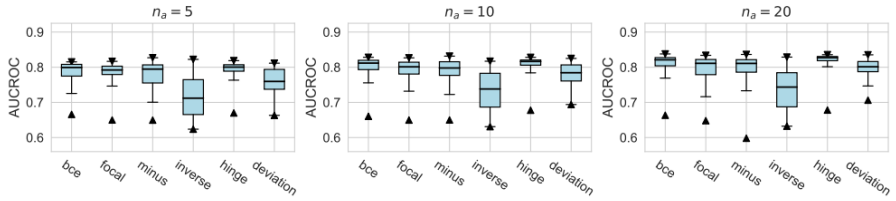
(d) Dropout



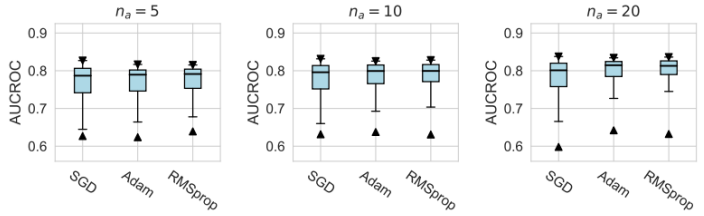
(e) Network initialization

- MLP is still a competitive baseline in AD tasks w.r.t. na = 5;
- ResNet model could also serve as both an effective and stable network architecture;
- No significant advantages of the FTTransformer;
- Tanh and LeakyReLU appear to be more effective than the ReLU function in tabular AD tasks;
- No significant differences in the design dimensions of hidden layers, dropout, and network initialization.

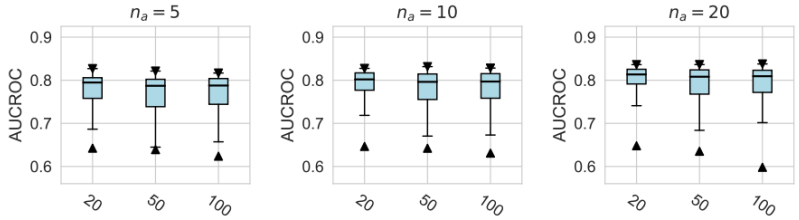
Goal I: Understanding Design Choices of Deep Anomaly Detection



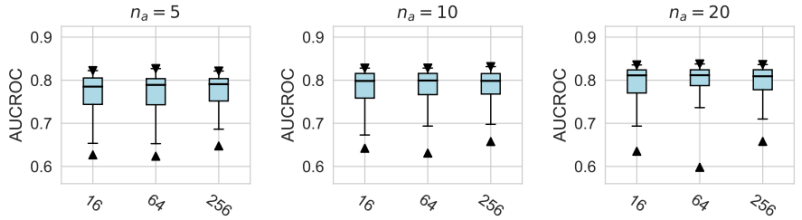
(a) Loss function



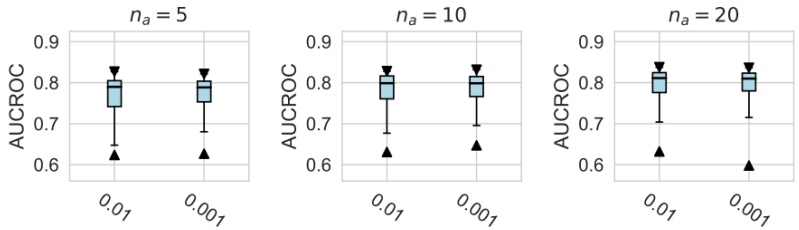
(b) Optimizer



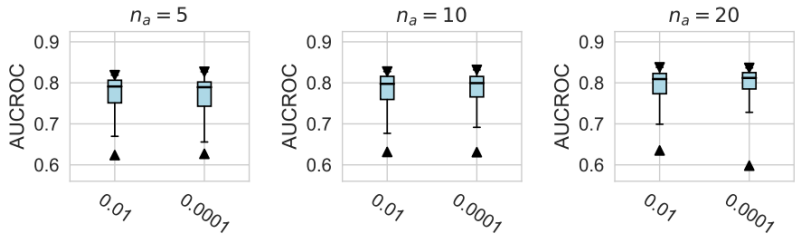
(c) Epochs



(d) Batch size



(e) Learning rate



(f) Weight decay

- The classical BCE loss and hinge loss could be a competitive baseline when batch resampling method is applied in the training process;
- Adam and RMSprop optimizers are better than the classical SGD;
- large training epochs lead to overfitting on limited labeled data;
- No obvious differences for the design dimensions of batch size, learning rate, and weight decay.

