



TS2Vec: Towards Universal Representation of Time Series

TS2Vec: 时间序列的通用表示

主要内容

Main Contents

1

摘要

2

背景

3

方法

4

实验

5

分析

6

结论



摘要



- TS2Vec: 一个在任意语义层学习时间序列表示的通用框架; 在增强的上下文视图上以分层的方式执行对比学习
- TS2Vec 在 125 个 UCR 数据集和 29 个 UEA 数据集上执行时间序列预测和异常检测任务上改进明显



背景



流行的研究方法：

- 实例级表示：描述了输入时间序列的整个部分，并在聚类和分类等任务中显示出巨大的成功。
- 对比损失：学习时间序列的内在结构。

现有的局限性：

细粒度、不同粒度的多尺度上下文信息。

现有的无监督时间序列表示方法大多受 CV 和 NLP 领域经验的启发，具有很强的归纳偏向性，如变换不变性和裁剪不变性。

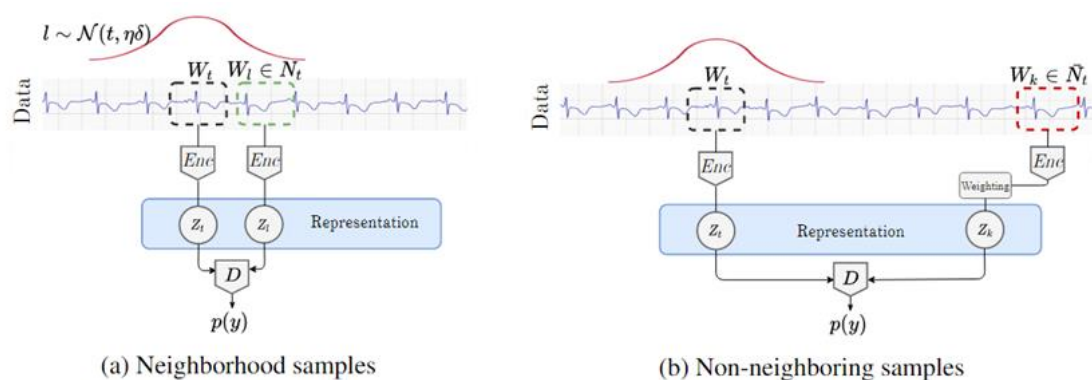
细粒度：比如预测、异常检测

多尺度：比如实例级表示没有时间对比这个维度

裁剪的子序列很可能具有与原始时间序列不同的分布

Tonekaboni, Eytan, and Goldenberg 2021: **Unsupervised Representation Learning for Time Series with Temporal Neighborhood Coding**

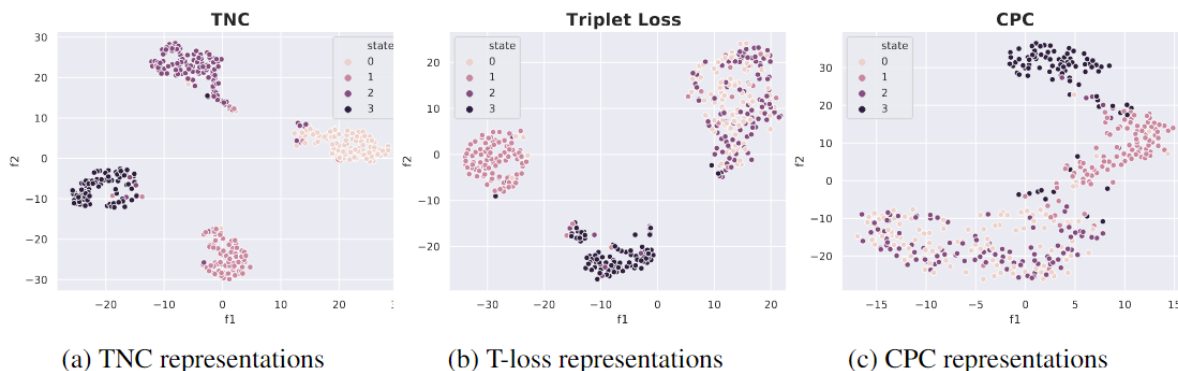
Temporal Neighborhood Coding (TNC), 利用局部平滑度来定义具有平稳属性的时间的邻域, 确保邻域内的信号分布与非邻域信号分布可区分。



TNC框架

正、负样本

来自 \bar{N}_t 未标记的样本



聚类：模拟数据集的信号表示的 **T-SNE 可视化**。图中的每个数据点呈现一个大小为 $\delta = 50$ 的时间序列的 10 维表示。

T-Loss即Triplet Loss

对比预测编码 (CPC, Contrastive Predictive Coding)

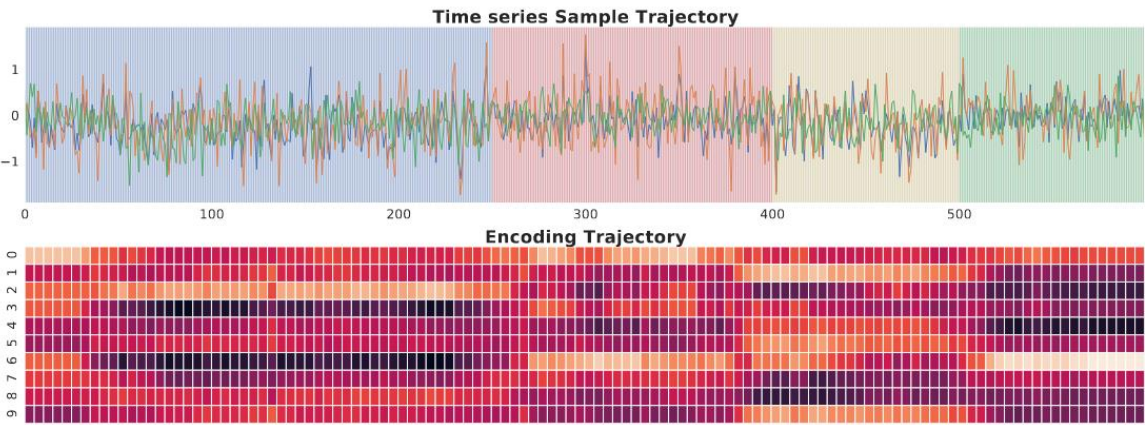
Method	Simulation		ECG Waveform		HAR	
	AUPRC	Accuracy	AUPRC	Accuracy	AUPRC	Accuracy
TNC	0.99±0.00	97.52±0.13	0.55±0.01	77.79±0.84	0.94±0.007	88.32±0.12
CPC	0.69±0.06	70.26±6.48	0.42±0.01	68.64±0.49	0.93±0.006	86.43±1.41
T-Loss	0.78±0.01	76.66±1.40	0.47±0.00	75.51±1.26	0.71±0.007	63.60±3.37
KNN	0.42±0.00	55.53±0.65	0.38±0.06	54.76±5.46	0.75±0.01	84.85±0.84
Supervised	0.99±0.00	98.56±0.13	0.67±0.01	94.81±0.28	0.98±0.00	92.03±2.48

分类

预测准确度和准确回忆曲线下面积（AUPRC）分数

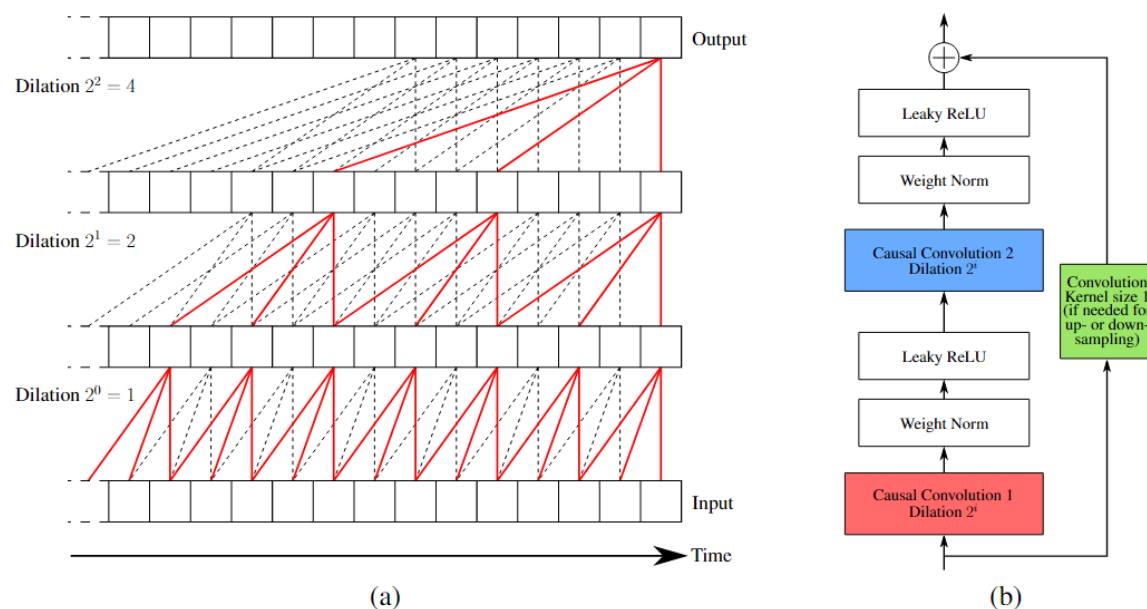
Triplet Loss从时间序列的重叠窗口中采样了积极的例子，无法学习更多的通用表示。

