上海人工智能研究院

编码方法：递归图法，格拉米角法，马尔科夫法，马尔科夫场构图法

递归图法

$$R_{i,j} = \Theta(\varepsilon - \|\vec{x}_i - \vec{x}_j\|), \quad \forall i,j \in \{1,\ldots,n-(m-1)\tau\}$$

$$+R(i,j) = \begin{cases} 1 & \text{if} \quad +\|\vec{+}x(i) - \vec{+}x(j)\| \le +\varepsilon+ \\ 0 & \text{otherwise,} \end{cases}$$

✍️ 注：此方法不可逆

方法步骤：

Step1：对时间序列进行采样

Step2：求指定点对之间的二范数距离

Step3： 根据阈值设置为1或0

```python
from pyts.image import RecurrencePlot
X, _, _, _ = load_gunpoint(return_X_y=True)
transformer = RecurrencePlot()
X_new = transformer.transform(X)
```

参考文章：Classification of Time-Series Images Using Deep Convolutional Neural Networks

Classification of Time-Series Images Using Deep Convolutional Neural Networks

格拉米角场

$$\begin{aligned} GASF &= [\cos(\phi_i + \phi_j)] && (4) \\ &= \tilde{X}' \cdot \tilde{X} - \sqrt{I - \tilde{X}^2}' \cdot \sqrt{I - \tilde{X}^2} && (5) \end{aligned}$$

$$\begin{aligned} GADF &= [\sin(\phi_i - \phi_j)] && (6) \\ &= \sqrt{I - \tilde{X}^2}' \cdot \tilde{X} - \tilde{X}' \cdot \sqrt{I - \tilde{X}^2} && (7) \end{aligned}$$

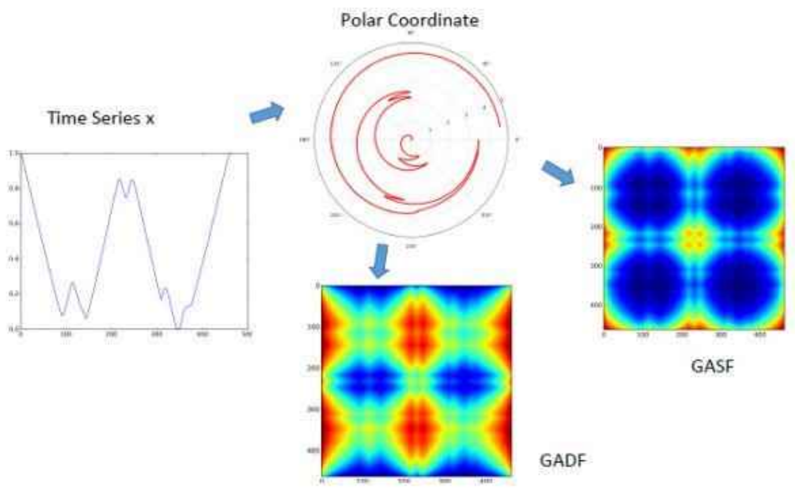


Figure 1: Illustration of the proposed encoding map of Gramian Angular Fields. X is a sequence of rescaled time series in the 'Fish' dataset. We transform X into a polar coordinate system by eq. (3) and finally calculate its GASF/GADF images with eqs. (5) and (7). In this example, we build GAFs without PAA smoothing, so the GAFs both have high resolution.

🍞 格拉米角场法

注：此方法可逆,每一个序列产生唯一的极坐标图

方法步骤：

Step1：对时间序列进行归一化（-1，1）or(0,1)

Step2：求对应时间序列cos对应的角度,时间戳代表半径长度

Step3：根据对应的角度生成GASF或GADF

缺点：

- 极坐标中半径表示时间戳，角度表示时间序列数值
- 通过半径r保持序列的时间依赖性
- 极坐标保留时间关系的绝对值（polar coordinates preserve absolute temporal relations）
- 每个序列产生唯一的极坐标映射图
- 当时间长度为N时候生成矩阵长度为N,可以使用分类聚合法PAA方法降维，

PAA：通过取每个 M 点的平均值来聚合时间序列以减小大小。

可逆性：可通过GAF矩阵的主对角线，恢复笛卡尔坐标下的原始时间序列

参考文章：

《Imaging Time Series to Improve Classification and Imputation》

Imaging Time-Series to Improve Classification and Imputation

《Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks》

Encoding Temporal Markov Dynamics in Graph for Time Series Visualization.