Goal II: Constructing AD Algorithms Automatically via ADGym

To predict the performance of a given pipeline on a new dataset, we propose to train a meta-predictor as a **regression problem**:

The input of meta-predictor is $\{E_i^{meta}, E_j^{meta}\}$,corresponding to the meta-feature (i.e., the unified representations of a dataset) of i dataset and the embedding of j AD component (i.e., the representation of a pipeline/components).

Given the meta-predictor $f(\cdot)$, we train it to map **dataset** and **pipeline** characteristics to their corresponding **performance ranking** across all historical datasets, as shown in

$$f \ : \ \underbrace{\mathbf{E}_i^{meta}}_{\text{meta features}} \ , \ \underbrace{\mathbf{E}_j^{comp}}_{\text{component embed.}} \mapsto \boldsymbol{P}_{i,j} \ , \ i \in \{1, \ldots, n\}, \ j \in \{1, \ldots, m\}$$

Goal II: Constructing AD Algorithms Automatically via ADGym

- The automatic construction of AD models is indeed capable of surpassing those SOTA methods, which holds true under various numbers of labeled anomalies;
- The clear advantage of GT indicates that existing AD solutions could be further improved through exploring the design space or automatic selection techniques;
- The meta-predictors generally benefit from a larger design space. More details can be seen in the paper.

Table 1: Baseline of SOTA AD methods.

n_a	GANomaly	REPEN	DeepSAD	DevNet	FEAWAD	ResNet	FTTransformer
5	0.605	**0.784**	0.718	0.761	0.745	0.617	0.782
10	0.616	0.787	0.750	0.792	0.789	0.673	**0.816**
20	0.615	0.801	0.780	0.828	0.829	0.735	**0.843**

Table 2: Performance of auto-selected pipelines by ADGym.

n_a	RS	SS	GT	DL-single		DL-ensemble		ML-single		ML-ensemble	
				2-stage	end2end	2-stage	end2end	XGB	CatB	XGB	CatB
5	0.735	0.613	0.902	0.800	0.811	0.813	0.813	0.802	0.804	0.815	**0.821**
10	0.757	0.696	0.912	0.836	0.833	0.843	0.837	0.836	0.839	**0.849**	0.847
20	0.777	0.747	0.921	0.859	0.850	0.865	0.857	0.857	0.861	**0.869**	0.872

Table 3: Performance of meta-predictors trained on large scale design space.

n_a	RS	SS	GT	DL-single		DL-ensemble		ML-single		ML-ensemble	
				2-stage	end2end	2-stage	end2end	XGB	CatB	XGB	CatB
5	0.738	0.657	0.904	0.824	0.808	**0.829**	0.826	0.814	0.814	0.825	0.825
10	0.767	0.731	0.912	0.842	0.830	**0.853**	0.850	0.843	0.846	0.850	0.851
20	0.791	0.750	0.922	0.863	0.846	**0.876**	0.859	0.860	0.863	0.870	0.874

Future Direction

Based on the experimental results and analysis, we give the several possible future direction for AD community:

- ADGym for Time Series Anomaly Detection;
- ADGym for unsupervised Anomaly Detection;
- Automatic pipeline for ensemble-tree-based techniques.

Anomaly Detection with Score Distribution Discrimination

Minqi Jiang[1], Songqiao Han[1], Hailiang Huang[1]
1 AI Lab, Shanghai University of Finance and Economics

Problem Statement

We propose a novel anomaly detection (AD) loss function for the **weakly-supervised** scenario, where the training data contains:

- Only a handful of labeled anomalies
- Abundant unlabeled instances (unlabeled normal ones & unlabeled anomalies)

Given such limited label information, our goal is to train a model, to effectively assign **higher** anomaly score for the **anomalies** and vice versa.