CPC和Triplet Loss没有明确说明当随机选择的负示例与参考Wt相似时发生的潜在采样偏差。

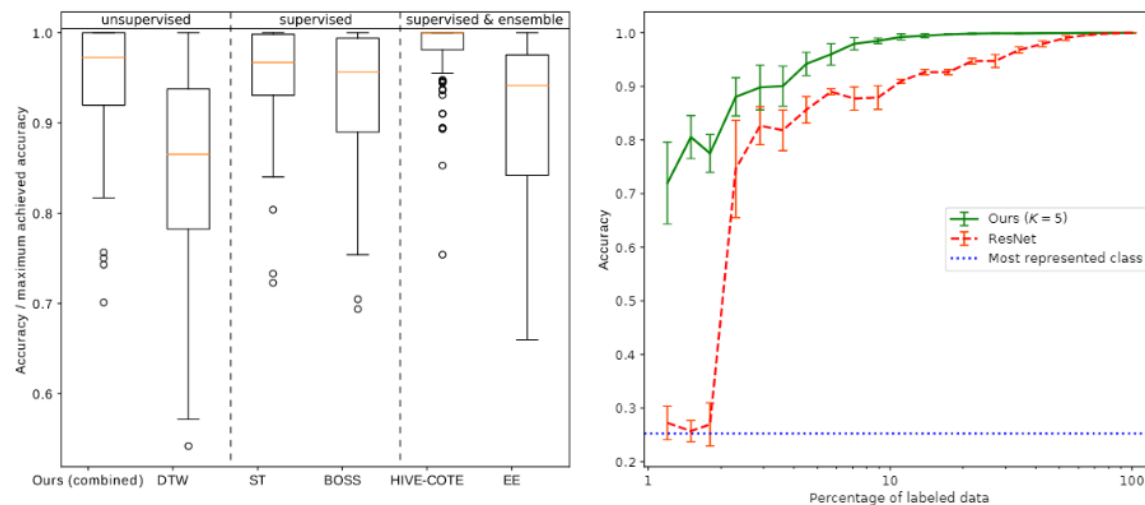


轨迹

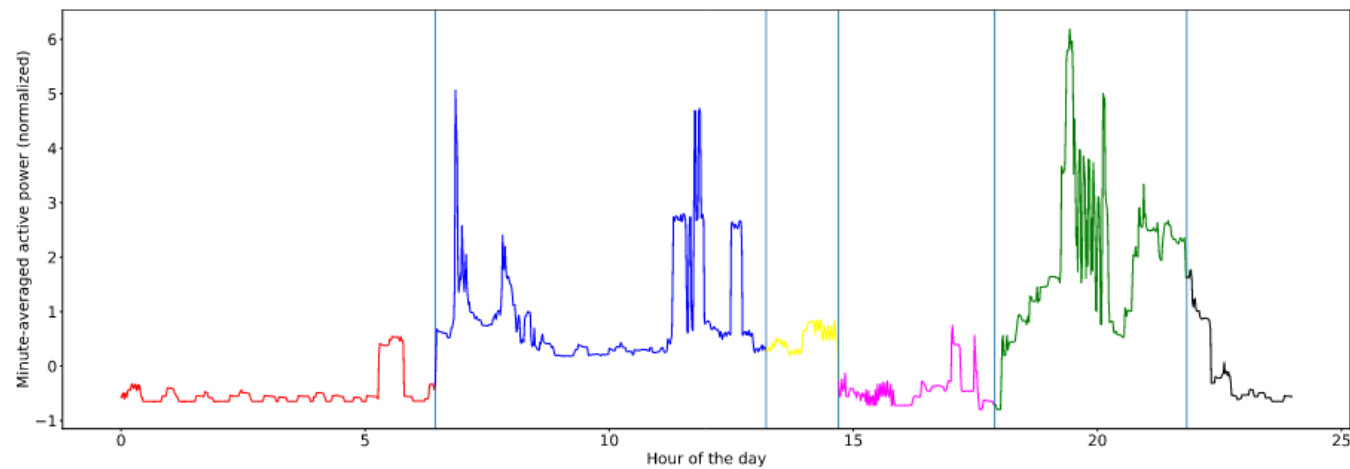
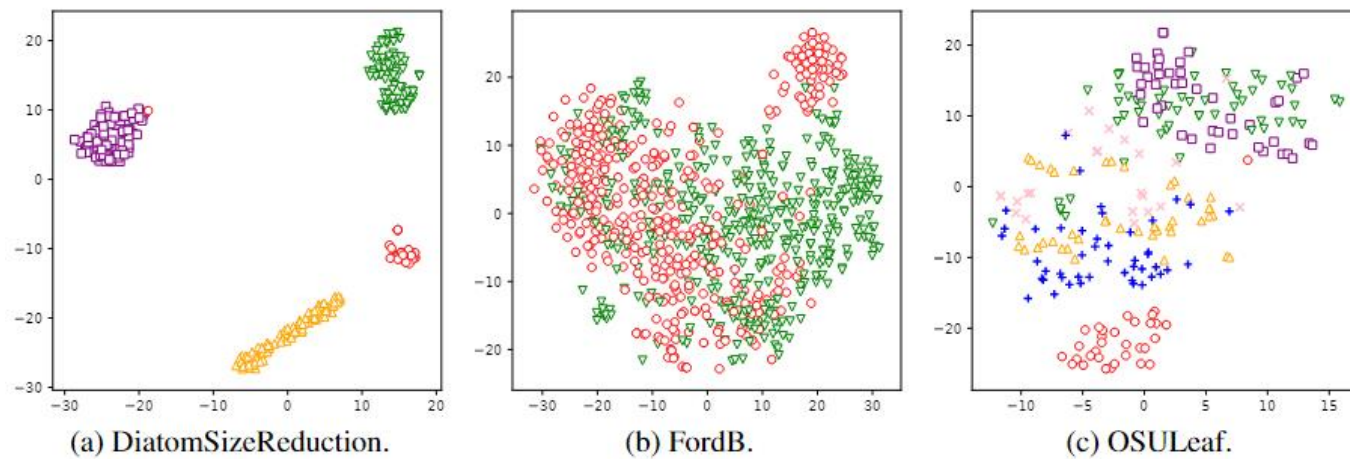
Franceschi, Dieuleveut, and Jaggi 2019: Unsupervised Scalable Representation Learning for Multivariate Time Series
Triplet Loss, 灵感来源于word2vec



不使用输入时间序列的未来值来计算



准确度



Wu et al.2018: Random Warping Series: A Random Features Method for Time-Series Embedding
动态时间扭曲(DTW)是测量时间序列之间差异的最广泛使用的技术，在这项工作中，我们研究了一系列对齐感知正定 (p.d.) 内核，其特征嵌入由随机扭曲系列 (RWS) 的分布给出。所提出的内核不受对角优势问题的影响，而自然具有随机特征(RF)近似，根据时间序列的数量和长度，将现有的基于DTW的技术的计算复杂度从二次降低到线性。

Classifier	RWS		TSEigen		TSMC	
Dataset	Accu	Time	Accu	Time	Accu	Time
Beef	0.733	0.3	0.633	2.1	0.433	0.6
DPTW	0.79	0.5	0.738	7.1	0.738	1.5
IPD	0.969	0.3	0.911	8.6	0.80	1.7
PPOAG	0.868	0.4	0.82	8.9	0.82	1.8
MPOC	0.711	0.8	0.653	19.3	0.653	2.4
POC	0.711	2.4	0.686	172.3	0.66	8.2
LKA	0.792	7.3	0.528	401.5	0.525	39.5
IWBS	0.619	8.9	0.633	784.6	0.57	31.9
TWOP	0.999	4.4	0.976	1395	0.946	32.8
ECG5T	0.933	10.6	0.932	1554	0.918	36.0
CHCO	0.572	6.3	0.529	1668	0.402	45.7
Wafer	0.993	9.6	0.89	3475	0.89	59.3
MALLAT	0.937	33.9	0.898	7982	0.888	282.6
FordB	0.727	43.5	0.704	10069	0.686	216.3
NIFECG	0.907	19.8	0.867	10890	0.582	265
HO	0.843	43.3	0.845	46509	0.82	979.1

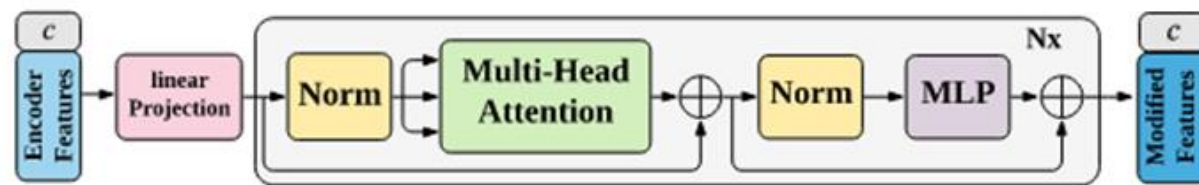
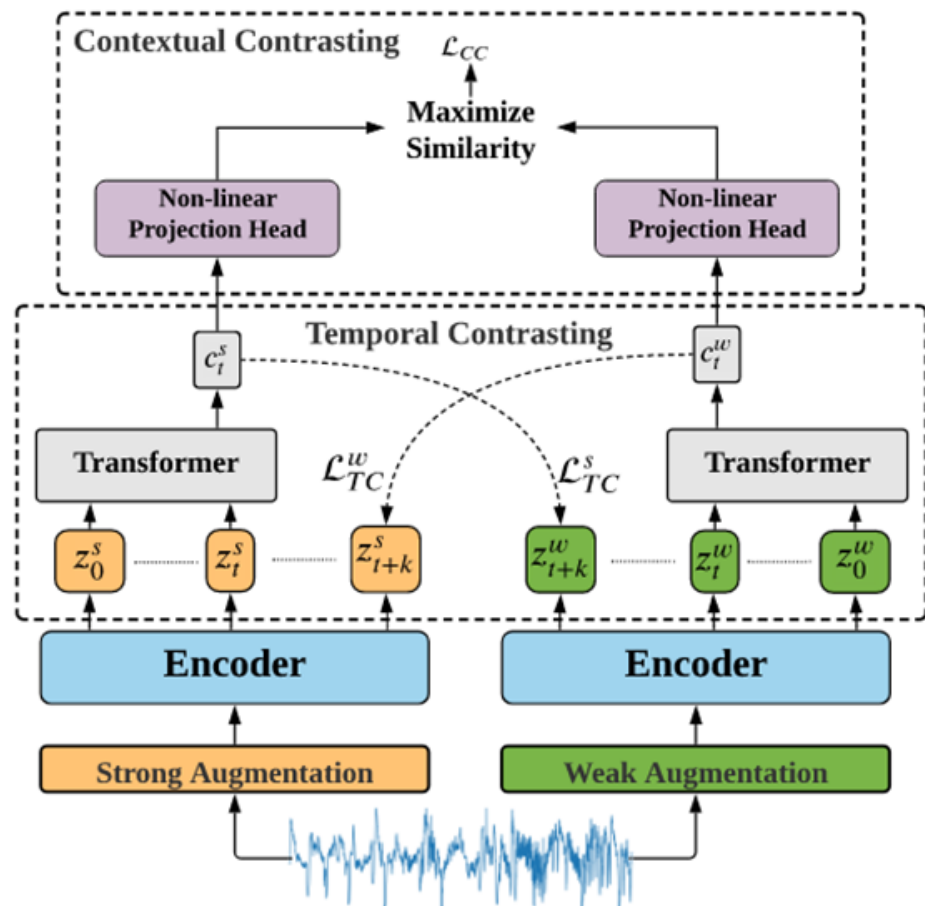
秩R = 32 的 RWS、TEigen 和 TSMC 的分类性能比较

TEigen [Hayashi et al., 2005]: 通过奇异值分解使用 DTW 距离计算的相似性矩阵的低秩特征表示。

TSMC [Lei et al., 2017]: 最近提出的使用矩阵补全方法的基于 DTW 的相似度矩阵的相似性保持表示。

分类、聚类性能比较

Eldele et al. 2021: Time-Series Representation Learning via Temporal and Contextual Contrasting
TS-TCC, 一个时间对比模块和一个上下文对比模块



我们使用Transformer作为自回归模型，MLP块由两个完全连接的层组成，具有非线性ReLU函数和中间的dropout。受BERT模型的启发，我们在输入中添加了一个令牌 $c \in \text{rhh}$ ，其状态在输出中充当代表性上下文向量。

对于强增强，采用置换和抖动策略，通过使用强增强 c_t^s 来预测弱增强 z_{t+k}^w 的未来时间步长

对比损失试图最小化同一样本的预测表示与真实表示之间的点积，同时最大化与小批量内其他样本 $N_{t,k}$ 之间的点积

	HAR		Sleep-EDF		Epilepsy	
Baseline	ACC	MF1	ACC	MF1	ACC	MF1
Random Initialization	57.89±5.13	55.45±5.49	35.61±6.96	23.80±7.96	90.26±1.77	81.12±4.22
Supervised	90.14±2.49	90.31±2.24	83.41±1.44	74.78±0.86	96.66±0.24	94.52±0.43
SSL-ECG [P. Sarkar, 2020]	65.34±1.63	63.75±1.37	74.58±0.60	65.44±0.97	93.72±0.45	89.15±0.93
CPC [Oord <i>et al.</i> , 2018]	83.85±1.51	83.27±1.66	82.82±1.68	73.94±1.75	96.61±0.43	94.44±0.69
SimCLR [Chen <i>et al.</i> , 2020]	80.97±2.46	80.19±2.64	78.91±3.11	68.60±2.71	96.05±0.34	93.53±0.63
TS-TCC (<i>ours</i>)	90.37±0.34	90.38±0.39	83.00±0.71	73.57±0.74	97.23±0.10	95.54±0.08

