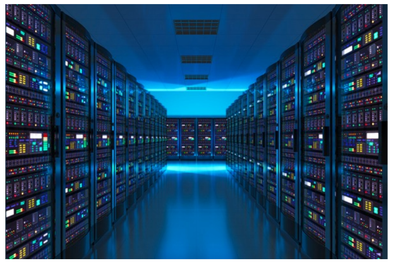
DCdetector: Dual Attention Contrastive Representation Learning for Time Series Anomaly Detection

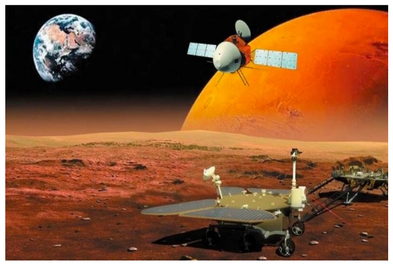
(KDD 2023 research track)

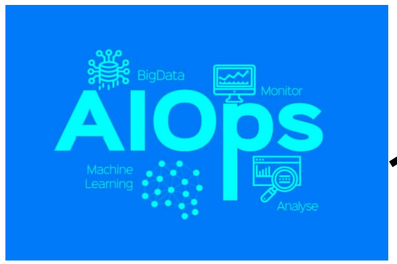
Yiyuan Yang[1], Chaoli Zhang[2], Tian Zhou[2], Qingsong Wen[2], Liang Sun[2]
[1]Department of Computer Science, University of Oxford, [2]DAMO Academy, Alibaba Group
yiyuan.yang@cs.ox.ac.uk, {chaoli.zcl, tian.zt, qingsong.wen, liang.sun}@alibaba-inc.com

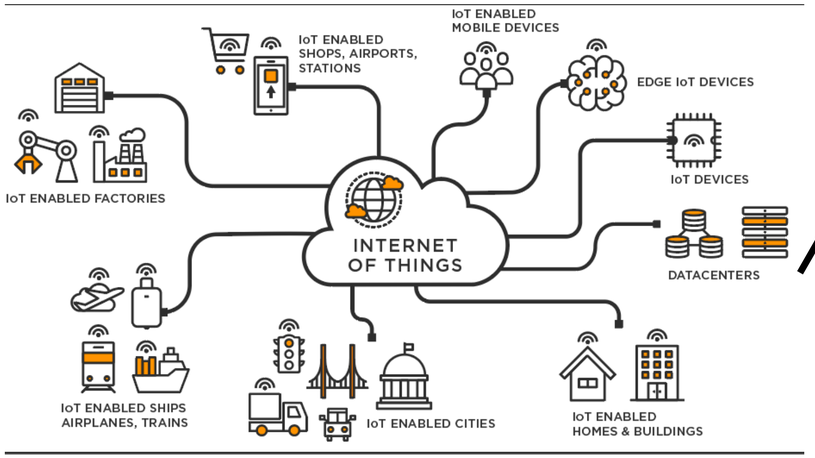
https://arxiv.org/abs/2306.10347
https://github.com/DAMO-DI-ML/KDD2023-DCdetector

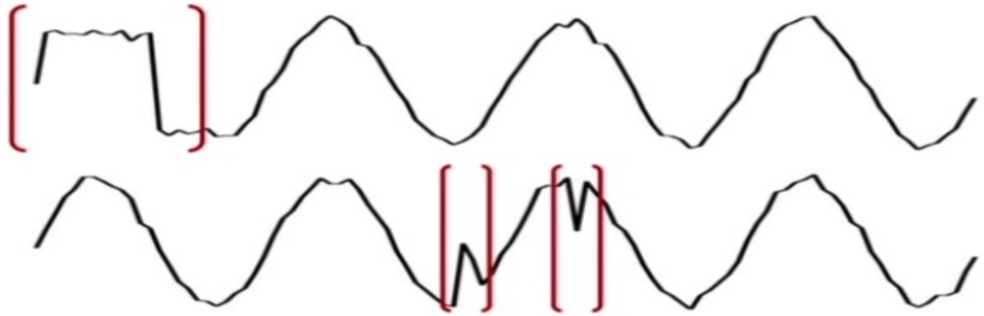
✓ Background: Time Series Anomaly Detection











Extracting useful information from the time-series to ensure security, avoid financial loss and social stability.

Detecting the abnormal time-step from the original time-series.