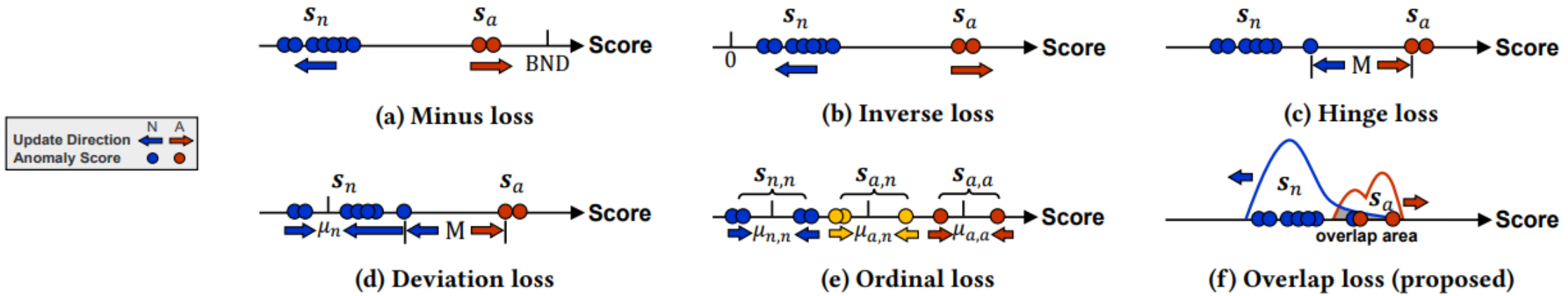
Related Work and Our Insights

In order to achieve this goal, previous AD studies propose following AD loss functions. However, they:

- rely on specific constant or margin hyper-parameter (e.g., M in Hinge loss) to guide model training;
- rely on bound hyper-parameter (e.g., BND in Minus loss) to avoid exploding loss;
- cause the output anomaly score to become excessively large (e.g., Inverse loss) and therefore twist the feature representation space.

Loss	**Publication**	**Formula**	**No Prior**
Minus loss	CCS 2019	$L=\lvert s_n\rvert+\max(0, BND-\lvert s_a\rvert)$	✗
Inverse loss	ICLR 2020	$L=\lvert s_n\rvert+1/\lvert s_a\rvert$	✓
Hinge loss	KDD 2018, KDD 2019	$L=\max(0,M+s_n-s_a)$	✗
Deviation loss	KDD 2019	$L=\lvert s_n\rvert+\max(0,M-s_a)$	✗
Ordinal loss	Arxiv 2020	$L=\lvert s_{n,n}-\mu_{n,n}\rvert+\lvert s_{a,n}-\mu_{a,n}\rvert+\lvert s_{a,a}-\mu_{a,a}\rvert$	✗

Related Work and Our Insights



We propose the Overlap loss to address these issues. Overlap loss *minimizes the overlap area between the score distributions of normal samples and abnormal ones* to realize anomaly score discrimination. Moreover, Overlap loss:

- **does not** rely on prior anomaly score targets;
- **does not** perform excessive optimization on anomaly scores;
- Is **naturally bounded** due to the property of probability density function (PDF);
- optimize from a perspective of anomaly **score distribution**;
- can be effectively instantiated in **various network architectures**.