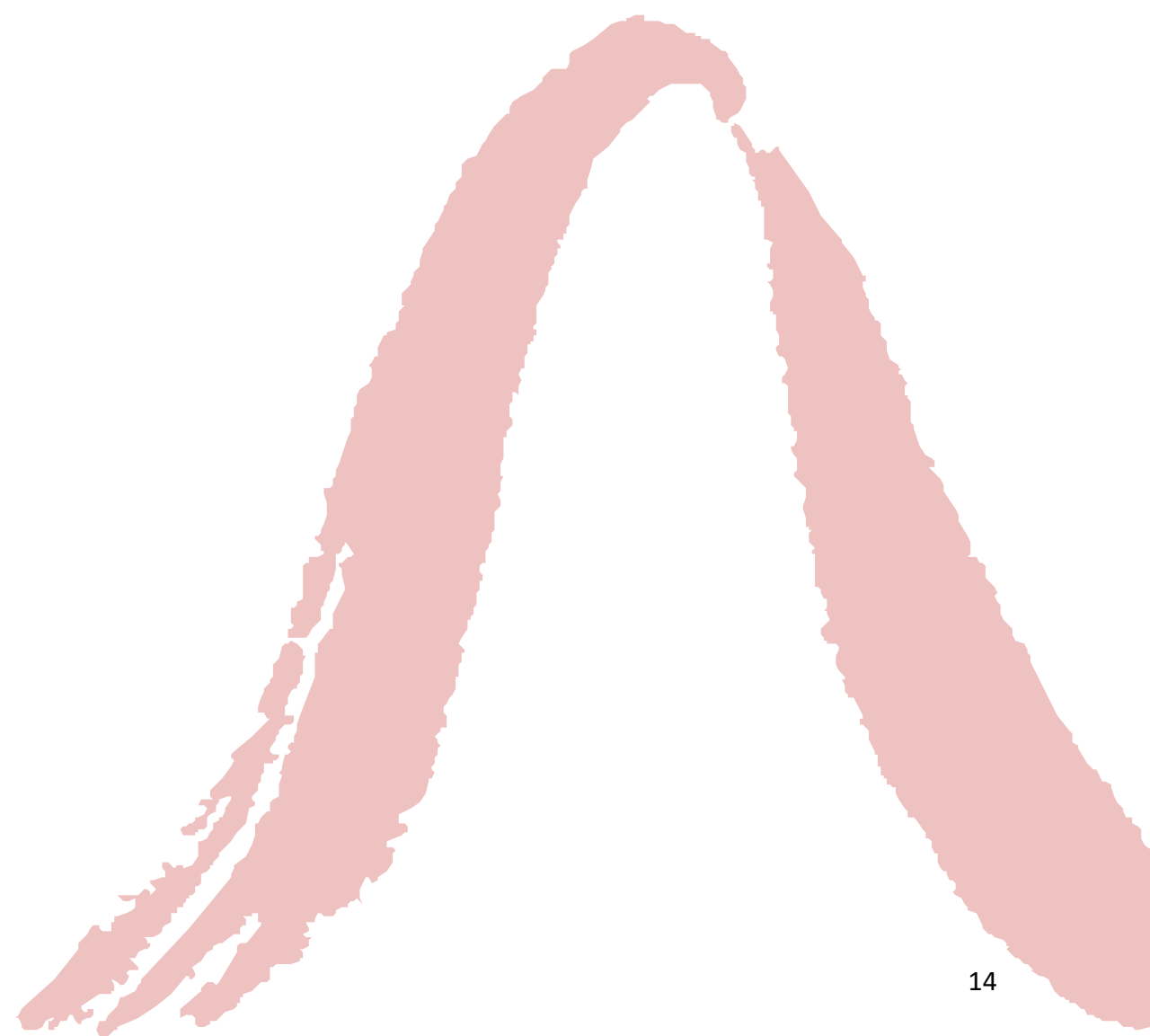
TS-TCC 模型与使用线性分类器评估实验的基线的比较

	A→B	A→C	A→D	B→A	B→C	B→D	C→A	C→B	C→D	D→A	D→B	D→C	AVG
Supervised	34.38	44.94	34.57	52.93	63.67	99.82	52.93	84.02	83.54	53.15	99.56	62.43	63.83
TS-TCC (<i>FT</i>)	43.15	51.50	42.74	47.98	70.38	99.30	38.89	98.31	99.38	51.91	99.96	70.31	67.82

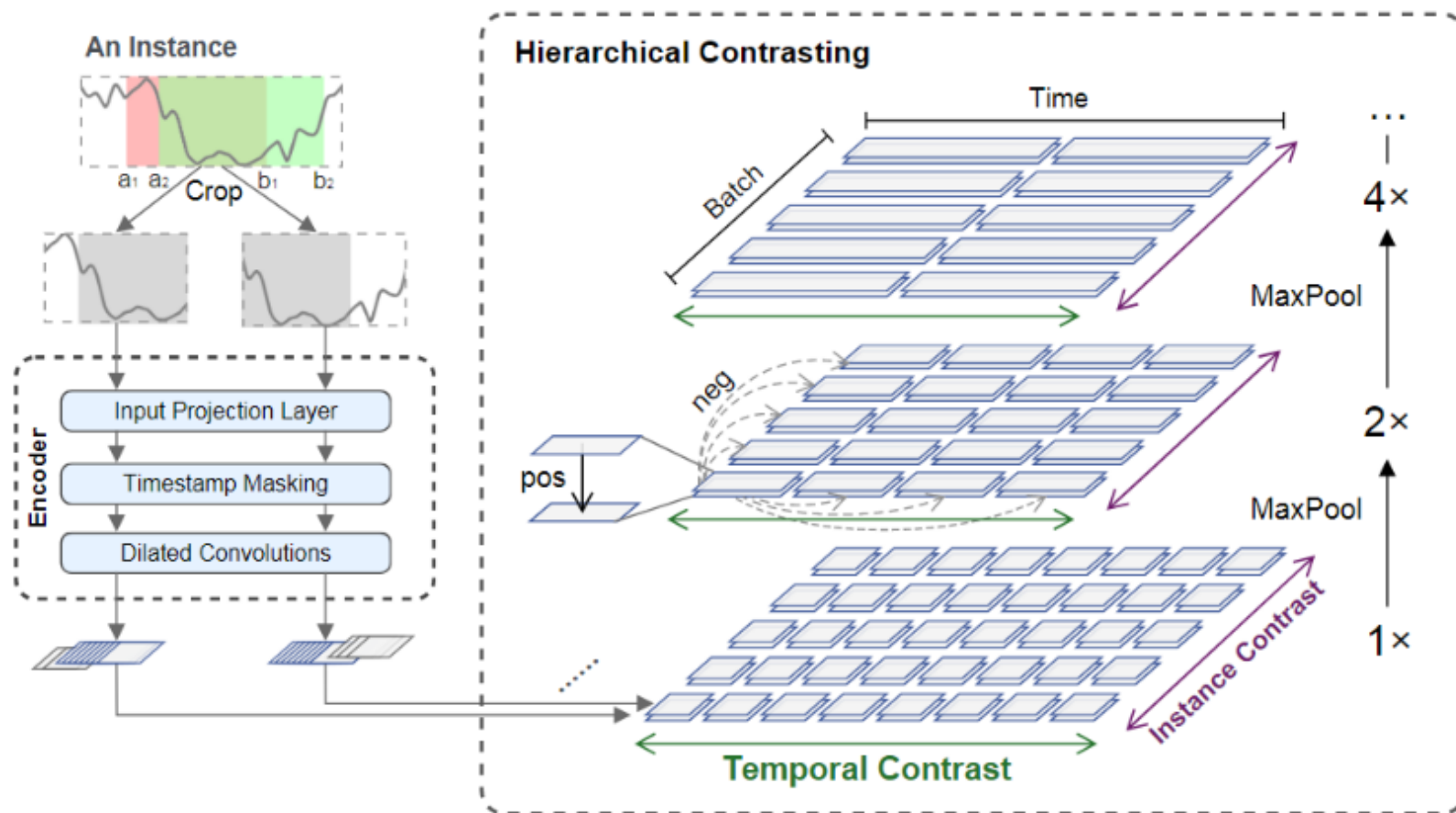
在准确性方面应用于故障诊断数据集的跨域迁移学习实验。(FT 代表微调)



方法



给定一组 N 个实例的时间序列 $X = \{x_1, x_2, \dots, x_N\}$ ，目标是学习一个非线性嵌入函数 f_θ 将每个 x_i 映射到最能描述本身的表示 r_i 。输入时间序列 x_i 维度为 $T \times F$ ，其中 T 是序列长度， F 是特征维度。表示 $r_i = \{r_{i,1}, r_{i,2}, \dots, r_{i,T}\}$ 包含每个时间戳 t 的表示向量 $r_{i,t} \in \mathbb{R}^K$ ，其中 K 是表示向量的维度。



对于每个输入 x_i ，输入投影层是一个完全连接的层，它将时间戳 t 处的观测 $x_{i,t}$ 映射到高维潜在向量 $z_{i,t}$ 。时间戳屏蔽模块在随机选择的时间戳屏蔽潜在向量以生成增强的上下文视图。

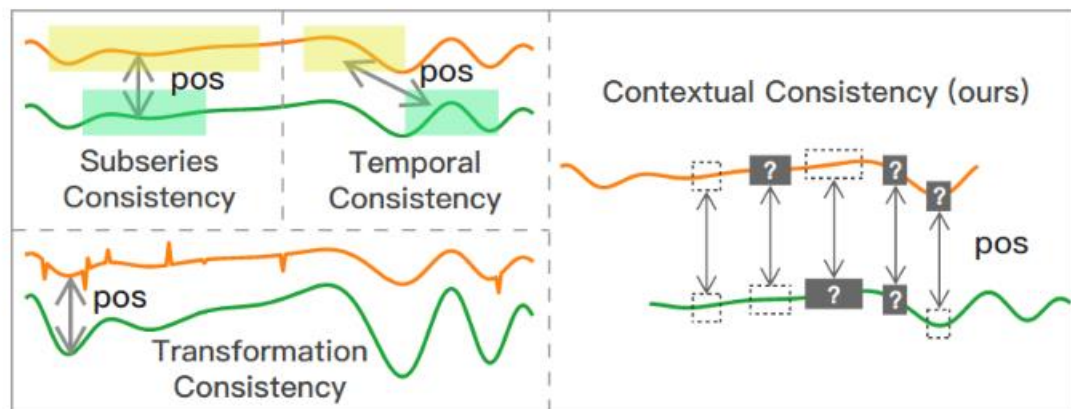
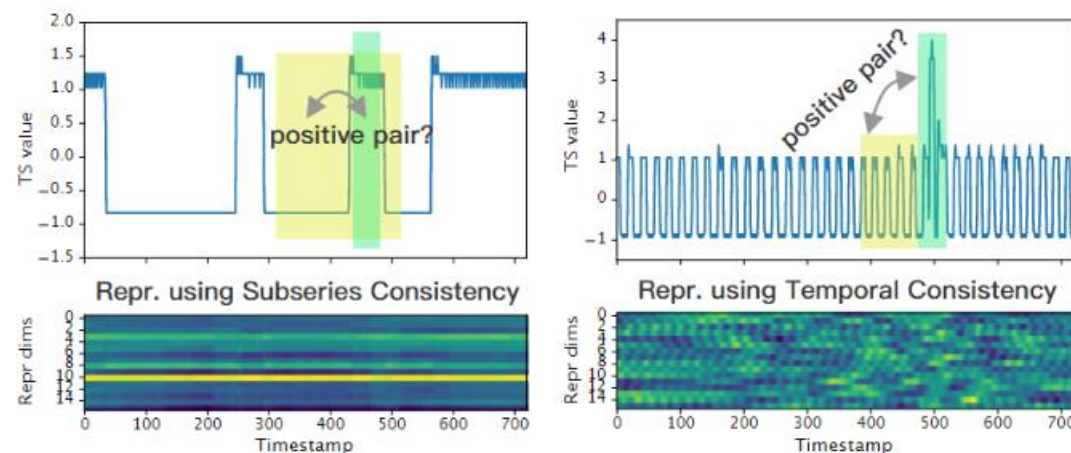


Figure 2: Positive pair selection strategies.

