针对时间序列分类（TSC）的深度模型

Deep Learning for Time Series Classification[1]

Xinyi Li

2019 年 11 月 20 日

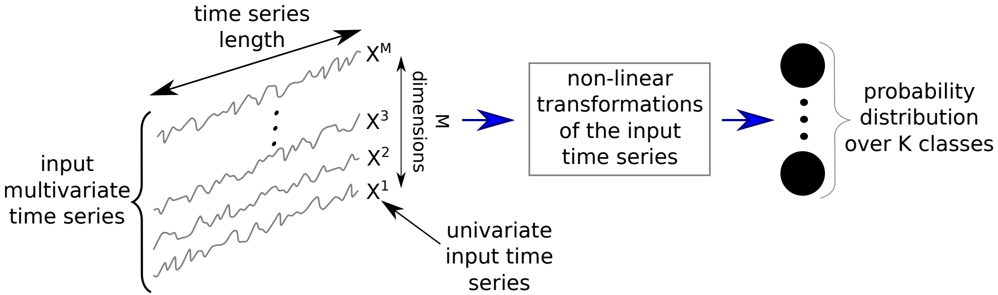
[1] Hassan Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (2019), pp. 917–963.

Overview

- 问题背景描述
- Strong Baseline
- 深度模型结构
  - MLP/DNN
  - FCN
  - ResNet
  - Encoder
  - MCNN
  - Time Le-Net
  - MCDCNN
  - Time-CNN
  - TWIESN
- 结论
  - Visualization Reasoning
- TSC 社区生态

Problem Description

univariance/multi-variance/panel



Different Univariate Learning Tasks

MTS

- different measurements of the **same** instance
- **high correlation**
- feeding features

panel data

- the same measurements on **different** instances
- i.i.d. assumption
- feeding sku/store/...

HIVE-COTE[2]:

SOTA classic algorithm[3]: Collective of Transformation based **Ensembles** (COTE) with a Hierarchical Vote system

1. Elastic Ensemble(**EE**): combination of 1-NN classifiers using different measurements
2. Shapelet Transform Ensemble(**ST**): top *k* shaplets (independent phase short pattern)
3. Bag-of-SFA-Symbols (**BOSS**) Ensemble: shapelets based on presence or absence
4. Time Series Forest (**TSF**): trained on selected $3\sqrt{m}$ features
5. Random Interval Features (**RIF**): spectral component of *Flat-COTE*

---

[2] Jason Lines, Sarah Taylor, and Anthony Bagnall. "Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification". In: *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE. 2016, pp. 1041–1046.

[3] Anthony Bagnall et al. "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances". In: *Data Mining and Knowledge Discovery* 31.3 (2017), pp. 606–660.

HIVE-COTE Algorithm[4]

ensemble of ensembles: ref to Flat-COTE

Algorithm 1 ProportionalEnsemble(*classifiers*, *train*, *test*)

```
trainAccs = ∅;
for i ← 1 to |classifiers| do
    trainAccs_i = loocv(train, classifiers[i])
    classifiers_i.buildClassifier(train)
testPreds = ∅
for i ← 1 to |test| do
    votes = ∅
    bsfWeight = −1;
    bsfClass = −1;
    for c ← 1 to |classifiers| do
        p = classifiers_c.classify(test_i)
        votes_p = votes_p + trainAccs_c;
        if votes_p > bsfWeight then
            bsfWeight = votes_p
            bsfClass = p
    testPreds_i = bsfClass
return testPreds
```

[4] https://github.com/TonyBagnall/py-hive-cote

MLP/DNN

Multi Layer Perceptrons/Fully-Connected(FC) Network

The Simplest DNN (e.g. `keras.Layers.Dense`)

$$\mathbf{X}_{i+1} = \sigma(\mathbf{W}_i \mathbf{X}_i + b_i)$$

Final (l-th) layer activate function: softmax

$$\hat{y}_k(\mathbf{X}_{l-1}) = (e^{\mathbf{W}_k \mathbf{X}_{l-1} + b_k}) / (\sum_{i=1}^{K} e^{\mathbf{W}_i \mathbf{X}_{l-1} + b_i})$$