马尔科夫法

$$M = \begin{bmatrix} w_{ij|x_1 \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_1 \in q_i, x_n \in q_j} \\ w_{ij|x_2 \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_2 \in q_i, x_n \in q_j} \\ \vdots & \ddots & \vdots \\ w_{ij|x_n \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_n \in q_i, x_n \in q_j} \end{bmatrix} \quad (8)$$

除了 GSAF 和 GSDF 之外，**《Imaging Time Series to Improve Classification and Imputation》，《Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks》，《Encoding Temporal Markov Dynamics in Graph for Time Series Visualization》**也提到了把时间序列转换成矩阵 Image 的算法 MTF。在 pyts 开源工具库里面，也提到了 MTF 算法的源码。

🌟

Step1:将数据范围值离散化分为Q段

Qi表示数据范围在第i段数据内

Step2：wi,j 表示从i段数据到j段数据的频次

Step3:对行进行归一化处理生成马尔科夫矩阵

⛱️ 马尔可夫场缺点分析：

马尔可夫转换场对时间序列的编码方式相对于格莱姆角场的编码方式有一个明显的缺点：在用马尔可夫转换场对时间序列数据进行编码之前，要选定状态的个数，这个操作具有随意性，而且每一个时间段包含的状态个数也是不同的，如果强行将两段时间序列数据的马尔可夫状态规定成一样是不合理的。

马尔可夫场构图法

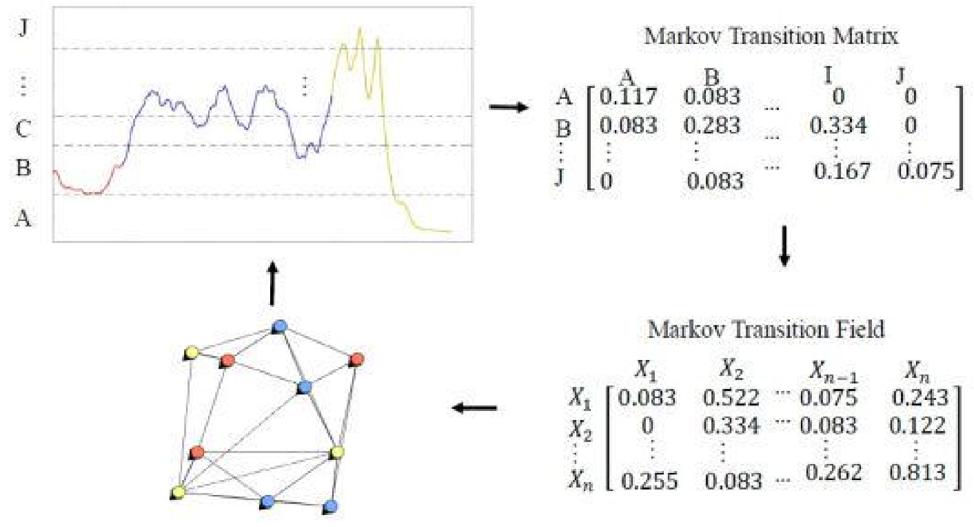
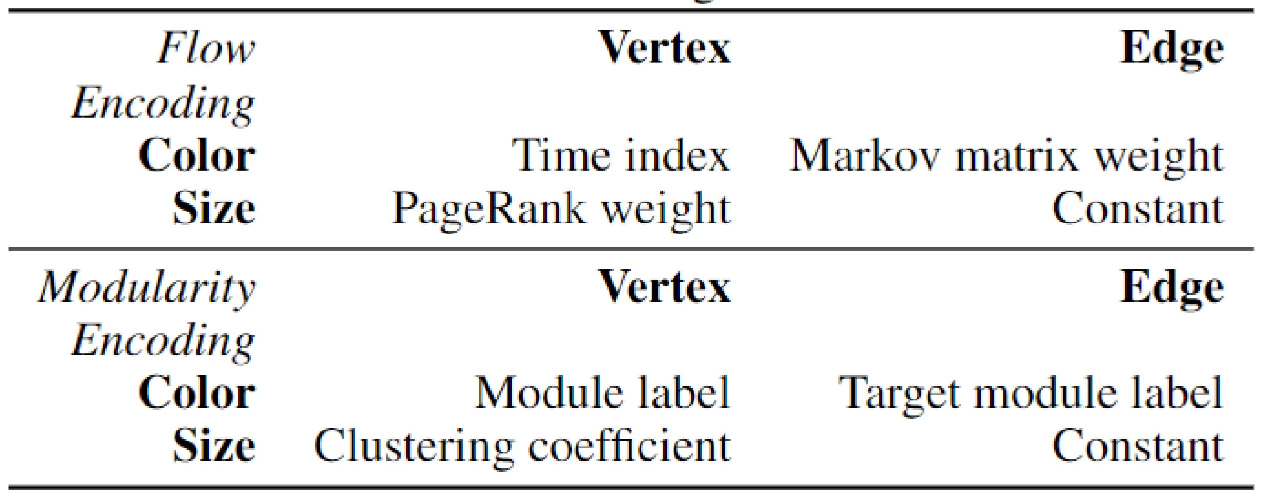


Figure 1: Illustration of the proposed encoding map of a Markov Transition Field. X is a sequence of the typical time series in the 'ECG' dataset. X is first discretized into Q quantile bins. Then we calculate its Markov Transition Matrix W and finally build its MTF M by Equation (2). We reduce the image size from 96×96 to 48×48 by averaging the values in each non-overlapping 2×2 patch.