上下文一致性，它将两个扩展上下文中相同时间戳的表示视为正对。通过对输入时间序列应用时间戳屏蔽和随机裁剪来生成上下文。

时间戳屏蔽：用二进制掩码 $m \in \{0,1\}^T$ 沿时间轴掩码输入投影层后的潜在向量 $z_i = \{z_i, t\}$ 。

随机裁剪：重叠段的上下文表示对于两个上下文审查应该是一致的。随机裁剪有助于学习位置不可知表示并避免表示崩溃。时间戳屏蔽和随机裁剪只在训练阶段应用。



(a) Level shifts.

(b) Anomalies.

Figure 3: Two typical cases of the distribution change of time series, with the heatmap visualization of the learned representations over time using subseries consistency and temporal consistency respectively.

为什么在输入投影层之后应用时间戳掩码？

原始输入可能包含值均为 0s 的段，这与掩码标记无法区分

$$z_{i,t} = Wx_{i,t} + b,$$

经输入投影层之后，再将每个掩码时间戳上的潜在向量设置为 0。可以证明网络有一组参数 $\{W, b\}$ ，使得任何

$$\tilde{x} \in \mathbb{R}^F \quad Wx + b \neq 0$$

	Avg. Acc.
Masking after Input Projection	0.829
Masking before Input Projection	0.822
w/o Timestamp Masking	0.820

在 128 个 UCR 数据集上时间戳屏蔽的消融结果

为什么随机裁剪能避免表示崩溃？

在进行时间对比时，去除随机裁剪模块会导致表示崩溃。这种现象可归因于隐式位置对比。已经证明卷积层可以将位置信息隐式编码到潜在空间中。

我们提出的随机裁剪使得网络不可能推断时间点在重叠中的位置。

我们定义一个度量 α 来衡量学习表示 r_i 和纯位置表示 p_i 之间的相似性

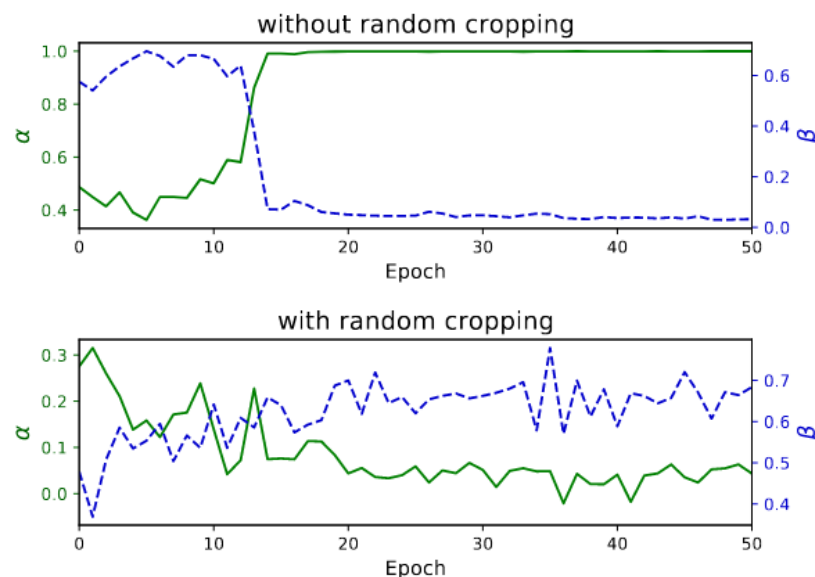
$$\alpha = \frac{1}{NT} \sum_i \sum_t \frac{r_{i,t} \cdot p_{i,t}}{\|r_{i,t}\|_2 \|p_{i,t}\|_2}.$$

β 被定义为测量不同样本的表示之间的偏差

$$\beta = \frac{1}{TK} \sum_t \sum_k \left(\sqrt{\sum_i \left(r_i - \sum_j r_j / N \right)^2 / N} \right)_{t,k}.$$

Without: 网络只学习位置嵌入作为表示，在这种情况下忽略了上下文信息

With: β 保持相对较高的水平，避免崩溃



α 和 β 在 ScreenType 数据集上时间对比损失的训练时期。

时间对比损失：TS2Vec 将输入时间序列的两个视图的相同时间戳的表示作为正数，而来自同一时间序列的不同时间戳的表示作为负数。设 i 是输入时间序列样本的索引， t 是时间戳。然后 $r_{i,t}$ 和 $r'_{i,t}$ 表示相同时间戳 t 的表示，但来自 x_i 的两个增强。时间戳 t 处第 i 个时间序列的时间对比损失可表示为

$$\ell_{temp}^{(i,t)} = -\log \frac{\exp(r_{i,t} \cdot r'_{i,t})}{\sum_{t' \in \Omega} (\exp(r_{i,t} \cdot r'_{i,t'}) + \mathbb{1}_{[t \neq t']} \exp(r_{i,t} \cdot r_{i,t'}))}, \quad (1)$$

其中 Ω 是两个子序列重叠内的时间戳集， $\mathbb{1}$ 是指示函数。

实例对比损失：索引为 (i, t) 的实例对比损失可表示为

$$\ell_{inst}^{(i,t)} = -\log \frac{\exp(r_{i,t} \cdot r'_{i,t})}{\sum_{j=1}^B (\exp(r_{i,t} \cdot r'_{j,t}) + \mathbb{1}_{[i \neq j]} \exp(r_{i,t} \cdot r_{j,t}))}, \quad (2)$$

其中 B 表示批量大小。我们使用与负样本在同一批次的时间戳 t 处其他时间序列的表示。

总损失定义为：

$$\mathcal{L}_{dual} = \frac{1}{NT} \sum_i \sum_t \left(\ell_{temp}^{(i,t)} + \ell_{inst}^{(i,t)} \right). \quad (3)$$

#这段代码定义了一个分层对比损失函数

```
def hierarchical_contrastive_loss(z1, z2, alpha=0.5, temporal_unit=0):
```

```
    loss = torch.tensor(0., device=z1.device)
```

```
    d = 0
```

```
    # 这是一个循环，它会在输入的 z1的第二维（时间维度）大于 1时执行。
```

```
    while z1.size(1) > 1:
```

```
        if alpha != 0:
```

```
            loss += alpha * instance_contrastive_loss(z1, z2)
```

```
        if d >= temporal_unit:
```

```
            if 1 - alpha != 0:
```

```
                loss += (1 - alpha) * temporal_contrastive_loss(z1, z2)
```

```
                # 增加层次计数器d，然后对z1和z2进行最大池化操作以减小时间分辨率，然后将第二维（时间维度）缩小一半。
```

```
                d += 1
```

```
            z1 = F.max_pool1d(z1.transpose(1, 2), kernel_size=2).transpose(1, 2)
```

```
            z2 = F.max_pool1d(z2.transpose(1, 2), kernel_size=2).transpose(1, 2)
```

```
    # 如果最终时间分辨率为1
```

```
    if z1.size(1) == 1:
```

```
        if alpha != 0:
```

```
            loss += alpha * instance_contrastive_loss(z1, z2)
```