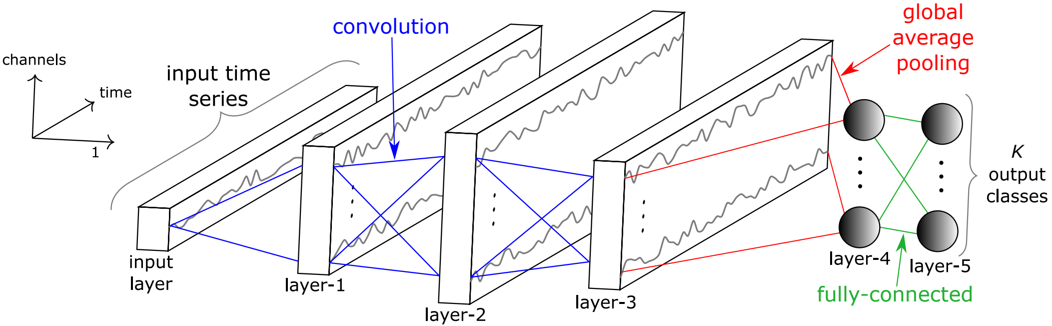
Objective loss: categorical cross entropy

$$Loss(\mathbf{X}) = -\sum_{i=1}^{K} y_i \log \hat{y}_i$$

minimized to learn the weights using **gradient descent** method

FCN[5]

Fully Convolutional Neural Network



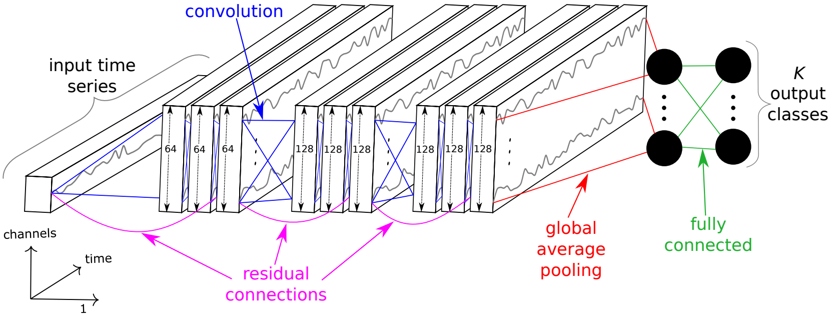
convolution layer ($\forall$ time stamp t shares filter ω with length l)

$$\mathbf{C}_t = \sigma(\omega * \mathbf{X}_{t-l/2:t+l/2} + b) | \forall t \in [1, T]$$

[5] John Cristian Borges Gamboa. "Deep learning for time-series analysis". In: *arXiv preprint arXiv:1701.01887* (2017).

ResNet[6]

Residual Network



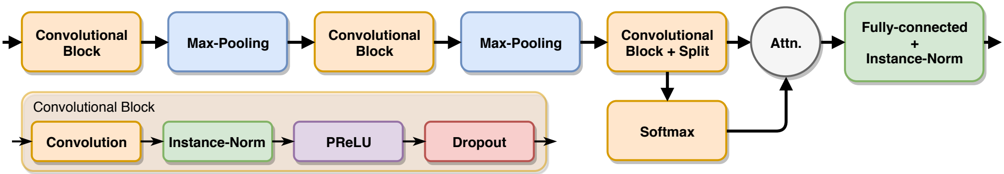
[6] Zhiguang Wang, Weizhong Yan, and Tim Oates. "Time series classification from scratch with deep neural networks: A strong baseline". In: *2017 international joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 1578–1585.

Encoder[7]

Hybrid deep CNN based on FCN

Modified from FCN

- GAP layer → attention layer (careful design for pre-train)
- normalization for each Conv layer output:
  1. ReLU → PReLU activation function (+ parameter)
  2. + dropout regularization
  3. + max pooling



[7] Joan Serrà, Santiago Pascual, and Alexandros Karatzoglou. "Towards a Universal Neural Network Encoder for Time Series.". In: *CCIA*. 2018, pp. 120–129.

MCNN[8]

Multi-scale Convolutional Neural Network

Similar to Traditional CNN:

1. 2 Conv layer (with max pooling)
2. 1 Fully-connected layer
3. final softmax layer

Heavy data pre-preprocessing step:

Window Slicing(WS)

for data augmentation:

1. slide a window over raw input
2. extract subsequences

Before training, $\forall$ subsequence

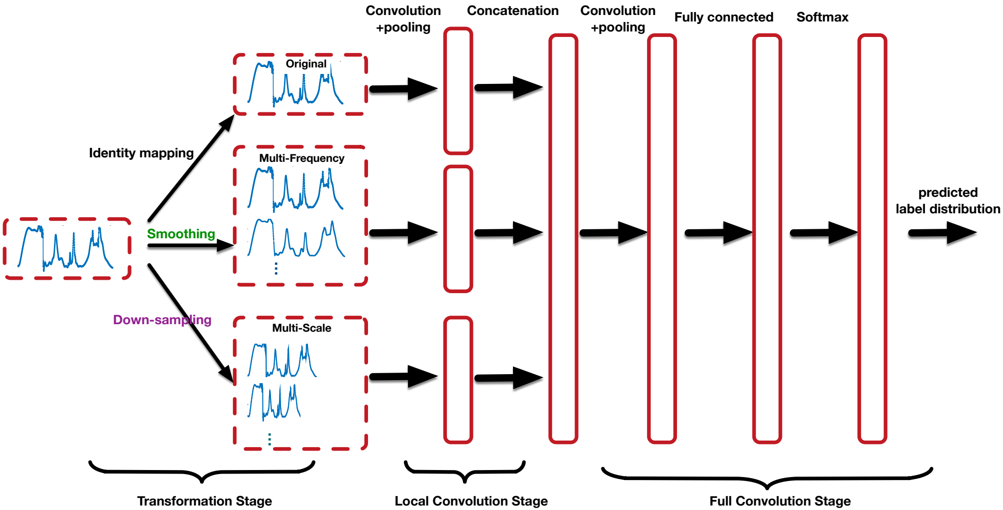
Transformations (parallel)

1. identity mapping
   - keep unchanged $\rightarrow$ 1-st Conv
2. down-sampling
   - $\rightarrow$ different shorter lengths subsequences
   - $\rightarrow$ 1-st Conv
3. smoothing:
   - equal length one
   - $\rightarrow$ 1-st Conv
   - $\rightarrow$ 2-nd Conv

[8] Zhicheng Cui, Wenlin Chen, and Yixin Chen. "Multi-scale convolutional neural networks for time series classification". In: *arXiv preprint arXiv:1603.06995* (2016).

MCNN

Framework



Time Le-Net[9]

Inspired by Le-Net[10],like CNN: 2 Conv + FC + final softmax

Compared to FCN:
GAP → FC

Local max pooling

- take max in a local pooling
- + invariance to small perturbations
- shorten a time series

Still #parameters ↑ #invariance ↓

Data augmentation to prevent overfitting
especially on relatively small datasets

Window Slicing(WS)

= method used in MCNN

Window Warping (WW)

For a time series with length l

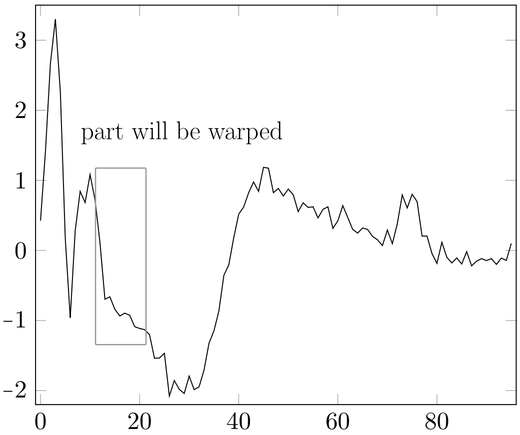
1. dilate $(\times 2) \rightarrow 2l$
2. squeeze $(\times \frac{1}{2}) \rightarrow \frac{1}{2}l$

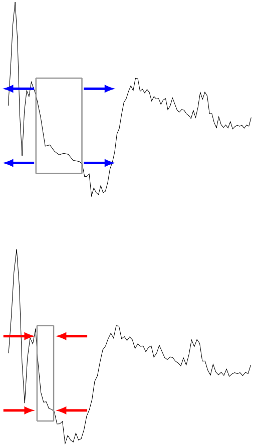
[9] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. "Data augmentation for time series classification using convolutional neural networks". In: 2016.

[10] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

Window Warping (WW)

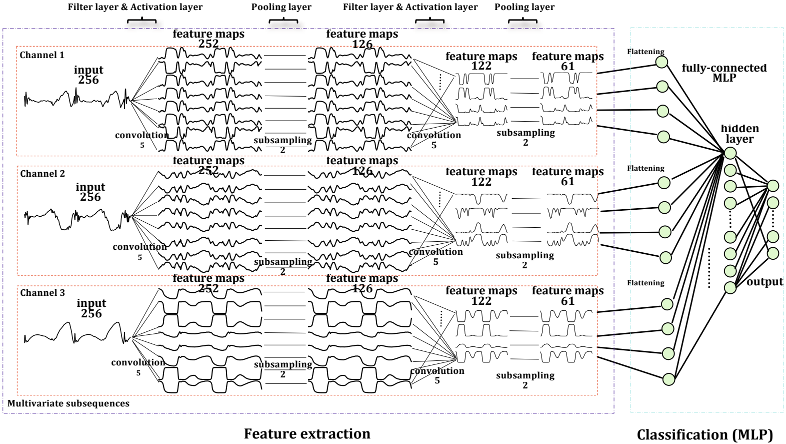
Then feed all data (length=l,$2l$,$\frac{1}{2}l$) into network