✓ Challenges: Time Series Anomaly Detection

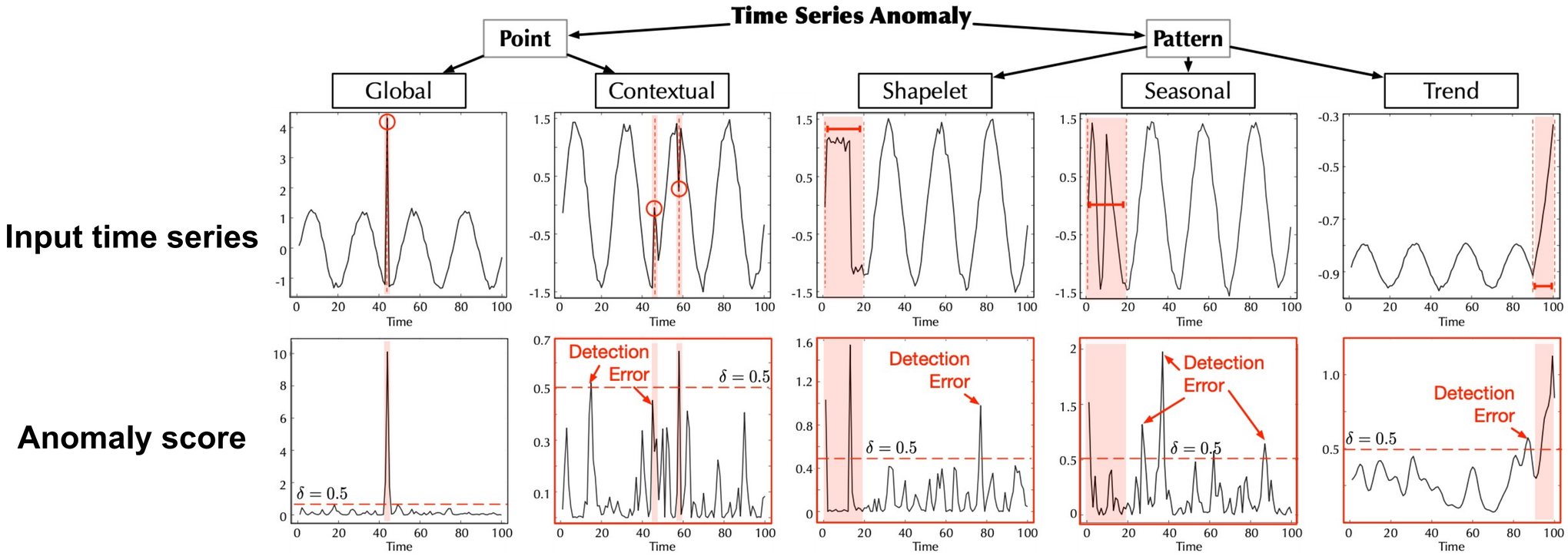
- **Lack of Labeled Data**: Anomalies are usually rare without many labels. Systems are in steady state in majority of cases.
- **Imbalance**: The number of anomalies is much smaller than the normal one. For example, in the financial system.
- **Noise Interference**: Time-series data may be affected by noise that may mask the true anomaly signal.
- **Complex Patterns**: Typical anomalies are often complex (e.g., wind turbines operate in different modes and conditions).
- **Multi-dimensional Features**: Models should consider temporal, multidimensional and non-stationary characteristics.
- **Explanatory and Interpretable**: In some application scenarios, explanatory and interpretable results for anomaly detection are needed to better understand why an anomaly was flagged and to be able to take action accordingly.

Anomaly detection based on unsupervised learning (learning without labeled data)

time series data → Per timestamp feature mining → Anomaly detection, evaluation at each timestamp

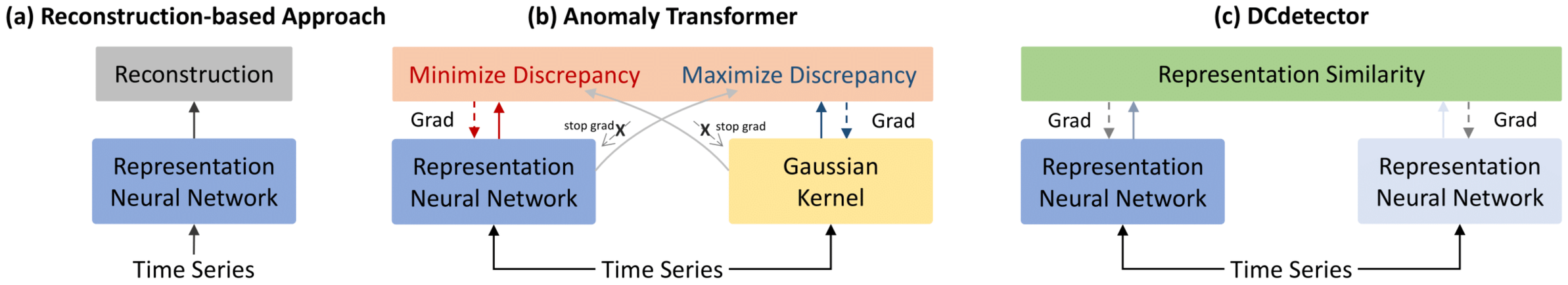
✓ Challenges: Time Series Anomaly Detection

Numerous Complex Patterns and Hard to Detect



✓ Related work: DCdetector vs. previous SOTA methods

Reconstructed-based problem: The raw time series has a mixture of normalities and anomalies with noise. So it is difficult to train a high quality encoder for reconstruction based models.



$$L_{rec} = \left\| x - \hat{x} \right\|_2^2$$

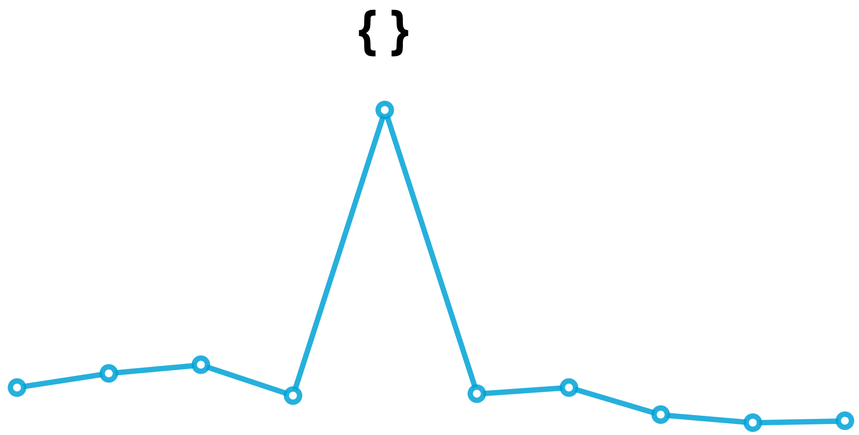
$$L_{AT} = \left\| x - \hat{x} \right\|_2^2 - \lambda \times \| AssDis(P, S; x) \|_1$$