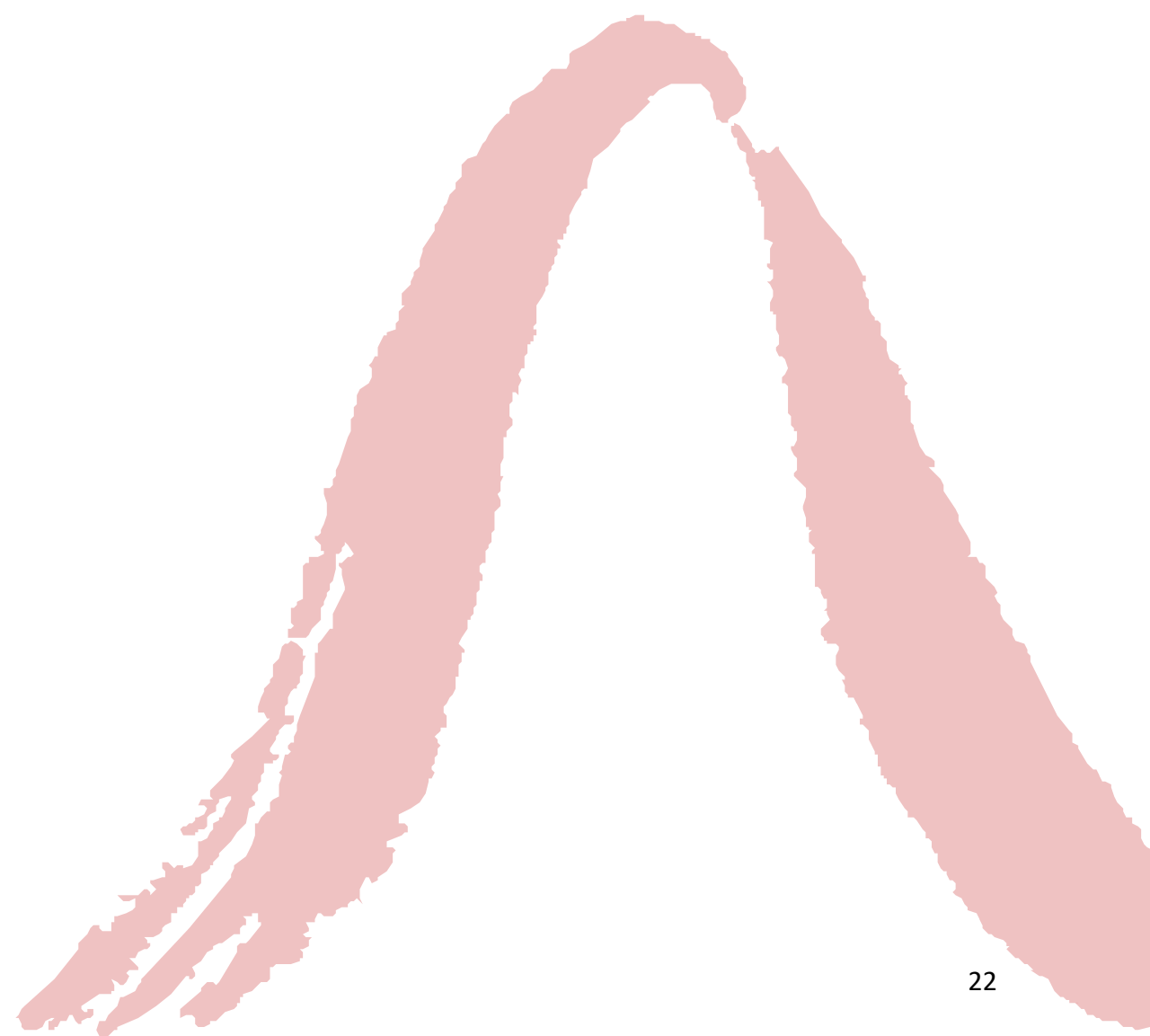
```
        d += 1
```

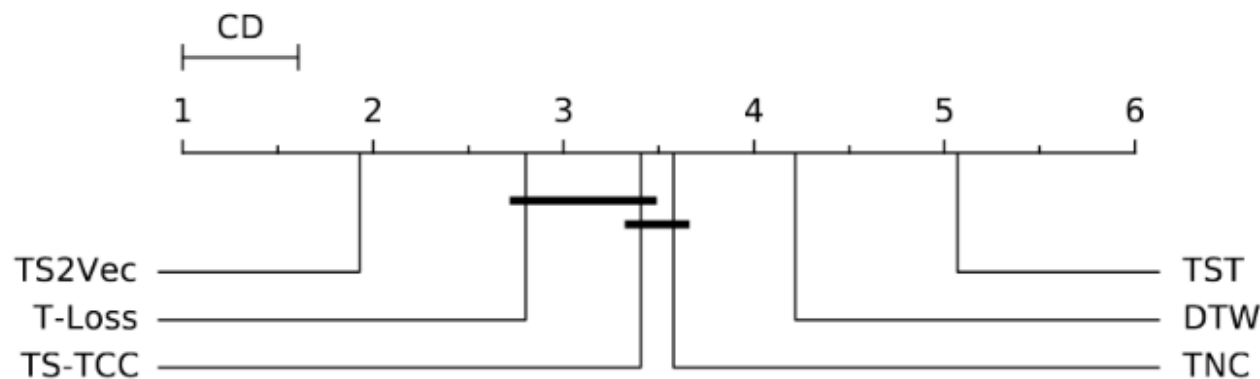
```
    # 返回总体损失除以层次数的结果，得到归一化后的损失值。
```

```
    return loss / d
```



实验





置信度为95%的时间序列分类任务表示学习方法的关键差异(CD)图

显示了所有数据集的 Nemenyi 测试的临界差异图 (Demšar 2006) (包括 125 个 UCR 和 29 个 UEA 数据集), 其中未由粗体线连接的分分类器平均排名存在显著差异。这验证了 TS2Vec 平均排名明显优于其他方法。

Method	125 UCR datasets			29 UEA datasets		
	Avg. Acc.	Avg. Rank	Training Time (hours)	Avg. Acc.	Avg. Rank	Training Time (hours)
DTW	0.727	4.33	–	0.650	3.74	–
TNC	0.761	3.52	228.4	0.677	3.84	91.2
TST	0.641	5.23	17.1	0.635	4.36	28.6
TS-TCC	0.757	3.38	1.1	0.682	3.53	3.6
T-Loss	0.806	2.73	38.0	0.675	3.12	15.1
TS2Vec	0.830 (+2.4%)	1.82	0.9	0.712 (+3.0%)	2.40	0.6

与其他时间序列表示方法相比的时间序列分类结果。TS2Vec、T-Loss、TS-TCC、TST 和 TNC 的表示维度都设置为 320，在 SVM 评估协议下进行公平比较。

TST 学习了一个具有掩码 MSE 损失的基于transformer的模型

ts2vec-main 版本控制

项目

data1.csv test_UCR.py out.pkl read.py test_forecast_csv_univar.py data2.csv classification.py ts2vec.py test_UEA.py ECG200_1

out.pkl

ECG200_UCR_20240121_111920

ECG200_UCR_20240121_111924

ECG200_UCR_20240121_111935

ETT-small

kpl_anomaly_detection_20240120_152754

kpl_anomaly_detection_20240120_174232

kpl_anomaly_detection_coldstart_20240120_15310

venv

运行 test_UCR (1) test_UCR

Arguments: Namespace(batch_size=8, dataset='ECG200', epochs=None, eval=False, gpu=0, irregular=0, iters=None, loader='UCR', lr=0.001, max_threads=None, max_train_length=3000, repr_dims=320, run_name='UCR', save_every=None, seed=None)

Loading data... done

Epoch #0: Loss=6.181175907452901

Epoch #1: Loss=2.5760156710942588

Epoch #2: Loss=2.257256865501484

Epoch #3: Loss=2.181268115838369

Epoch #4: Loss=2.037153512239456

Epoch #5: Loss=1.9996108015378316

Epoch #6: Loss=1.8921663665751393

Epoch #7: Loss=1.8263798157374065

Epoch #8: Loss=1.8706985612710316

Epoch #9: Loss=1.734854926665624

Epoch #10: Loss=1.726209580898285

Epoch #11: Loss=1.6602771878242493

Epoch #12: Loss=1.6760532557964325

Epoch #13: Loss=1.6595763961474101

Epoch #14: Loss=1.5502471427122753

Epoch #15: Loss=1.5410606403944652

Training time: 0:00:10.825511

Evaluation result: {'acc': 0.87, 'auprc': 0.9751162089083254}

Finished.

进程已结束，退出代码为 0

ts2vec-main 版本控制

项目

data1.csv test_UCR.py out.pkl read.py test_forecast_csv_univar.py data2.csv classification.py ts2vec.py test_UEA.py ECG200_1

out.pkl

ECG200_UCR_20240121_111210

ECG200_UCR_20240121_111327

eval_res.pkl

model.pkl

out.pkl

ECG200_UCR_20240121_111920

ECG200_UCR_20240121_111924

ECG200_UCR_20240121_111935

ETT-small

运行 test_UCR (1) test_UEA

return callback

if __name__ == '__main__':

parser = argparse.ArgumentParser()

parser.add_argument('name_or_flag', type=str, default='BasicMotions', help='The dataset name')

parser.add_argument('name_or_flag', type=str, default='UEA',

help='The folder name used to save model, output and evaluation metrics. This can be set to any word')

parser.add_argument('name_or_flag', type=str, default='UEA',

help='The data loader used to load the experimental data. This can be set to UCR, UEA, forecast_csv, forecast_csv_univar, anomaly, or anomaly_co

parser.add_argument('name_or_flag', type=int, default=0,

help='The gpu no. used for training and inference (defaults to 0)')

parser.add_argument('name_or_flag', type=int, default=8, help='The batch size (defaults to 8)')

parser.add_argument('name_or_flag', type=int, default=0.001, help='The learning rate (defaults to 0.001)')

save_checkpoint_callback()

运行 test_UCR (1) test_UEA

C:\Users\xuetong\ts2vec\python.exe E:\Anomaly_Detection\ts2vec-main\test_UEA.py

Dataset: BasicMotions

Arguments: Namespace(batch_size=8, dataset='BasicMotions', epochs=None, eval=False, gpu=0, irregular=0, iters=None, loader='UEA', lr=0.001, max_threads=None, max_train_length=3000, repr_dims=320, run_name='UEA', save_every=None, seed=None)

Loading data... done

Epoch #0: Loss=10.316367244720459

Epoch #1: Loss=4.158628789793091

Epoch #2: Loss=2.867875337680708

Epoch #3: Loss=2.671728324890137

Epoch #4: Loss=2.7416100978851317

Epoch #5: Loss=2.294074225425272

Epoch #6: Loss=2.25501998550415

Epoch #7: Loss=2.33913197517395

Epoch #8: Loss=2.2627058629174085

Epoch #9: Loss=2.2084735870361327

Epoch #10: Loss=2.84999794940802196

Epoch #11: Loss=2.1214486598968505

Epoch #12: Loss=1.8402123928070049

Epoch #13: Loss=2.155638113793457

Epoch #14: Loss=2.190473508834839

Epoch #15: Loss=2.25407508026702883

Epoch #16: Loss=1.9992911338806152

Epoch #17: Loss=2.8652053117752076

Epoch #18: Loss=2.8532889366149902

Epoch #19: Loss=1.9647539854049683

Epoch #20: Loss=2.0417030572891237

Epoch #21: Loss=2.162206268110547

Epoch #22: Loss=2.1433438777923586

Training time: 0:00:10.166818

Evaluation result: {'acc': 0.975, 'auprc': 0.9977272727272727}

Finished.

进程已结束，退出代码为 0

ts2vec-main 版本控制

项目

data1.csv test_UCR.py out.pkl read.py

out.pkl

ECG200_UCR_20240121_111210

ECG200_UCR_20240121_111327

eval_res.pkl

model.pkl

out.pkl

ECG200_UCR_20240121_111920

ECG200_UCR_20240121_111924

ECG200_UCR_20240121_111935

ETT-small

运行 test_UCR (1) test_UEA

return callback

if __name__ == '__main__':

parser = argparse.ArgumentParser()

parser.add_argument('name_or_flag', type=str, default='BasicMotions', help='The dataset name')

parser.add_argument('name_or_flag', type=str, default='UEA',

help='The folder name used to save model, output and evaluation metrics. This can be set to any word')

parser.add_argument('name_or_flag', type=str, default='UEA',

help='The data loader used to load the experimental data. This can be set to UCR, UEA, forecast_csv, forecast_csv_univar, anomaly, or anomaly_co

parser.add_argument('name_or_flag', type=int, default=0,

help='The gpu no. used for training and inference (defaults to 0)')

parser.add_argument('name_or_flag', type=int, default=8, help='The batch size (defaults to 8)')

parser.add_argument('name_or_flag', type=int, default=0.001, help='The learning rate (defaults to 0.001)')

save_checkpoint_callback()