Overlap loss for Score Distribution Discrimination

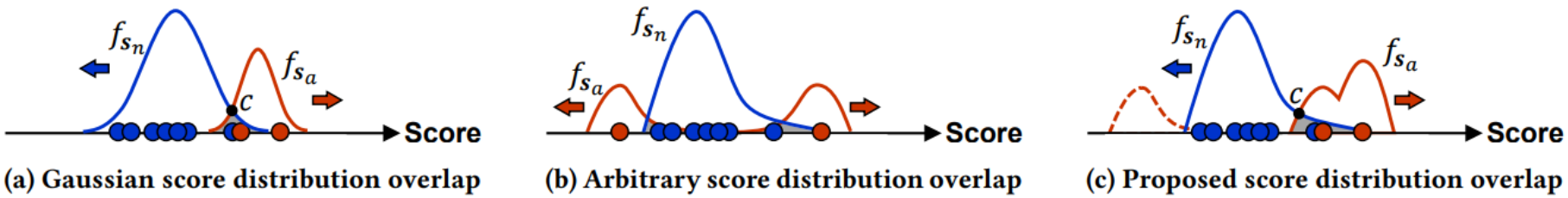
Overlap loss consists of *Score Distribution Estimator* and *Overlap Area Calculation* to:

1. Realize estimation of arbitrary anomaly score distributions;
2. Ensure correct order in output anomaly scores.

On the contrary, compared to the proposed score distribution overlap (c):

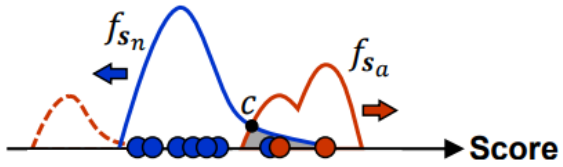
(a) The prior assumption of Gaussian distributions limits the representational ability of neural networks;

(b) Simply minimizing the overlap of arbitrary score distributions may lead to the disorder in anomaly scores.



(a) Gaussian score distribution overlap (b) Arbitrary score distribution overlap (c) Proposed score distribution overlap

Overlap loss for Score Distribution Discrimination



1. estimating the PDF of arbitrary anomaly score distribution via Kernel Density Estimation (KDE);

$$\hat{f}(s) = \lim_{h\to 0} \frac{\hat{F}_N(s+h) - \hat{F}_N(s-h)}{2h} \approx \frac{1}{2Nh} \sum_{i=1}^{N} \left(\mathbf{1}_{s_i \le s+h} - \mathbf{1}_{s_i \le s-h}\right)$$

$$= \frac{1}{2Nh} \sum_{i=1}^{N} \left(\mathbf{1}_{s-h \le s_i \le s+h}\right) = \frac{1}{Nh} \sum_{i=1}^{N} \frac{1}{2} \mathbf{1}\left(\frac{|s - s_i|}{h} \le 1\right) \quad (2)$$

2. acquiring the intersection point c between the two PDFs and approximating their CDF via the trapezoidal rule;

$$\hat{F}_{\boldsymbol{s}}(c) = \int_{-\infty}^{c} \hat{f}_{\boldsymbol{s}}(t)dt = \int_{\min(\boldsymbol{s}_n, \boldsymbol{s}_a)}^{c} \hat{f}_{\boldsymbol{s}}(t)dt \approx \sum_{k=1}^{N} \frac{\hat{f}_{\boldsymbol{s}}(s_k) + \hat{f}_{\boldsymbol{s}}(s_{k+1})}{2} \Delta s_k \quad (8)$$

3. calculating overlap area based on the estimated CDF, which would penalize score disorder (both $P(\boldsymbol{s}_n > c)$ and $P(\boldsymbol{s}_a < c)$ are close to 1);

$$O(\boldsymbol{s}_n, \boldsymbol{s}_a) = P(\boldsymbol{s}_n > c) + P(\boldsymbol{s}_a < c) = 1 - \hat{F}_{\boldsymbol{s}_n}(c) + \hat{F}_{\boldsymbol{s}_a}(c) \quad (6)$$