Table 1: Visual Encoding Framework



Flow Encoding	**Vertex**	**Edge**
Color	Time index	Markov matrix weight
Size	PageRank weight	Constant
Modularity Encoding	**Vertex**	**Edge**
Color	Module label	Target module label
Size	Clustering coefficient	Constant

根据马尔可夫矩阵

Step1:每一个时间节点作为图节点

Step2:PageRank作为节点权重

Step3：马尔可夫权重作为边权重

对图进行社团检测

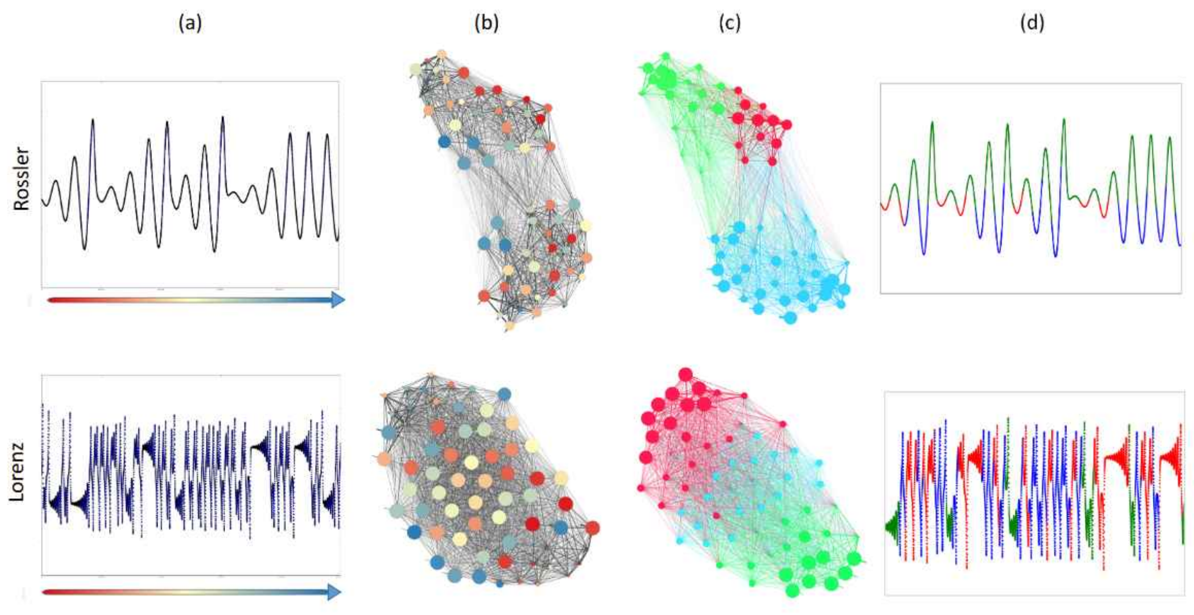
反映射回时间序列

《Encoding Temporal Markov Dynamics in Graph for Time Series Visualization》

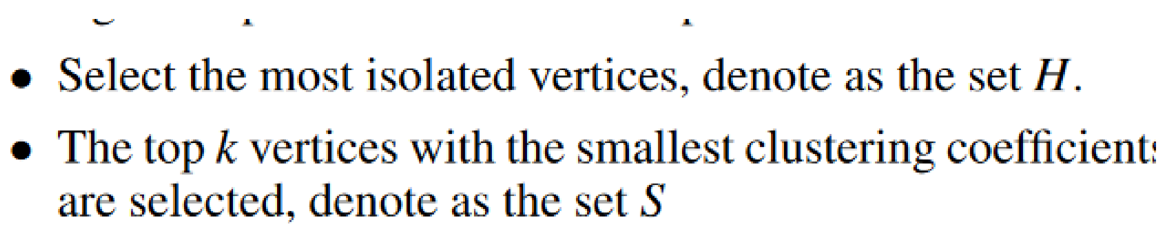
Encoding Temporal Markov Dynamics in Graph for Time Series Visualization.