运行 test_UCR (1) test_UEA

Epoch #20: Loss=2.0417030572891237

Epoch #21: Loss=2.162206268110547

Epoch #22: Loss=2.1433438777923586

Epoch #23: Loss=2.0981831550598145

Epoch #24: Loss=1.8497313022613526

Epoch #25: Loss=1.985724973678589

Epoch #26: Loss=1.8677937507629394

Epoch #27: Loss=1.896958595275879

Epoch #28: Loss=1.8763044357299805

Epoch #29: Loss=1.8864251203536987

Epoch #30: Loss=1.9702609035154888

Epoch #31: Loss=1.678030717006395

Epoch #32: Loss=1.7670438766479493

Epoch #33: Loss=1.6415636634826647

Epoch #34: Loss=1.540149756406419

Epoch #35: Loss=1.52182086340332

Epoch #36: Loss=1.4224648237228394

Epoch #37: Loss=1.4414125680923462

Epoch #38: Loss=1.4452088356018867

Epoch #39: Loss=1.4072967852459717

Training time: 0:00:10.166818

Evaluation result: {'acc': 0.975, 'auprc': 0.9977272727272727}

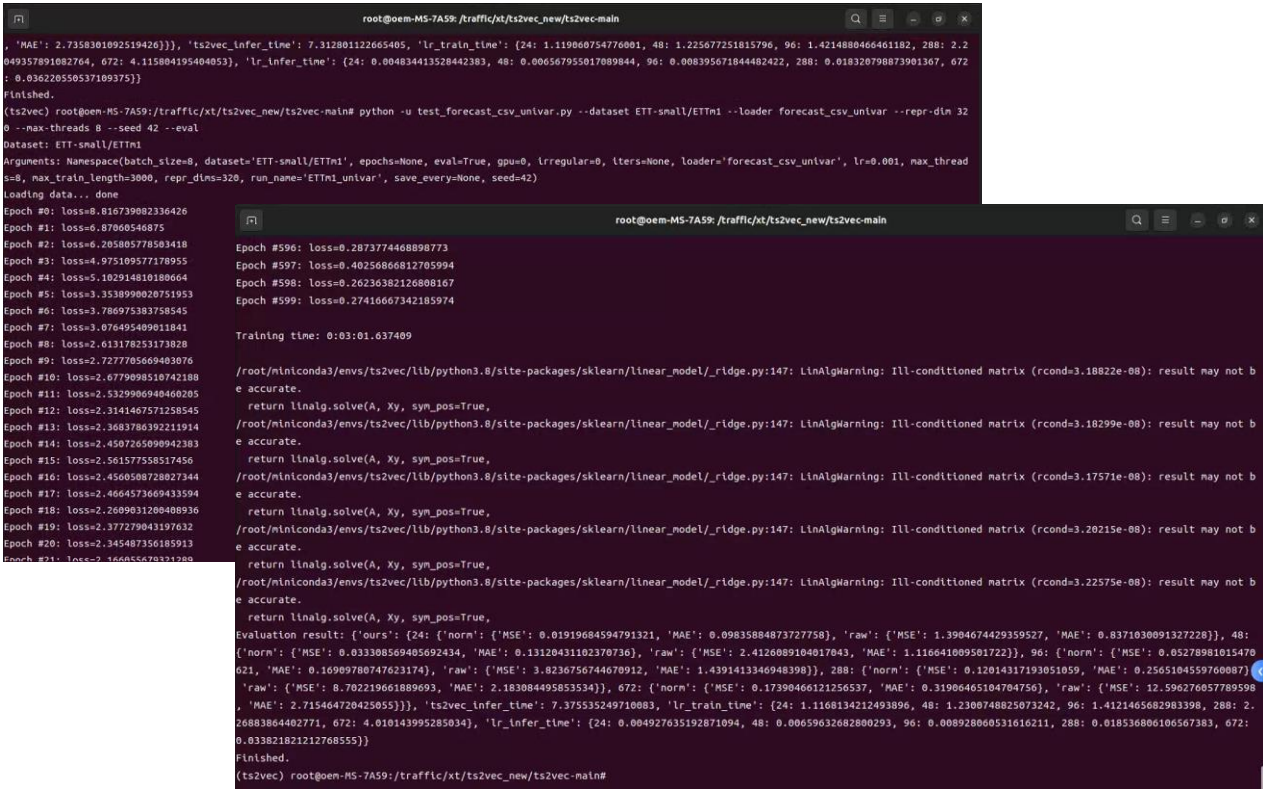
Finished.

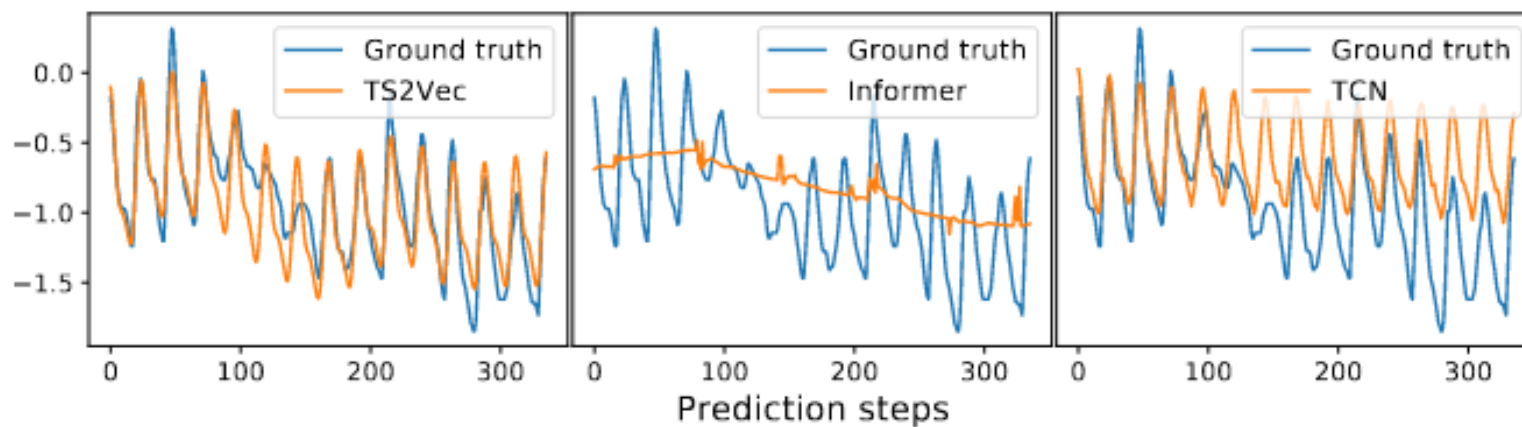
进程已结束，退出代码为 0

Dataset	H	TS2Vec	Informer	LogTrans	N-BEATS	TCN	LSTnet
ETTh ₁	24	0.039	0.098	0.103	0.094	0.075	0.108
	48	0.062	0.158	0.167	0.210	0.227	0.175
	168	0.134	0.183	0.207	0.232	0.316	0.396
	336	0.154	0.222	0.230	0.232	0.306	0.468
	720	0.163	0.269	0.273	0.322	0.390	0.659
ETTh ₂	24	0.090	0.093	0.102	0.198	0.103	3.554
	48	0.124	0.155	0.169	0.234	0.142	3.190
	168	0.208	0.232	0.246	0.331	0.227	2.800
	336	0.213	0.263	0.267	0.431	0.296	2.753
	720	0.214	0.277	0.303	0.437	0.325	2.878
ETTM ₁	24	0.015	0.030	0.065	0.054	0.041	0.090
	48	0.027	0.069	0.078	0.190	0.101	0.179
	96	0.044	0.194	0.199	0.183	0.142	0.272
	288	0.103	0.401	0.411	0.186	0.318	0.462
	672	0.156	0.512	0.598	0.197	0.397	0.639
Electric.	24	0.260	0.251	0.528	0.427	0.263	0.281
	48	0.319	0.346	0.409	0.551	0.373	0.381
	168	0.427	0.544	0.959	0.893	0.609	0.599
	336	0.565	0.713	1.079	1.035	0.855	0.823
	720	0.861	1.182	1.001	1.548	1.263	1.278
Avg.		0.209	0.310	0.370	0.399	0.338	1.099

MSE的单变量时间序列预测结果

附录中提供了完整的预测结果（MSE 和 MAE 的单变量和多变量预测）。TS2Vec 在单变量设置下的平均 MSE 降低了 32.6%，在多变量设置下平均 MSE 降低了 28.2%。





ETTh2 测试集上 TS2Vec、Informer 和 TCN 的预测切片 (H=336)

Phase	H	TS2Vec	Informer
Training	24	60.42 + 2.47	402.31
	48	60.42 + 3.63	163.41
	96	60.42 + 5.10	392.40
	288	60.42 + 10.76	706.94
	672	60.42 + 21.38	938.36
Inference	24	3.01 + 0.01	15.91
	48	3.01 + 0.02	4.85
	96	3.01 + 0.03	14.57
	288	3.01 + 0.10	21.82
	672	3.01 + 0.21	28.49

ETTm1 数据集上多变量预测任务的运行时间（以秒为单位）比较

效率

将异常分数定义为从掩码和未掩码输入计算的表示的差异。具体来说，在推理阶段，经过训练的 TS2Vec 为输入转发两次：第一次，我们只屏蔽最后一个观察 x_t ；第二次，没有应用掩码。

$$\alpha_t = \|r_t^u - r_t^m\|_1. \quad (4)$$

为了避免漂移，我们取前面 Z 点的局部平均值 $\bar{\alpha}_t = \frac{1}{Z} \sum_{i=t-Z}^{t-1} \alpha_i$ ，
通过 $\alpha_t^{adj} = \frac{\alpha_t - \bar{\alpha}_t}{\bar{\alpha}_t}$ 调整异常分数。

在推理中，当 $\alpha_t^{adj} > \mu + \beta\sigma$ 时，时间戳 t 被预测为异常点，其中 μ 和 σ 分别是历史分数的均值和标准差， β 是超参数。

每个时间序列样本前半
部分用于无监督训练，
第二部分用于测试

18.2%

	Yahoo			KPI		
	F ₁	Prec.	Rec.	F ₁	Prec.	Rec.
SPOT	0.338	0.269	0.454	0.217	0.786	0.126
DSPOT	0.316	0.241	0.458	0.521	0.623	0.447
DONUT	0.026	0.013	0.825	0.347	0.371	0.326
SR	0.563	0.451	0.747	0.622	0.647	0.598
TS2Vec	0.745	0.729	0.762	0.677	0.929	0.533
<i>Cold-start:</i>						
FFT	0.291	0.202	0.517	0.538	0.478	0.615
Twitter-AD	0.245	0.166	0.462	0.330	0.411	0.276
Luminol	0.388	0.254	0.818	0.417	0.306	0.650
SR	0.529	0.404	0.765	0.666	0.637	0.697
TS2Vec [†]	0.726	0.692	0.763	0.676	0.907	0.540