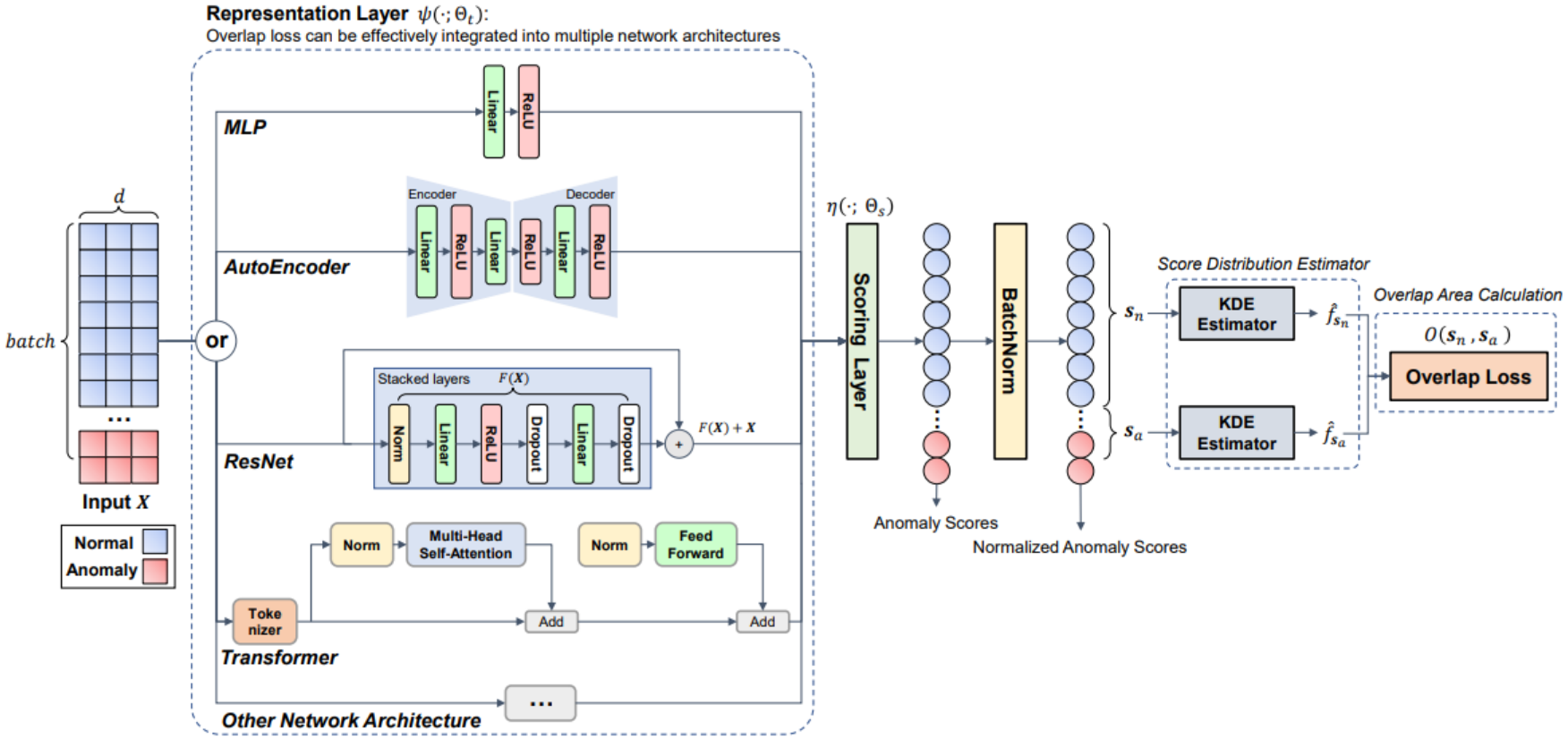
4. Based on the above notations, the Overlap loss is defined as:

$$\mathcal{L}_{\text{Overlap}}(\boldsymbol{x} \mid \Theta) = O(\boldsymbol{s}_n, \boldsymbol{s}_a) =$$

$$1 - \sum_{k=1}^{N} \frac{\hat{f}_{\boldsymbol{s}_n}(s_{n,k}) + \hat{f}_{\boldsymbol{s}_n}(s_{n,k+1})}{2} \Delta s_{n,k} + \sum_{k=1}^{N} \frac{\hat{f}_{\boldsymbol{s}_a}(s_{a,k}) + \hat{f}_{\boldsymbol{s}_a}(s_{a,k+1})}{2} \Delta s_{a,k} \quad (9)$$

Overlap loss for Score Distribution Discrimination

Overlap loss can be effectively integrated into multiple popular network architectures, including widely-used MLP and AutoEncoder in AD tasks, and some cutting-edge ones like ResNet and Transformer.