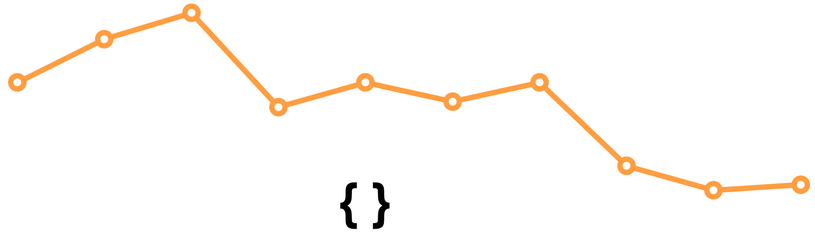
$$\boldsymbol{L_{DCdetector} = KL(R_1(x) \,||\, R_2(x))}$$

Highlight: DCdetector Conducts Time Series Anomaly Detection without Reconstruction

✓ Method: DCdetector's framework

An intuition: Normal points are closely related to other sample points, while abnormal points are discrete from other sample points. By constructing different representations (Patch-wise and In-patch) between the sample points, if the similarity of the different representations is high, it means that they are normal points.





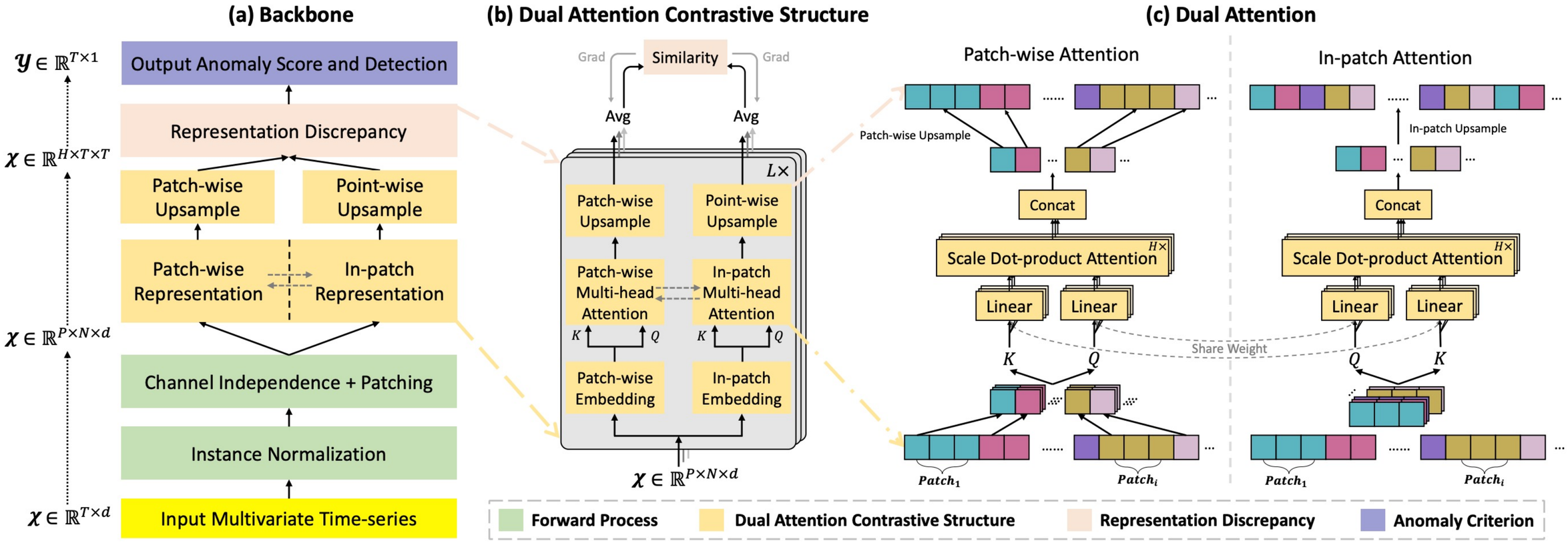
Anomaly: Weak relation between adjacent data