5.5%

所有时间序列都用于测试

19.7%

1.0%

单变量时间序列异常检测结果

我们的方法在这两种设置上获得了相似的分數，证明了TS2Vec从一个数据集到另一个数据集中的可转移性。


```
C:\Users\xuetong\tsevec\python.exe E:\Anomaly_Detection\ts2vec-main\test_anomaly.py
Dataset: kpi
Arguments: Namespace(batch_size=8, dataset='kpi', epochs=None, eval=False, gpu=0, irregular=0, iters=None, loader='anomaly', lr=0.001, max_threads=None, max_train_length=3000, repr_dims=320, run_name='anomaly_detection', save_every=None, seed=None)
Loading data... done
Epoch #0: loss=25.442627690055154

Training time: 0:00:25.283923

Evaluation result: {'f1': 0.7058823529411764, 'precision': 0.5454545454545454, 'recall': 1.0, 'infer_time': 3.6861443519592285}
Finished.

进程已结束，退出代码为 0
```

```
C:\Users\xuetong\tsevec\python.exe E:\Anomaly_Detection\ts2vec-main\test_anomaly_coldstart.py
Dataset: kpi
Arguments: Namespace(batch_size=8, dataset='kpi', epochs=None, eval=False, gpu=0, irregular=0, iters=None, loader='anomaly_coldstart', lr=0.001, max_threads=None, max_train_length=3000, repr_dims=320, run_name='anomaly_detection_coldstart', save_every=None, seed=None)
Loading data... done
Epoch #0: loss=1.344932545820872

Training time: 0:00:30.291440

Evaluation result: {'f1': 0.39236857757240245, 'precision': 0.7332421340629275, 'recall': 0.2678493347492030, 'infer_time': 273.9827013015747}
Finished.

进程已结束，退出代码为 0
```

```
(ts2vec) root@oem-MS-7A59:/traffic/xt/ts2vec_new/ts2vec-main# python -u test_anomaly_coldstart.py --dataset kpi --loader anomaly_coldstart --repr-dim 320 --max-threads 8 --seed 1 --eval
Dataset: kpi
Arguments: Namespace(batch_size=8, dataset='kpi', epochs=None, eval=True, gpu=0, irregular=0, iters=None, loader='anomaly_coldstart', lr=0.001, max_threads=8, max_train_length=3000, repr_dims=320, run_name='anomaly_detection_coldstart', save_every=None, seed=1)
Loading data... done
Epoch #0: loss=1.3387185575564702

Training time: 0:01:39.288835

Evaluation result: {'f1': 0.40109274933942407, 'precision': 0.7085443037974684, 'recall': 0.27971765881691546, 'infer_time': 73.02941250801086}
Finished.
```

```
root@oem-MS-7A59:/traffic/xt/ts2vec_new/ts2vec-main
urllib3      2.1.0      pyqt_0     pyqt
wheel        0.41.2     py38h0ea308_0  defaults
xz           5.4.5      h5ee18b_0  defaults
zlib         1.2.13     h5ee18b_0  defaults

(ts2vec) root@oem-MS-7A59:/traffic/xt/ts2vec_new/ts2vec-main# python -u test_anomaly.py --dataset kpi --loader anomaly --repr-dim 320 --max-threads 8 --seed 1 --eval
Dataset: kpi
Arguments: Namespace(batch_size=8, dataset='kpi', epochs=None, eval=True, gpu=0, irregular=0, iters=None, loader='anomaly', lr=0.001, max_threads=8, max_train_length=3000, repr_dims=320, run_name='anomaly_detection', save_every=None, seed=1)
Loading data... done
Epoch #0: loss=116.42627781087702
Epoch #1: loss=7.234282773358721
Epoch #2: loss=5.325899384238503
Epoch #3: loss=3.9105288332158867
Epoch #4: loss=3.4981740684747427
Epoch #5: loss=3.5002740946683017
Epoch #6: loss=3.2108063914559106
Epoch #7: loss=3.445848963477395
Epoch #8: loss=3.1874612894925205
Epoch #9: loss=3.215845909985629
Epoch #10: loss=3.221645723689080
Epoch #11: loss=3.2184423597730498
Epoch #12: loss=3.081863053824605
Epoch #13: loss=3.0970517071010635
Epoch #14: loss=3.048519828182895
Epoch #15: loss=2.9268847785256128
Epoch #16: loss=2.9842149777845903
Epoch #17: loss=2.8894124031066895
Epoch #18: loss=2.758573748848655
Epoch #19: loss=2.758057675621726
Epoch #20: loss=2.696889357133345
Epoch #21: loss=2.6760752417824487
```

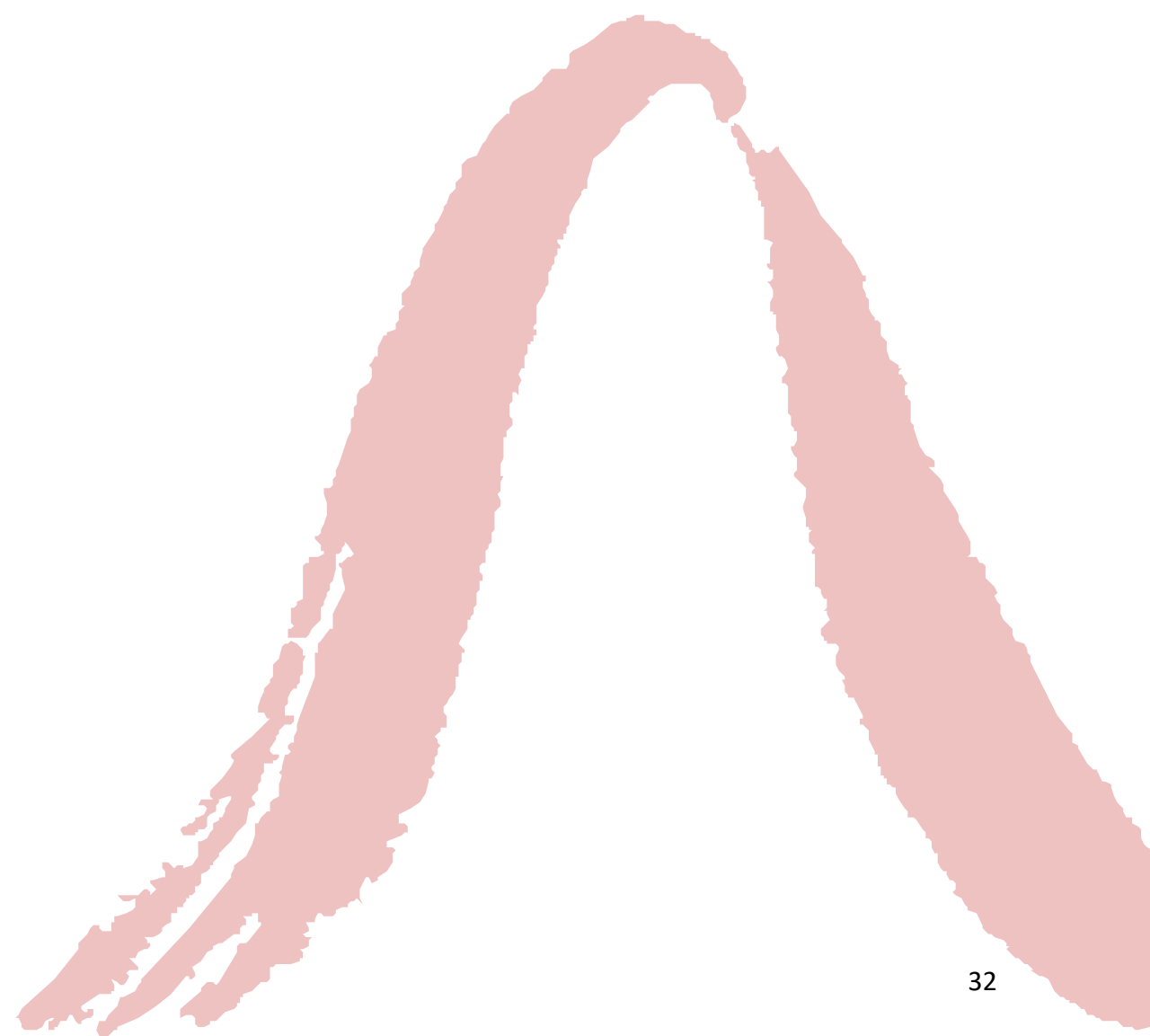
```
root@oem-MS-7A59:/traffic/xt/ts2vec_new/ts2vec-main
Epoch #20: loss=2.4207919294183906
Epoch #20: loss=2.2817410575208865
Epoch #31: loss=2.193384376439181
Epoch #32: loss=2.315580677472095
Epoch #33: loss=2.1742031899365513
Epoch #34: loss=2.2619225169095127
Epoch #35: loss=2.280133832584728
Epoch #36: loss=2.0966768156398428
Epoch #37: loss=2.282624483542009
Epoch #38: loss=2.1036650375886397
Epoch #39: loss=2.049603147940202
Epoch #40: loss=2.123935027555986
Epoch #41: loss=2.053636659275402
Epoch #42: loss=1.95160004529086
Epoch #43: loss=2.0387767986817793
Epoch #44: loss=2.0341271378777246
Epoch #45: loss=2.104169336232272
Epoch #46: loss=1.9881688356399536
Epoch #47: loss=1.9666666534390537
Epoch #48: loss=1.9227091436386108
Epoch #49: loss=2.001801620830189
Epoch #50: loss=1.970127896829085
Epoch #51: loss=2.119343226606196
Epoch #52: loss=2.142772154374556
Epoch #53: loss=2.0205733234232124

Training time: 0:02:44.868306

Evaluation result: {'f1': 0.176790509844994, 'precision': 0.5466321243523317, 'recall': 0.1054472763618191, 'infer_time': 100.98668694496155}
Finished.
(ts2vec) root@oem-MS-7A59:/traffic/xt/ts2vec_new/ts2vec-main#
```



分析



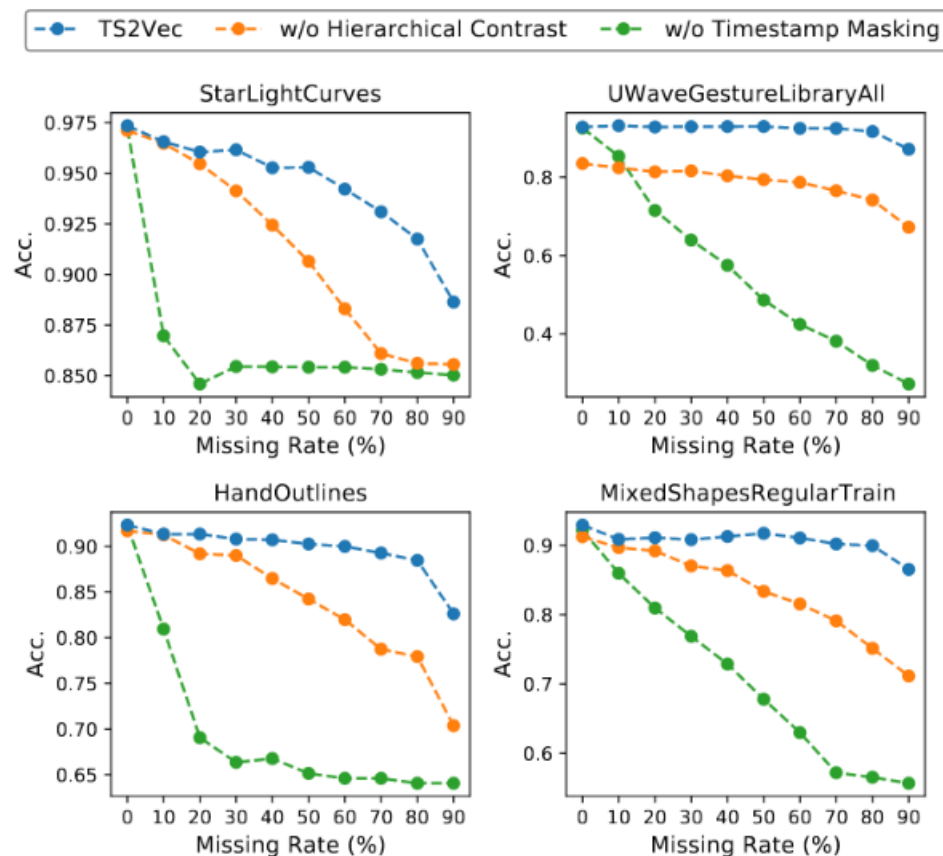
	Avg. Accuracy
TS2Vec	0.829
w/o Temporal Contrast	0.819 (-1.0%)
w/o Instance Contrast	0.824 (-0.5%)
w/o Hierarchical Contrast	0.812 (-1.7%)
w/o Random Cropping	0.808 (-2.1%)
w/o Timestamp Masking	0.820 (-0.9%)
w/o Input Projection Layer	0.817 (-1.2%)
<i>Positive Pair Selection</i>	
Contextual Consistency	
→ Temporal Consistency	0.807 (-2.2%)
→ Subseries Consistency	0.780 (-4.9%)
<i>Augmentations</i>	
抖动 + Jitter	0.814 (-1.5%)
缩放 + Scaling	0.814 (-1.5%)
排列 + Permutation	0.796 (-3.3%)
<i>Backbone Architectures</i>	
Dilated CNN	
→ LSTM	0.779 (-5.0%)
→ Transformer	0.647 (-18.2%)