Experiment Settings

- Datasets: 25 public real-world datasets, including several domains like disease diagnosis, speech recognition, and image identification.
- Metrics: AUC-ROC and AUC-PR. Pairwise Wilcoxon signed rank test is applied to examine the model significance.

Dataset	*N*	*D*	#anomalies	#anomaly ratio (%)
ALOI	49534	27	1508	3.04
annthyroid	7200	6	534	7.42
Cardiotocography	2114	21	466	22.04
fault	1941	27	673	34.67
http	567498	3	2211	0.39
landsat	6435	36	1333	20.71
letter	1600	32	100	6.25
magic.gamma	19020	10	6688	35.16
mammography	11183	6	260	2.32
mnist	7603	100	700	9.21
musk	3062	166	97	3.17
optdigits	5216	64	150	2.88
PageBlocks	5393	10	510	9.46
pendigits	6870	16	156	2.27
satellite	6435	36	2036	31.64
satimage-2	5803	36	71	1.22
shuttle	49097	9	3511	7.15
skin	245057	3	50859	20.75
SpamBase	4207	57	1679	39.91
speech	3686	400	61	1.65
thyroid	3772	6	93	2.47
vowels	1456	12	50	3.43
Waveform	3443	21	100	2.90
Wilt	4819	5	257	5.33
yeast	1484	8	507	34.16

Baselines	Publication	Category
Iforest	ICDM 2008	Unsup
ECOD	TKDE 2022	Unsup
DeepSVDD	ICML 2018	Unsup
GANomaly	ACCV 2018	Semi
DeepSAD	ICLR 2020	Semi
REPEN	KDD 2018	Semi
DevNet	KDD 2019	Semi
PReNet	Arxiv 2020	Semi
FEAWAD	TNNLS 2021	Semi
ResNet	NeurIPS 2021	Sup
FTTransformer	NeurIPS 2021	Sup

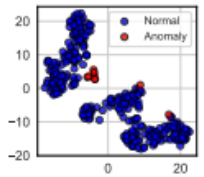
Experiment Results

The average AUC-PR performance over 25 real-world datasets indicating that the proposed Overlap loss is more effective on various network architectures, including MLP, AutoEncoder, ResNet and Transformer w.r.t. different ratios of labeled anomalies $\gamma_l = 5\%, 10\%$ and 20%.

Architecture	Model	Supervision	$\gamma_l = 5\%$		$\gamma_l = 10\%$		$\gamma_l = 20\%$	
			AUC-PR	Δ Perf.	AUC-PR	Δ Perf.	AUC-PR	Δ Perf.
Typical	Iforest	Unsup	0.389±0.295	/	0.389±0.295	/	0.389±0.295	/
	ECOD	Unsup	0.315±0.239	/	0.315±0.239	/	0.315±0.239	/
	DeepSVDD	Unsup	0.147±0.120	/	0.147±0.120	/	0.147±0.120	/
	GANomaly	Semi	0.297±0.191	/	0.296±0.195	/	0.306±0.201	/
	DeepSAD	Semi	0.506±0.253	/	0.601±0.275	/	0.675±0.284	/
	REPEN	Weak	0.560±0.300	/	0.603±0.308	/	0.639±0.306	/
MLP	DevNet	Weak	0.606±0.311	+2.89%	0.626±0.307	+7.75%**	0.652±0.305	+6.74%*
	PReNet	Weak	0.612±0.305	+1.82%	0.638±0.307	+5.67%*	0.660±0.303	+5.49%
	MLP-Overlap (ours)	Weak	**0.623±0.291**	/	**0.674±0.286**	/	**0.696±0.288**	/
AutoEncoder	FEAWAD	Sup	0.509±0.269	+28.04%***	0.620±0.270	+12.05%***	0.678±0.270	+5.17%**
	FEAWAD	Weak	0.596±0.286	+9.29%***	0.645±0.293	+7.71%***	0.682±0.283	+4.56%**
	AE-Overlap (ours)	Weak	**0.652±0.290**	/	**0.695±0.294**	/	**0.713±0.296**	/
ResNet	ResNet	Sup	0.401±0.241	+56.30%***	0.483±0.224	+44.81%***	0.598±0.235	+23.92%***
	ResNet-Overlap (ours)	Weak	**0.627±0.297**	/	**0.699±0.289**	/	**0.742±0.283**	/
Transformer	FTTransformer	Sup	0.594±0.299	+5.50%*	0.644±0.308	+6.61%*	0.691±0.305	+5.65%*
	FTTransformer-Overlap (ours)	Weak	**0.627±0.277**	/	**0.686±0.282**	/	**0.730±0.285**	/