时序分类

对构成的图使用社团检测算法，再反映射为时间序列

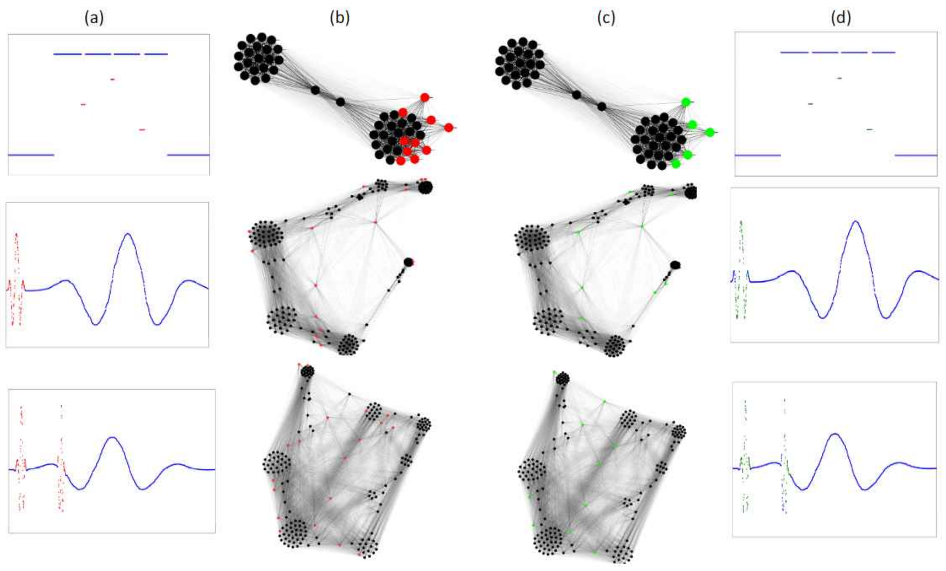


异常检测任务



- Select the most isolated vertices, denote as the set H.
- The top k vertices with the smallest clustering coefficient are selected, denote as the set S

找到最孤立的节点，找到前k个具有最小聚类系数的节点



实验设计

⚽ 设计GASF,GADF,Markov,RecuurencePlot生成图片，

尺寸设置为96*96

使用Resnet18做baseline

epoch设置为100

batch_size 设置为4，设置过大担心影响效果

采用5个UCR 时序数据集 做实验

Pytorch 1.8/1.1

GPU:v100/ p100 Re/cpu Markov

数据集地址：

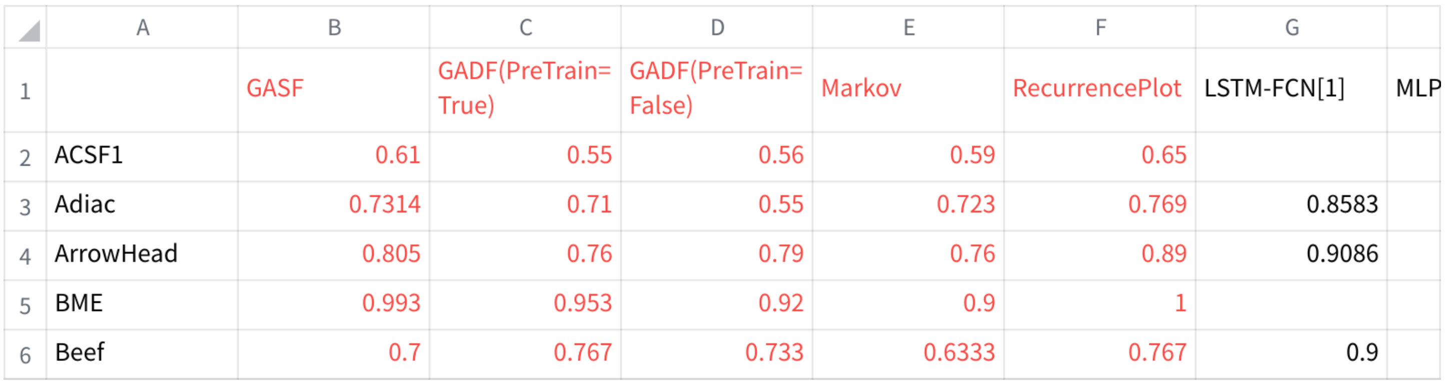
（PS：需要注意到

1.本实验采用了预训练模型，但实际上序列生成的图不应当具备ImgaeNet相关的图片参数，不确定是否从头开始训练会更有利于本模型

2.在训练过程中，使用对训练集做测试，训练结果基本可以达到百分之百的正确率，说明模型可以具备相应的拟合能力但是数据量不够

)

{'ACSF1': 0.56, 'Adiac': 0.5549872122762148, 'ArrowHead': 0.7942857142857143, 'BME': 0.92, 'Beef': 0.7333333333333333, 'BeetleFly': 0.7, 'BirdChicken': 0.75}



	A	B	C	D	E	F	G	
1		GASF	GADF(PreTrain=True)	GADF(PreTrain=False)	Markov	RecurrencePlot	LSTM-FCN[1]	MLP
2	ACSF1	0.61	0.55	0.56	0.59	0.65		
3	Adiac	0.7314	0.71	0.55	0.723	0.769	0.8583	
4	ArrowHead	0.805	0.76	0.79	0.76	0.89	0.9086	
5	BME	0.993	0.953	0.92	0.9	1		
6	Beef	0.7	0.767	0.733	0.6333	0.767	0.9	