Normal point: Strong relation between adjacent data

Differentiation by similarity of the different representations

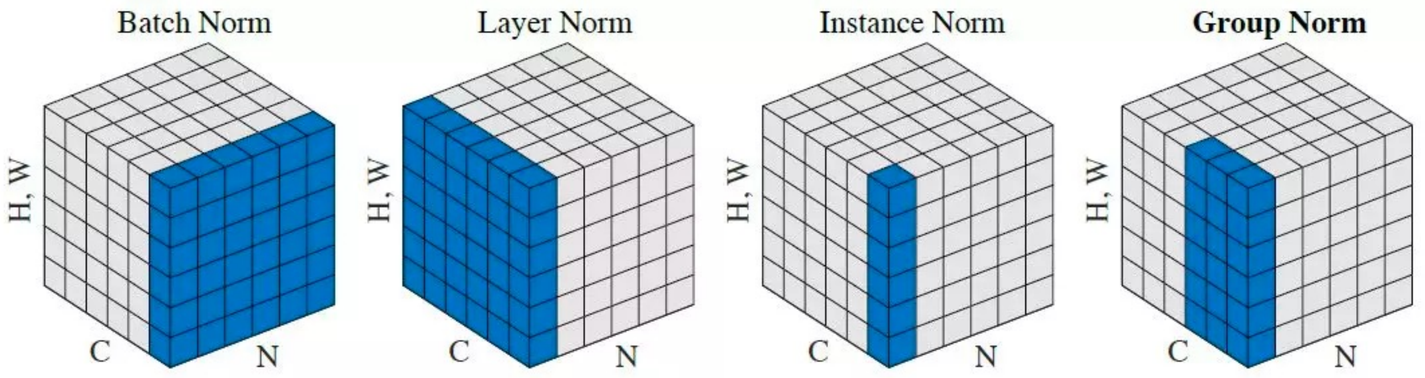
✓ Method: DCdetector's framework

An intuition: Normal points are closely related to other sample points, while abnormal points are discrete from other sample points. By constructing different representations (Patch-wise and In-patch) between the sample points, if the similarity of the different representations is high, it means that they are normal points.

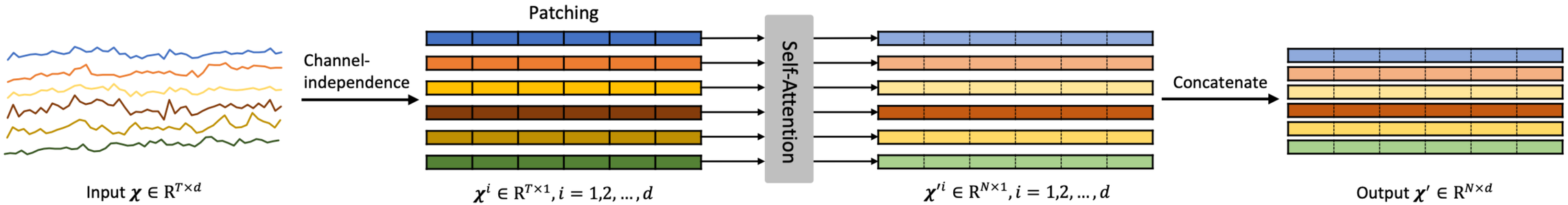


✓ Method: DCdetector's framework - Forward Process

Instance normalisation (tackle non-stationarity problem)