Further Analysis of Overlap Loss

Since Overlap loss only needs to minimizes the overlap area of score distributions between normal samples and anomalies, it avoids unnecessary further updates of anomaly scores. Therefore Overlap loss:

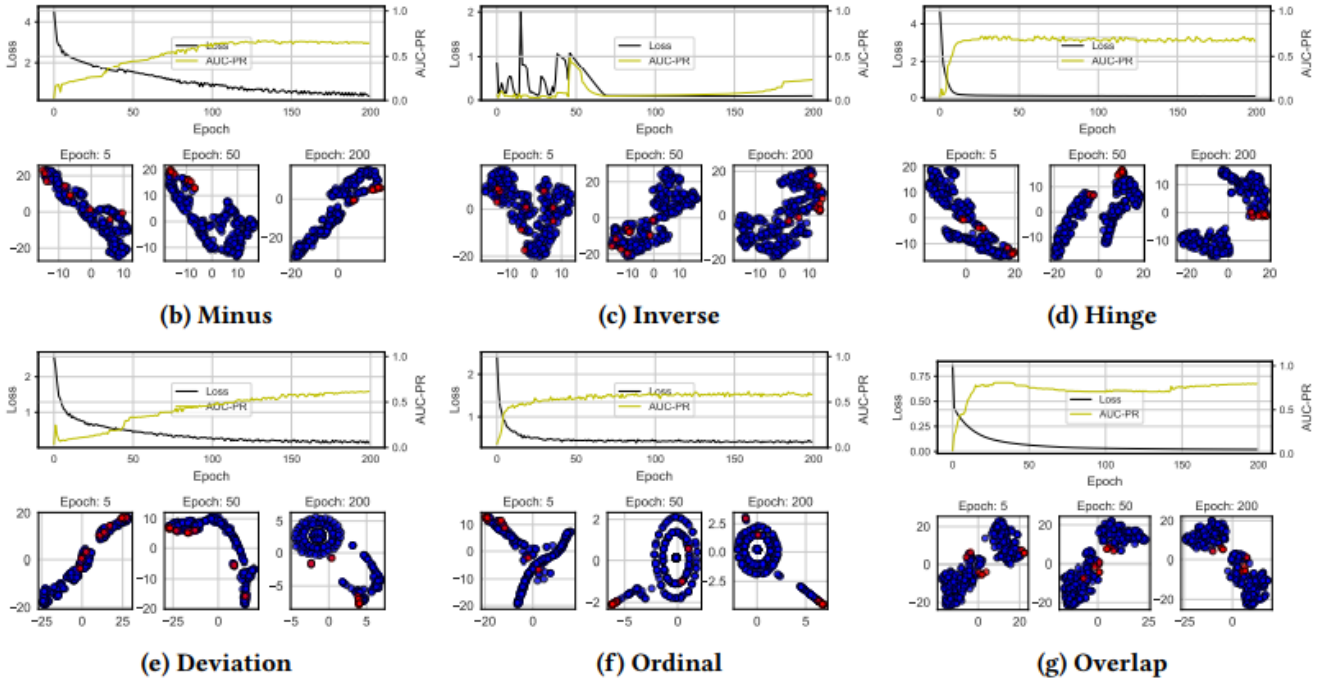


(a) t-SNE [60] plots of the input feature of vowels dataset.

- remains more fine-grained information of input data, generating mild transformation in feature representation;

- generates minor network parameter changes during model training, where drastic changes in network parameters could suffer from the problem of catastrophic forgetting.