UCR数据集可视化以及相关介绍

问题：不应该使用预训练参数，因为图像纹理结构不一样

[1]LSTM Fully Convolutional Networks for Time Series Classification

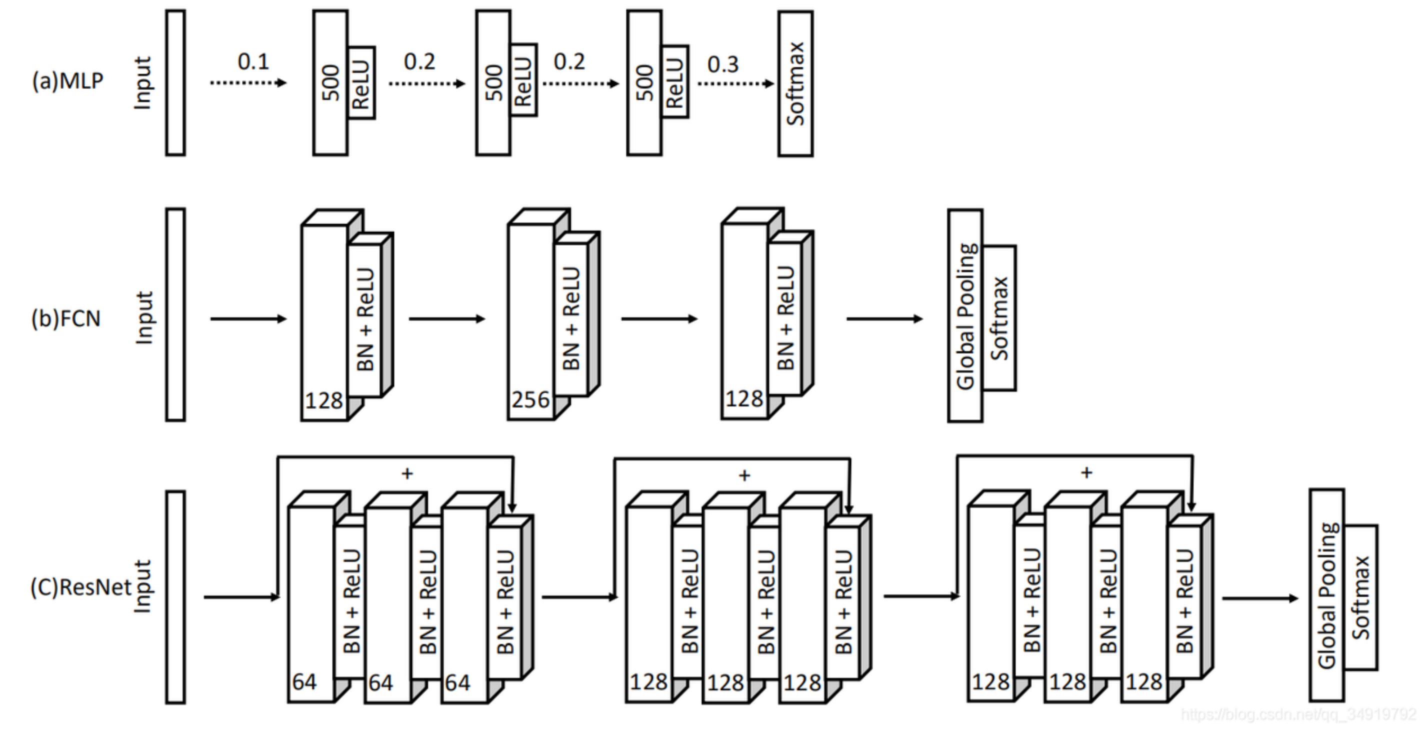
[2]Time series classification from scratch with deep neural networks: A strong baseline