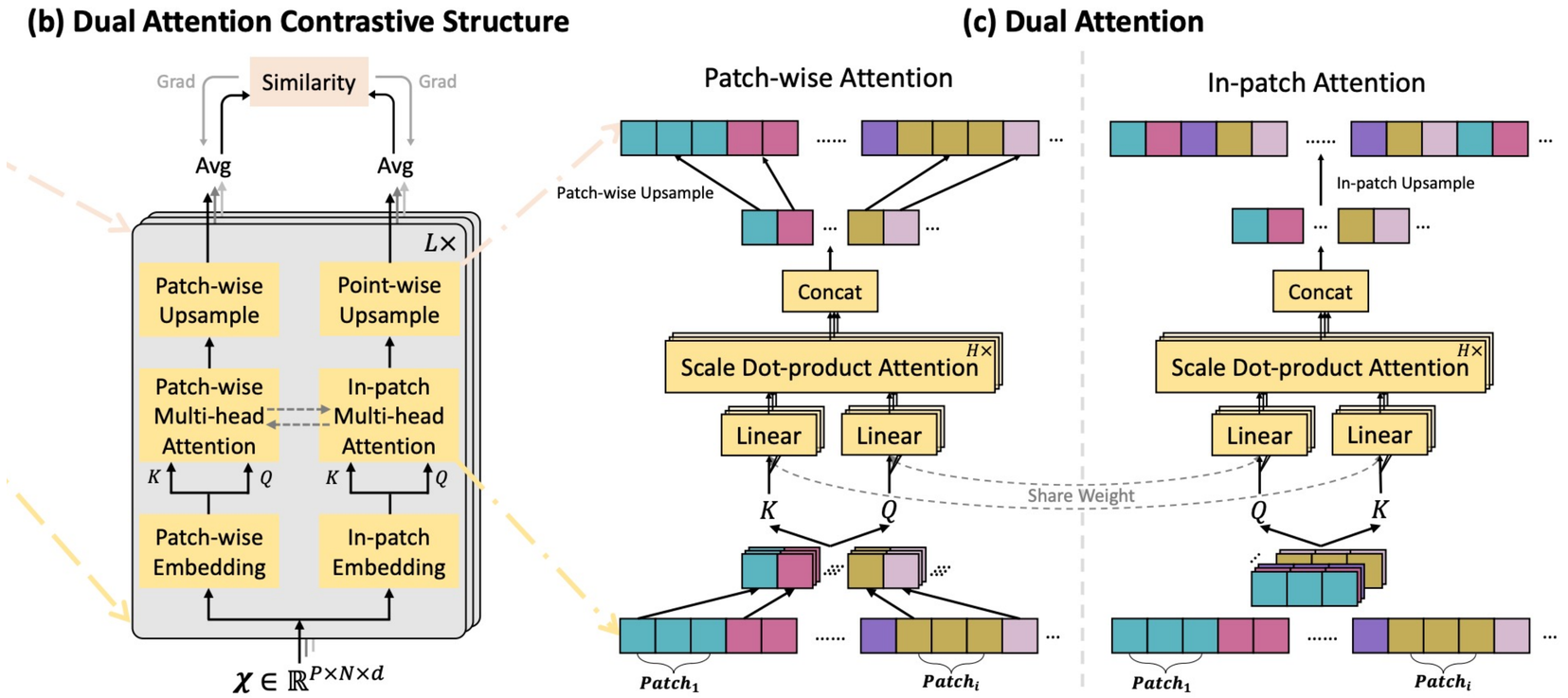


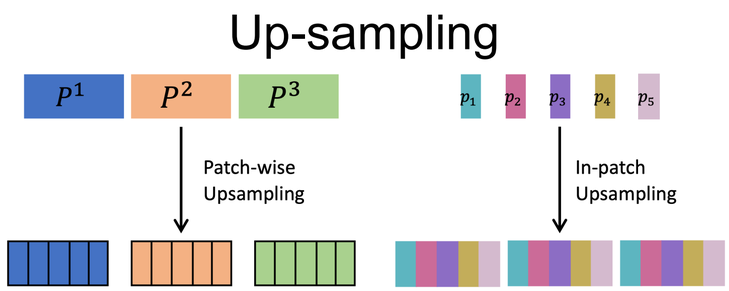
Patching and channel independence (tackle multidimensional problem)



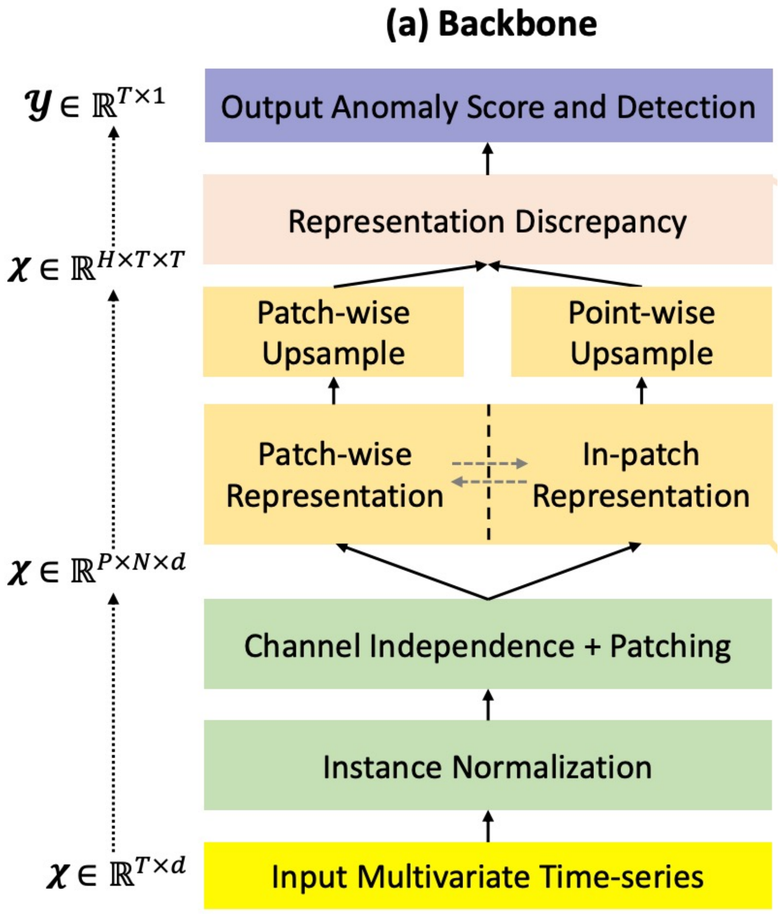
✓ Method: DCdetector's framework - Comparative Learning Architecture

- **Patch-wise representation**: Attention scores between different patches.
- **In-patch representation**: Attention scores of internal patch.





✓ Method: DCdetector's framework - Representation Discrepancy and Anomaly Evaluation



$$L_{DCdetector} = KL(R_1(x) \,||\, R_2(x))$$

$$\text{AnomalyScore}(X) = \frac{1}{2}\mathcal{D}(\mathcal{P}, \mathcal{N}) + \frac{1}{2}\mathcal{D}(\mathcal{N}, \mathcal{P}).$$

$$y_i = \begin{cases} 1\text{: anomaly} & \text{AnomalyScore}(X_i) \geq \delta \\ 0\text{: normal} & \text{AnomalyScore}(X_i) < \delta \end{cases}$$

✓ Evaluation: Datasets and Implement (all open-sourced)

- Datasets & Baselines
  6+1 benchmarks, **26** baselines
- Evaluation Criteria
  10 metrics (F1, Affiliation, VUS)
- Performance on Parameter Sensitivity
- Visual Analysis
- Time-Cost and Memory Used

- One NVIDIA Tesla-V100 32GB GPU
- batch size to 128
- 3 epochs
- other hyper-parameters…
- Inference time less than 1s for all datasets
- All test scripts open source

Table 8: Details of benchmark datasets. AR (anomaly ratio) represents the abnormal proportion of the whole dataset.