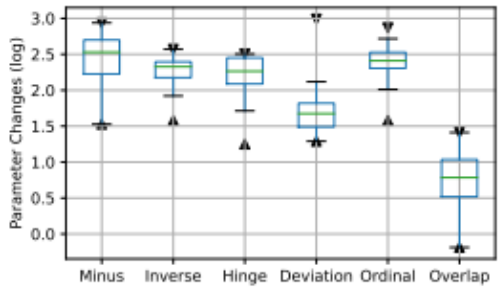
Embedding transformations during model training



(b) Minus (c) Inverse (d) Hinge

(e) Deviation (f) Ordinal (g) Overlap

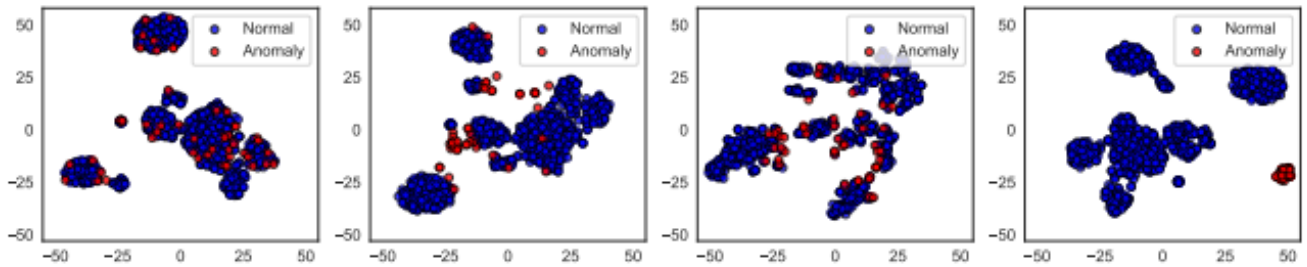
Network parameter changes during model training



Further Analysis of Overlap Loss

‼ Besides, while extensive AD methods have been proven to be effective on real-world datasets, they often neglect to discuss AD methods regarding **specific types of anomalies**.

We follow our previous work ADBench to create and evaluate different AD loss functions on four types of anomalies. Both AUC-ROC and AUC-PR results indicate that the proposed **Overlap loss is more effective when detecting various types of anomalies**.



(a) Local (b) Global (c) Dependency (d) Clustered

AUC-ROC Results

Loss	Local	Global	Clustered	Dependency
Minus	0.255	0.822	0.992	0.369
Inverse	0.235	0.647	0.900	0.198
Hinge	0.271	0.853	0.996	0.413
Deviation	0.246	0.851	0.987	0.303
Ordinal	0.247	0.849	0.991	0.327
Overlap	**0.439**	**0.929**	**0.998**	**0.571**

AUC-PR Results

Loss	Local	Global	Clustered	Dependency
Minus	0.629	0.936	0.996	0.738
Inverse	0.547	0.823	0.937	0.570
Hinge	0.607	0.938	0.997	0.761
Deviation	0.588	0.959	0.990	0.652
Ordinal	0.604	0.954	0.994	0.687
Overlap	**0.742**	**0.981**	**0.998**	**0.847**

Conclusions

We propose a novel AD loss function called Overlap loss, which:

- ✔ liberates AD methods from predefined anomaly score targets (e.g., constant or margin hyper-parameter(s));
- ✔ optimizes anomaly scores from the perspective of distribution and is naturally bounded;
- ✔ can be effectively instantiated in various network architectures, outperforming its counterpart AD losses;
- ✔ remains fine-grained representational information and generates minor network parameter changes;
- ✔ performs better for different types of anomalies.

Future Work and Limitations

🤔 Utilizing more complex techniques like Gaussian Mixture Model (GMM) to improve the estimation process of anomaly score distributions;

🤔 Extending the data scenarios to unsupervised AD tasks, or more general scenarios in weakly-supervised AD like inaccurate and inexact supervisions;

🤔 Verifying the effectiveness of the proposed Overlap loss in other data modalities like NLP and CV.