128 个 UCR 数据集上的消融结果

50% 的缺失值
精度下降率

2.1%

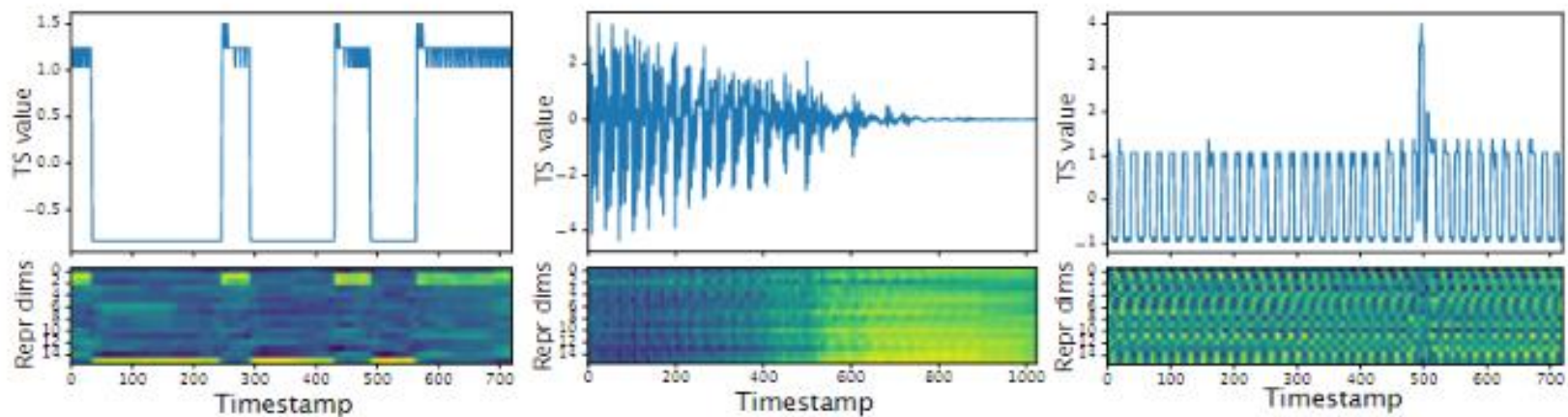
2.1%



时间戳屏蔽使网络能够推断不完整上下文下的表示。分层对比带来了远程信息。

1.2%

UCR 档案中前 4 个最大数据集相对于缺失点率的准确度得分

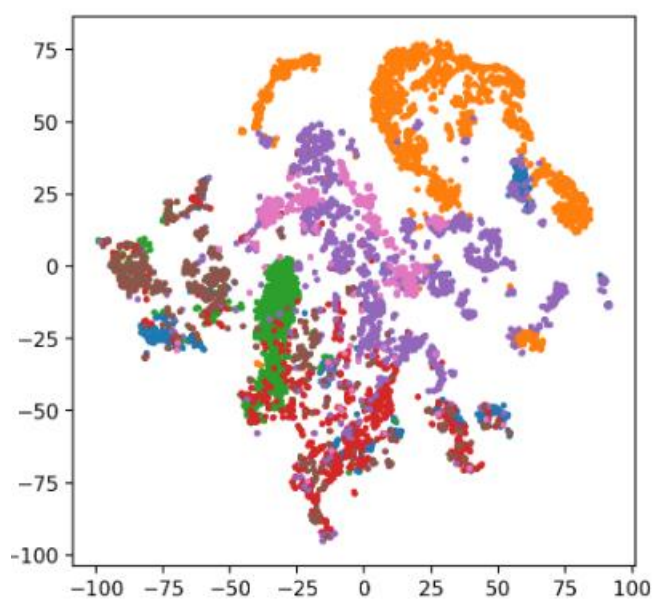


(a) ScreenType.

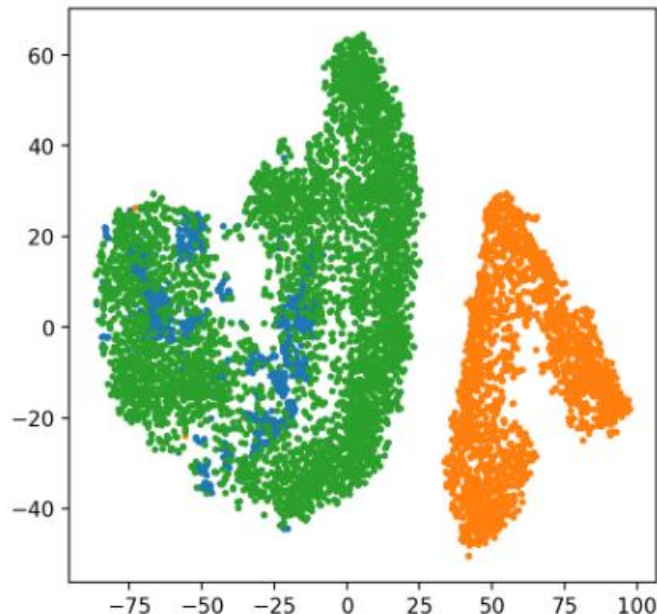
(b) Phoneme.

(c) RefrigerationDevices.

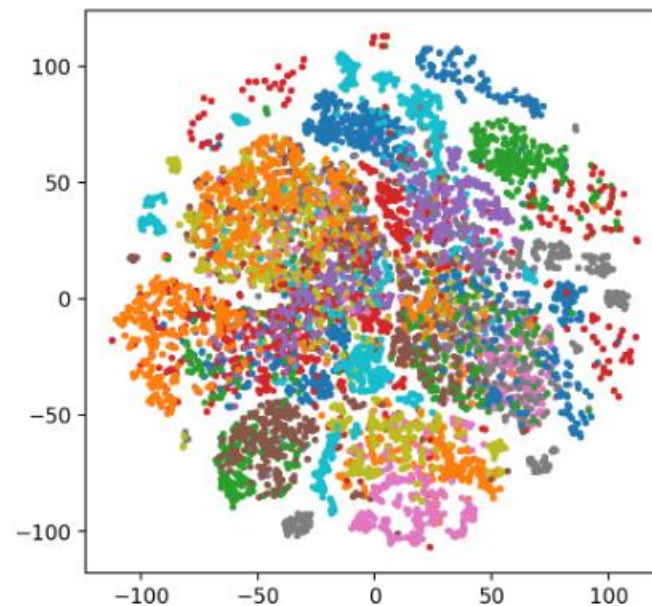
TS2Vec 学习表示随时间的热图可视化



(a) ElectricDevices.




(b) StarLightCurves.



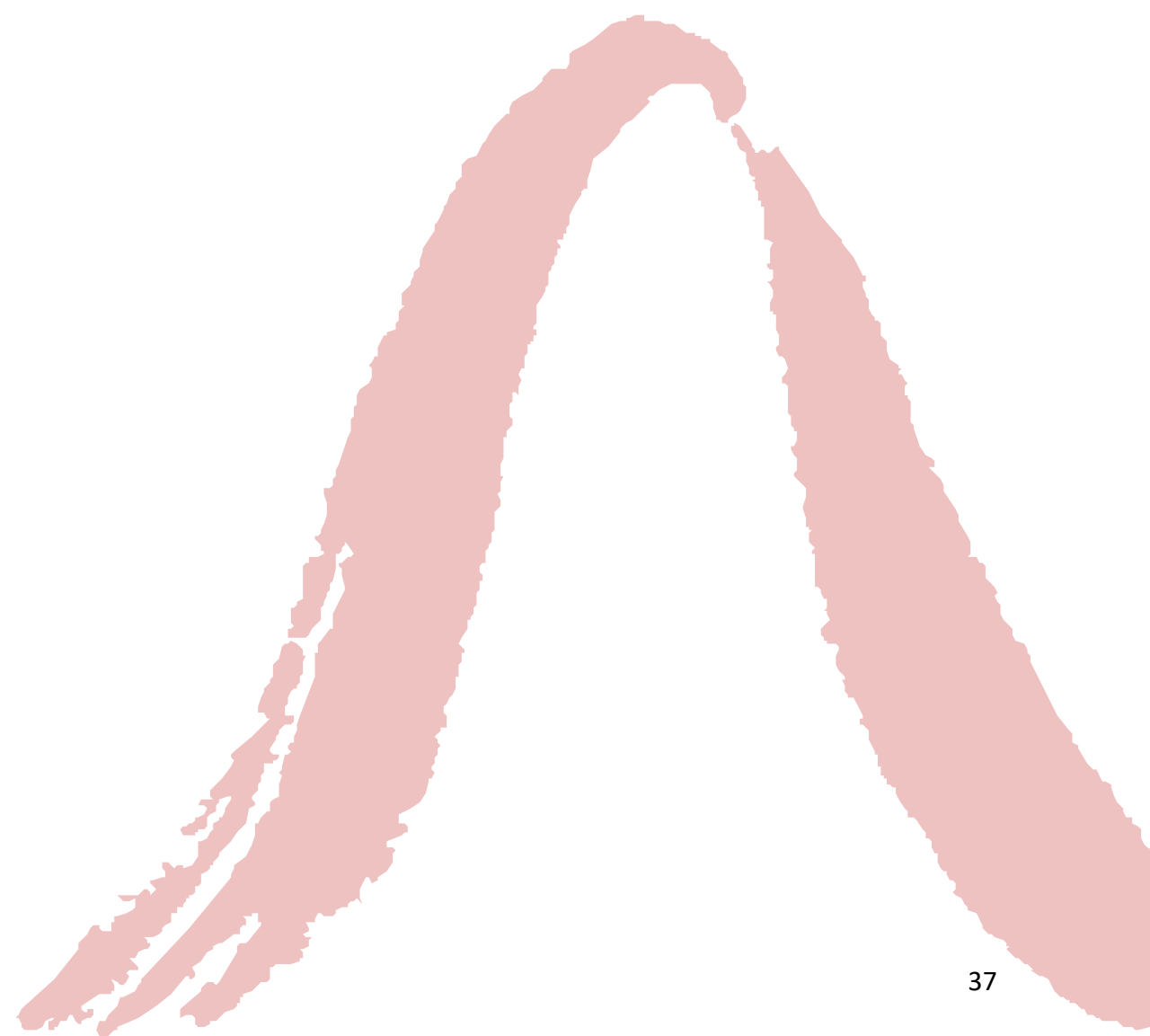
(c) Crop.

TS2Vec 在具有最大测试样本数量的前 3 个 UCR 数据集上学习表示的 T-SNE 可视化。不同的颜色代表不同的类别。

这意味着学习的表示可以区分潜在空间中的不同类别。



结论



TS2Vec:
分层、时间、实例对比

时间序列分类、预测和异常检测

消融研究

不完整数据

可视化

参考: [8.2_TS2Vec回归代码_哔哩哔哩_bilibili](#)



谢谢