Benchmark	Source	Dimension	Window	Patch Size	#Training	#Test (Labeled)	AR (%)
MSL	NASA Space Sensors	55	90	[3,5]	58,317	73,729	10.5
SMAP	NASA Space Sensors	25	105	[3,5,7]	135,183	427,617	12.8
PSM	eBay Server Machine	25	60	[1,3,5]	132,481	87,841	27.8
SMD	Internet Server Machine	38	105	[5,7]	708,405	708,420	4.2
NIPS-TS-SWAN	Space (Solar) Weather	38	36	[1,3]	60,000	60,000	32.6
NIPS-TS-GECCO	Water Quality for IoT	9	90	[1,3,5]	69,260	69,261	1.1
UCR	Various Natural Sources	1	105	[3,5,7]	2,238,349	6,143,541	0.6

✓ Evaluation: Multivariate Dataset Results

Dataset	MSL			SMAP			PSM			SMD		
Metric	P	R	F1	P	R	F1	P	R	F1	P	R	F1
LOF	47.72	85.25	61.18	58.93	56.33	57.60	57.89	90.49	70.61	56.34	39.86	46.68
OCSVM	59.78	86.87	70.82	53.85	59.07	56.34	62.75	80.89	70.67	44.34	76.72	56.19
U-Time	57.20	71.66	63.62	49.71	56.18	52.75	82.85	79.34	81.06	65.95	74.75	70.07
IForest	53.94	86.54	66.45	52.39	59.07	55.53	76.09	92.45	83.48	42.31	73.29	53.64
DAGMM	89.60	63.93	74.62	86.45	56.73	68.51	93.49	70.03	80.08	67.30	49.89	57.30
ITAD	69.44	84.09	76.07	82.42	66.89	73.85	72.80	64.02	68.13	86.22	73.71	79.48
VAR	74.68	81.42	77.90	81.38	53.88	64.83	90.71	83.82	87.13	78.35	70.26	74.08
MMPCACD	81.42	61.31	69.95	88.61	75.84	81.73	76.26	78.35	77.29	71.20	79.28	75.02
CL-MPPCA	73.71	88.54	80.44	86.13	63.16	72.88	56.02	**99.93**	71.80	82.36	76.07	79.09
TS-CP2	86.45	68.48	76.42	87.65	83.18	85.36	82.67	78.16	80.35	87.42	66.25	75.38
Deep-SVDD	91.92	76.63	83.58	89.93	56.02	69.04	95.41	86.49	90.73	78.54	79.67	79.10
BOCPD	80.32	87.20	83.62	84.65	85.85	85.24	80.22	75.33	77.70	70.9	82.04	76.07
LSTM-VAE	85.49	79.94	82.62	92.20	67.75	78.10	73.62	89.92	80.96	75.76	90.08	82.30
BeatGAN	89.75	85.42	87.53	92.38	55.85	69.61	90.30	93.84	92.04	72.90	84.09	78.10
LSTM	85.45	82.50	83.95	89.41	78.13	83.39	76.93	89.64	82.80	78.55	85.28	81.78
OmniAnomaly	89.02	86.37	87.67	92.49	81.99	86.92	88.39	74.46	80.83	83.68	86.82	85.22
InterFusion	81.28	92.70	86.62	89.77	88.52	89.14	83.61	83.45	83.52	87.02	85.43	86.22
THOC	88.45	90.97	89.69	92.06	89.34	90.68	88.14	90.99	89.54	79.76	90.95	84.99
AnomalyTrans	91.92	96.03	93.93	93.59	**99.41**	96.41	96.94	97.81	97.37	**88.47**	**92.28**	**90.33**
DCdetector	**93.69**	**99.69**	**96.60**	**95.63**	98.92	**97.02**	**97.14**	98.74	**97.94**	83.59	91.10	87.18

✓ Evaluation: Other Dataset and Evaluation Criteria Results

Table 4: Multi-metrics results on NIPS-TS datasets. All results are in %, and the best ones are in Bold.

Dataset	Method	Acc	P	R	F1	Aff-P	Aff-R	R_A_R	R_A_P	V_ROC	V_PR
NIPS-TS-SWAN	AnomalyTrans	84.57	90.71	47.43	62.29	**58.45**	**9.49**	86.42	93.26	84.81	92.00
	DCdetector	**85.94**	**95.48**	**59.55**	**73.35**	50.48	5.63	**88.06**	**94.71**	**86.25**	**93.50**
NIPS-TS-GECCO	AnomalyTrans	98.03	25.65	28.48	26.99	49.23	81.20	56.35	22.53	55.45	21.71
	DCdetector	**98.56**	**38.25**	**59.73**	**46.63**	**50.05**	**88.55**	**62.95**	**34.17**	**62.41**	**33.67**