[3]Classification of Time-Series Images Using Deep Convolutional Neural Networks

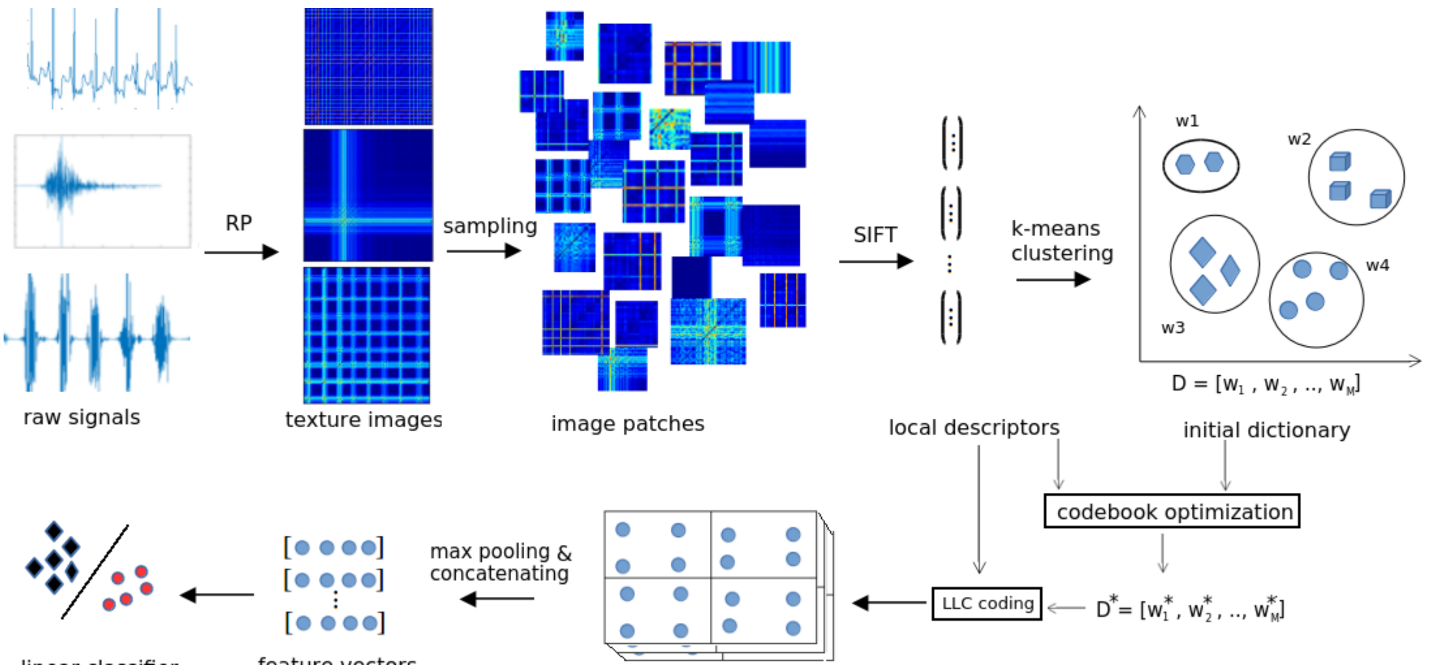
[4]Imaging Time-Series to Improve Classification and Imputation

Baseline:

Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline



(a)MLP
Input
0.1
500
ReLU
0.2
500
ReLU
0.2
500
ReLU
0.3
Softmax
(b)FCN
Input
128
BN + ReLU
256
BN + ReLU
128
BN + ReLU
Global Pooling
Softmax
(C)ResNet
Input
+
64
BN + ReLU
64
BN + ReLU
64
BN + ReLU
+
128
BN + ReLU
128
BN + ReLU
128
BN + ReLU
+
128
BN + ReLU
128
BN + ReLU
128
BN + ReLU
Global Pooling
Softmax

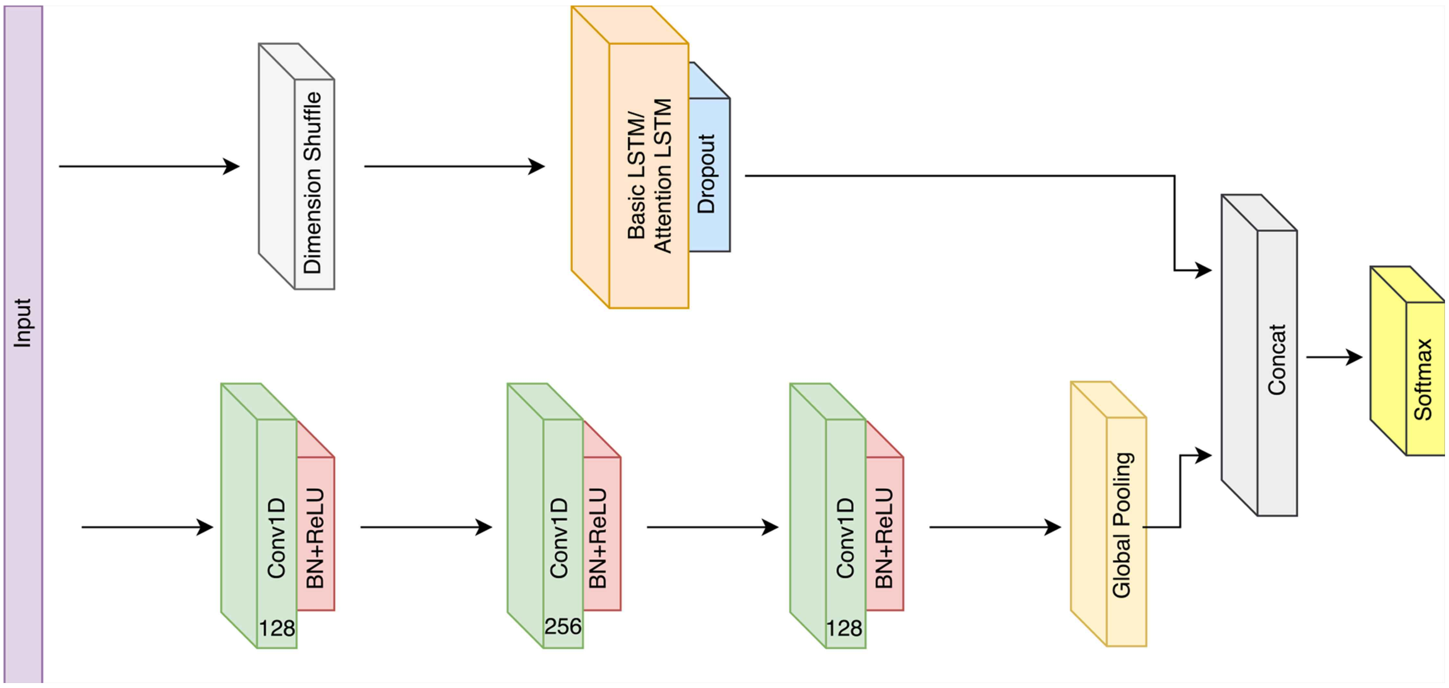


raw signals
RP
texture images
sampling
image patches
SIFT
local descriptors
k-means clustering
w1
w2
w3
w4
D = [w1 , w2 , .., wM]
initial dictionary
codebook optimization
D*= [w1*, w2*, .., wM*]
LLC coding
max pooling & concatenating

TABLE 1. Performance comparison of proposed models with the rest.

Dataset	Existing SOTA	LSTM-FCN	Refined LSTM-FCN	ALSTM-FCN	Refined ALSTM-FCN
Adiac	0.8570 [11]	0.8593	0.8849	0.8670	0.8900*
ArrowHead	0.8800 [11]	0.9086	0.9029	0.9257*	0.9200
Beef	0.9000 [38]	0.9000	0.9330	0.9333*	0.9333*
BeetleFly	0.9500 [11, 19]	0.9500	1.0000*	1.0000*	1.0000*
BirdChicken	0.9500 [11, 17]	1.0000*	1.0000*	1.0000*	1.0000*
Car	0.9330 [11]	0.9500	0.9670	0.9667	0.9833*
CBF	1.0000 [11]	0.9978	1.0000*	0.9967	0.9967
ChloConc	0.8720 [11]	0.8099	1.0000*	0.8070	0.8070
CinC_ECG	0.9949 [8]	0.8862	0.9094	0.9058	0.9058
Coffee	1.0000 [19, 39]	1.0000*	1.0000*	1.0000*	1.0000*
Computers	0.8480 [11]	0.8600	0.8600	0.8640*	0.8640*
Cricket_X	0.8210 [11]	0.8077	0.8256*	0.8051	0.8051
Cricket_Y	0.8256 [8]	0.8179	0.8256*	0.8205	0.8205
Cricket_Z	0.8154 [8]	0.8103	0.8257	0.8308	0.8333*
DiaSizeRed	0.9670 [20]	0.9673	0.9771*	0.9739	0.9739
DistPhxAgeGp	0.8350 [11]	0.8600	0.8600	0.8625*	0.8600
DistPhxCorr	0.8200 [11]	0.8250	0.8217	0.8417*	0.8383
DistPhxTW	0.7900 [11]	0.8175	0.8100	0.8175	0.8200*
Earthquakes	0.8010 [11]	0.8354*	0.8261	0.8292	0.8292
ECG200	0.9200 [11]	0.9000	0.9200*	0.9100	0.9200
ECG5000	0.9482 [19]	0.9473	0.9478	0.9484	0.9496*
ECGFiveDays	1.0000 [19, 39]	0.9919	0.9942	0.9954	0.9954
ElectricDevices	0.7993 [17]	0.7681	0.7633	0.7672	0.7672
FaceAll	0.9290 [11]	0.9402	0.9680	0.9657	0.9728*
FaceFour	1.0000 [16, 19]	0.9432	0.9772	0.9432	0.9432
FacesUCR	0.9580 [11]	0.9293	0.9898*	0.9434	0.9434
FiftyWords	0.8198 [20]	0.8044	0.8066	0.8242	0.8286*
Fish	0.9890 [11]	0.9829	0.9886	0.9771	0.9771
FordA	0.9727 [19]	0.9272	0.9733*	0.9267	0.9267
FordB	0.9173 [39]	0.9180	0.9186*	0.9158	0.9158
Gun_Point	1.0000 [19, 38]	1.0000*	1.0000*	1.0000*	1.0000*
Ham	0.7810 [11]	0.7714	0.8000	0.8381*	0.8000
HandOutlines	0.9487 [19]	0.8930	0.8870	0.9030	0.9030
Haptics	0.5510 [11]	0.5747*	0.5584	0.5649	0.5584
Herring	0.7030 [11]	0.7656*	0.7188	0.7500	0.7656*
InlineSkate	0.6127 [19]	0.4655	0.5000	0.4927	0.4927
InsWngSnd	0.6525 [8]	0.6616	0.6696	0.6823*	0.6818
ItPwDmd	0.9700 [11]	0.9631	0.9699	0.9602	0.9708*
LrgKitApp	0.8960 [11]	0.9200*	0.9200*	0.9067	0.9120
Lighting2	0.8853 [20]	0.8033	0.8197	0.7869	0.7869
Lighting7	0.8630 [11]	0.8356	0.9178*	0.8219	0.9178*
Mallat	0.9800 [11]	0.9808	0.9834	0.9838	0.9842*
Meat	1.0000 [11]	0.9167	1.0000*	0.9833	1.0000*
MedicalImages	0.7920 [11]	0.8013	0.8066*	0.7961	0.7961
MidPhxAgeGp	0.8144 [16]	0.8125	0.8150	0.8175*	0.8075
MidPhxCorr	0.8076 [16]	0.8217	0.8333	0.8400	0.8433*
MidPhxTW	0.6120 [11]	0.6165	0.6466	0.6466*	0.6316
MoteStrain	0.9500 [11]	0.9393	0.9569*	0.9361	0.9361
NonInv_Thor1	0.9610 [11]	0.9654	0.9657	0.9751	0.9756*
NonInv_Thor2	0.9550 [11]	0.9623	0.9613	0.9664	0.9674*
OliveOil	0.9333 [19]	0.8667	0.9333	0.9333	0.9667*
OSULeaf	0.9880 [11]	0.9959*	0.9959*	0.9959*	0.9917