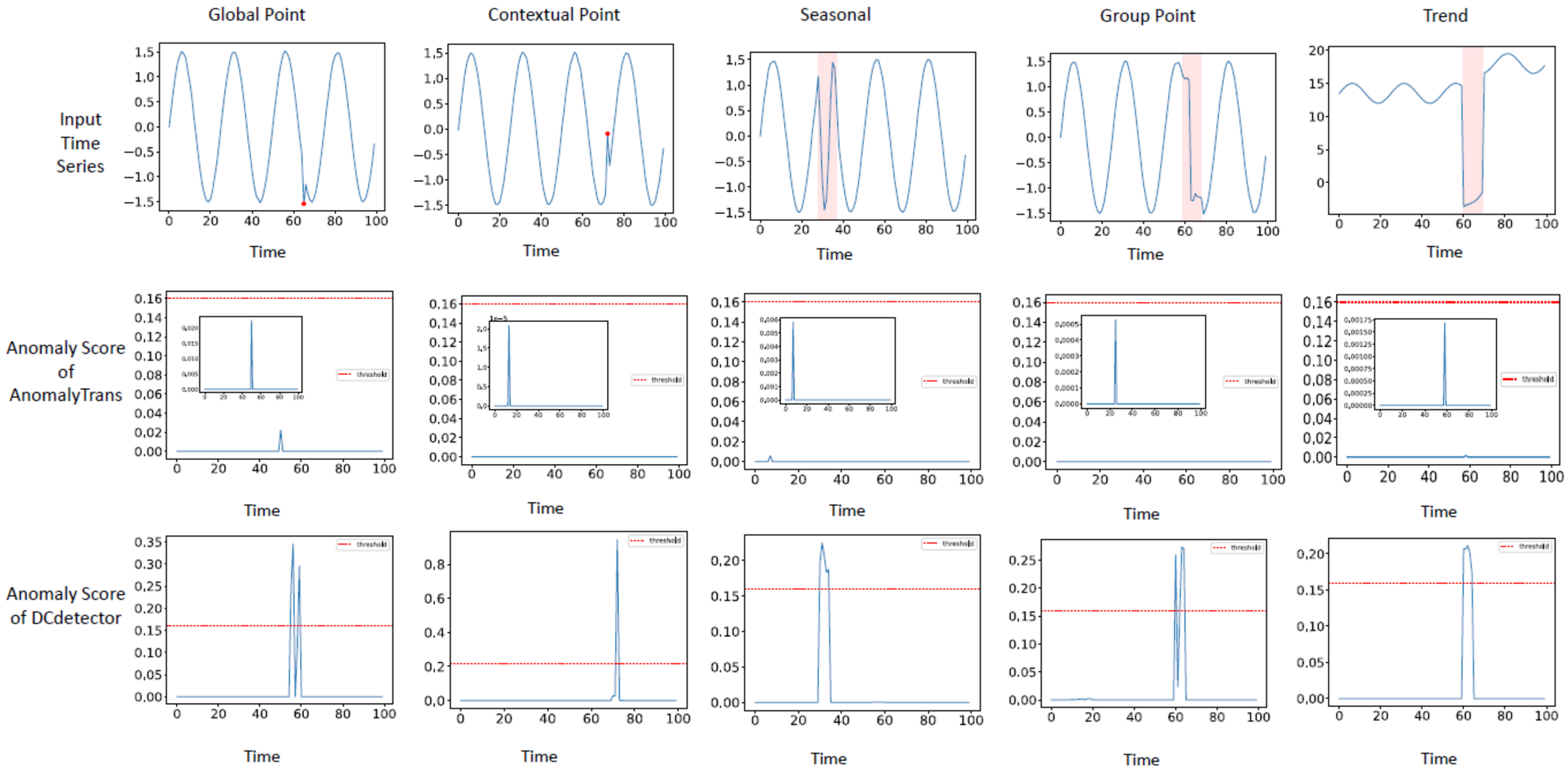
Overall results on NIPS-TS datasets.

Dataset	NIPS-TS-GECCO			NIPS-TS-SWAN		
Metric	P	R	F1	P	R	F1
OCSVM*	2.1	34.1	4.0	19.3	0.1	0.1
MatrixProfile	4.6	18.5	7.4	16.7	17.5	17.1
GBRT	17.5	14.0	15.6	44.7	37.5	40.8
LSTM-RNN	34.3	27.5	30.5	52.7	22.1	31.2
Autoregression	39.2	31.4	34.9	42.1	35.4	38.5
OCSVM	18.5	**74.3**	29.6	47.4	49.8	48.5
IForest*	39.2	31.5	39.0	40.6	42.5	41.6
AutoEncoder	42.4	34.0	37.7	49.7	52.2	50.9
AnomalyTrans	25.7	28.5	27.0	90.7	47.4	62.3
IForest	**43.9**	35.3	39.1	56.9	**59.8**	58.3
DCdetector	38.3	59.7	**46.6**	**95.5**	59.6	**73.4**

Overall results on univariate dataset.

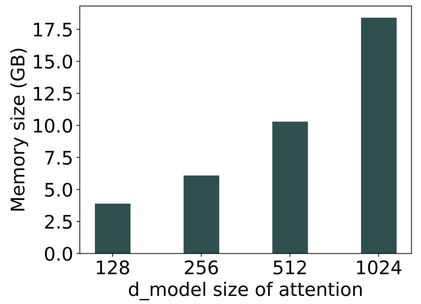
Dataset	UCR				
Metric	Acc	P	R	F1	Count
AnomalyTrans	99.49	60.41	**100**	73.08	42
DCdetector	**99.51**	**61.62**	**100**	**74.05**	**46**

✓ Visualization: Anomaly Scores and Comparison of Different Anomaly Types

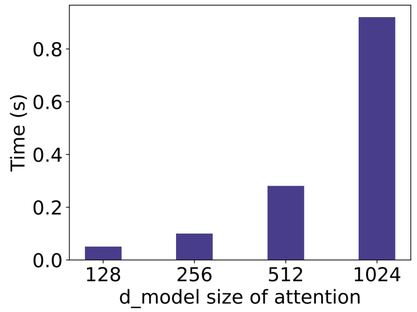


✓ Ablation Experiments and Parameter Sensitivity

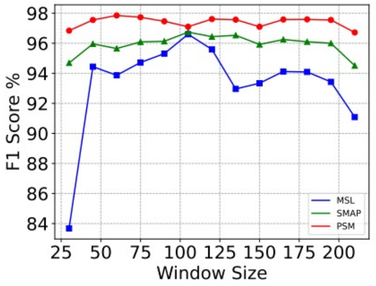
- Window size
- Multi-scale size (patch size)
- Encoder layer number & Attention head number
- Two branches and single branch
- Anomaly threshold
- Loss function and Forward process
- Other model's parameter



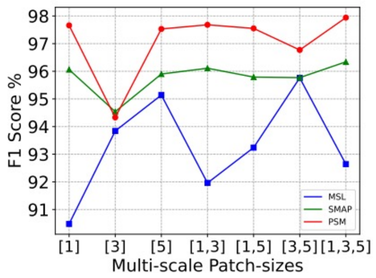
(a) Memory used



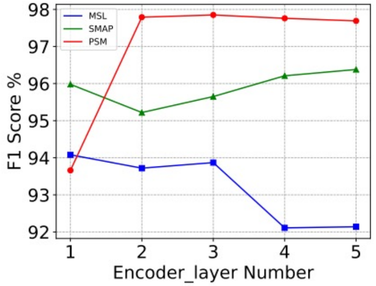
(b) Time cost



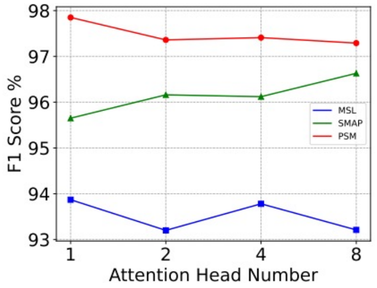
(a) Window size



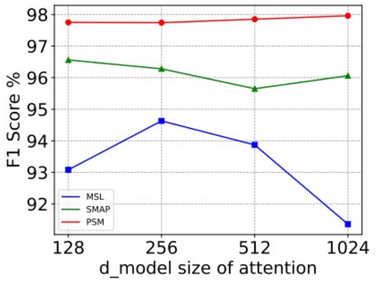
(b) Multi-scale size



(c) Encoder layer number



(d) Attention head number



(e) d_{model} of attention

✓ Conclusion of DCdetector

❖ **Architecture**:

- A contrastive learning-based dual-branch attention structure
- Channel independence patching is proposed to enhance local semantic information
- Multi-scale structure in the attention module can reduce information loss during patching

❖ **Optimization:**

- An effective and robust loss function is designed based on the similarity of two branches
- Model is trained purely contrastively without reconstruction loss, reducing distractions from anomalies

❖ **Performance:**

- DCdetector achieves SOTA performance in 7 benchmarks with 10 metrics, compared with 26 baselines

✓ Open-sourced Codes

https://github.com/DAMO-DI-ML/KDD2023-DCdetector

Get Start

1. Install Python 3.6, PyTorch >= 1.4.0.
2. Download data. You can obtain all benchmarks from Google Cloud. All the datasets are well pre-processed.
3. Train and evaluate. We provide the experiment scripts of all benchmarks under the folder `./scripts`. You can reproduce the experiment results as follows:

```
bash ./scripts/SMD.sh
bash ./scripts/MSL.sh
bash ./scripts/SMAP.sh
bash ./scripts/PSM.sh
bash ./scripts/NIPS_TS_Swan.sh
bash ./scripts/NIPS_TS_Water.sh
bash ./scripts/UCR.sh
```

Also, some scripts of ablation experiments.

```
bash ./scripts/Ablation_attention_head.sh
bash ./scripts/Ablation_encoder_layer.sh
bash ./scripts/Ablation_Multiscale.sh
bash ./scripts/Ablation_Window_Size.sh
```

Code Description

There are ten files/folders in the source.

- data_factory: The preprocessing folder/file. All datasets preprocessing codes are here.
- dataset: The dataset folder, and you can download all datasets here.
- main.py: The main python file. You can adjustment all parameters in there.
- metrics: There is the evaluation metrics code folder, which includes VUC, affiliation precision/recall pair, and other common metrics. The details can be corresponding to paper's Section 4.2.
- model: DCdetector model folder. The details can be corresponding to paper's Section 3.
- result: In our code demo, we can automatically save the results and train processing log in this folder.
- scripts: All datasets and ablation experiments scripts. You can reproduce the experiment results as get start shown.
- solver.py: Another python file. The training, validation, and testing processing are all in there.
- utils: Other functions for data processing and model building.
- img: Images needed in readme.md.

Thanks and Q&A

https://arxiv.org/abs/2306.10347

https://github.com/DAMO-DI-ML/KDD2023-DCdetector