OSULeaf	0.9880 [11]	0.9959*	0.9959*	0.9959*	0.9917
PhalCorr	0.8300 [11]	0.8368	0.8392*	0.8380	0.8357
Phoneme	0.3492 [8]	0.3776*	0.3602	0.3671	0.3623
Plane	1.0000 [19, 20]	1.0000*	1.0000*	1.0000*	1.0000*
ProxPhxAgeGp	0.8832 [38]	0.8927*	0.8878	0.8878	0.8927*
ProxPhxCorr	0.9180 [11]	0.9450*	0.9313	0.9313	0.9381
ProxPhxTW	0.8150 [17]	0.8350	0.8275	0.8375*	0.8375*
RefDev	0.5813 [38]	0.5813	0.5947*	0.5840	0.5840
ScreenType	0.7070 [11]	0.6693	0.7073	0.6907	0.6907
ShapeletSim	1.0000 [17, 19]	0.9722	1.0000*	0.9833	0.9833
ShapesAll	0.9183 [19]	0.9017	0.9150	0.9183	0.9217*
SmlKitApp	0.8030 [11]	0.8080	0.8133*	0.7947	0.8133*
SonyAIBOI	0.9850 [11]	0.9817	0.9967	0.9700	0.9983*
SonyAIBOII	0.9620 [11]	0.9780	0.9822*	0.9748	0.9790
StarlightCurves	0.9796 [8]	0.9756	0.9763	0.9767	0.9767
Strawberry	0.9760 [17]	0.9838	0.9864	0.9838	0.9865*
SwedishLeaf	0.9664 [19]	0.9792	0.9840	0.9856*	0.9856*
Symbols	0.9668 [17]	0.9839	0.9849	0.9869	0.9889*
Synth_Cntr	1.0000 [8, 20]	0.9933	1.0000*	0.9900	0.9900
ToeSeg1	0.9737 [8]	0.9825	0.9912*	0.9868	0.9868
ToeSeg2	0.9615 [17]	0.9308	0.9462	0.9308	0.9308
Trace	1.0000 [19, 20]	1.0000*	1.0000*	1.0000*	1.0000*
Two_Patterns	1.0000 [11, 20]	0.9968	0.9973	0.9968	0.9968
TwoLeadECG	1.0000 [8, 20]	0.9991	1.0000*	0.9991	1.0000*
uWavGest_X	0.8308 [16]	0.8490	0.8498	0.8481	0.8504*
uWavGest_Y	0.7585 [8]	0.7672*	0.7661	0.7658	0.7644
uWavGest_Z	0.7725 [16]	0.7973	0.7993	0.7982	0.8007*
uWavGestAll	0.9685 [20]	0.9618	0.9609	0.9626	0.9626
Wafer	1.0000 [19, 38]	0.9992	1.0000*	0.9981	0.9981
Wine	0.8890 [11]	0.8704	0.8890	0.9074*	0.9074*
WordsSynonyms	0.7790 [20]	0.6708	0.6991	0.6677	0.6677
Worms	0.8052 [19]	0.6685	0.6851	0.6575	0.6575
WormsTwoClass	0.8312 [38]	0.7956	0.8066	0.8011	0.8011
yoga	0.9183 [17]	0.9177	0.9163	0.9190	0.9237*
Count	-	43	65	51	57
MPCE	-	0.0318	0.0283	0.0301	0.0294
Arith. Mean	-	-	2.1529	-	2.5647
Geom. Mean	-	-	1.8046	-	1.8506


Input
Dimension Shuffle
Basic LSTM/ Attention LSTM
Dropout
Conv1D
BN+ReLU
128
Conv1D
BN+ReLU
256
Conv1D
BN+ReLU
128
Global Pooling
Concat
Softmax