MCDCNN[11]

Multi Channel Deep Convolutional Neural Network: **Independent** Conv specified for **MTS**



[11] Yi Zheng et al. "Time series classification using multi-channels deep convolutional neural networks". In: *International Conference on Web-Age Information Management*. Springer. 2014, pp. 298–310.

Time-CNN[12]

both for univariate and multivariate

Main differences compared to previous models:

1. loss function: categorical cross-entropy $\rightarrow$ MSE
2. activate function of final layer: softmax $\rightarrow$ sigmoid

$$\sum_{i=1}^{K} p(\hat{Y}_i) \neq 1$$

3. throughout CNN: local max pooling $\rightarrow$ local average pooling

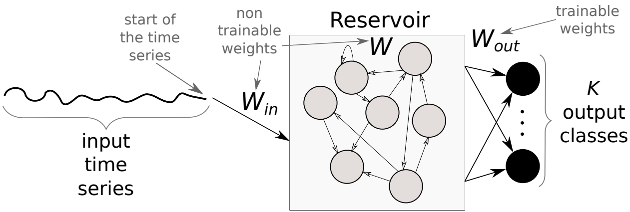
Compared to some models:

- apply 1 Conv for all dimensions (unlike MCDCNN)
- Conv directly fully connected to final layer (modify FCN: GAP $\rightarrow$ FC)

[12] Bendong Zhao et al. "Convolutional neural networks for time series classification". In: *Journal of Systems Engineering and Electronics* 28.1 (2017), pp. 162–169.

Echo State Network (ESN)[13]

Based on RNN



Core part: reservoir

- sparsely connected random RNN
- each neuron create its own nonlinear activation of the incoming signal

Internal state at time t : $I(t) \in R^{N_r}$, $N_r = \#$neurons inside reservoir

$$I(t) = \sigma(W_{\text{in}} X(t) + W I(t-1)) | \forall t \in [1, T]$$

Finally

$$\hat{Y}(t) = W_{\text{out}} I(t)$$

[13] Claudio Gallicchio and Alessio Micheli. "Deep echo state network (deepesn): A brief survey". In: *arXiv preprint arXiv:1712.04323* (2017).

TWIESN[14]

Time Warping Invariant Echo State Network

ESN orginally proposed for time series forecasting $\rightarrow$ predicts a probability distribution

Training

1. reservoir space ($\forall t$): $X(t) \rightarrow$ **higher** dimensional space
2. train a Ridge classifier to predict the class of each $X(t)$

Testing

1. $\forall X(t)$, trained Ridge classifier predicts a probability distribution
2. for each class k, calculate a posteriori probability
3. label k with the largest average probability

$$\arg\max_{k} \frac{1}{T} \sum_{t=1}^{T} \hat{Y}_k(t)$$

[14] Pattreeya Tanisaro and Gunther Heidemann. "Time series classification using time warping invariant echo state networks". In: *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2016, pp. 831–836.

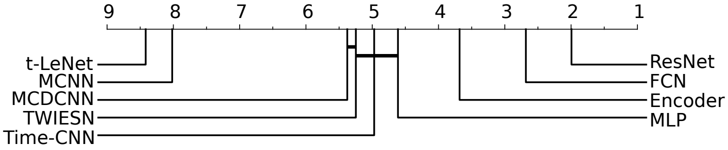
Summary

Methods	Architecture							
	#Layers	#Conv	#Invar	Normalize	Pooling	Feature	Activate	Regularize
MLP	4	0	0	None	None	FC	ReLU	Dropout
FCN	5	3	4	Batch	None	GAP	ReLU	None
ResNet	11	9	10	Batch	None	GAP	ReLU	None
Encoder	5	3	4	Instance	Max	Att	PReLU	Dropout
MCNN	4	2	2	None	Max	FC	Sigmoid	None
t-LeNet	4	2	2	None	Max	FC	ReLU	None
MCDCNN	4	2	2	None	Max	FC	ReLU	None
Time-CNN	3	2	2	None	Avg	Conv	Sigmoid	None

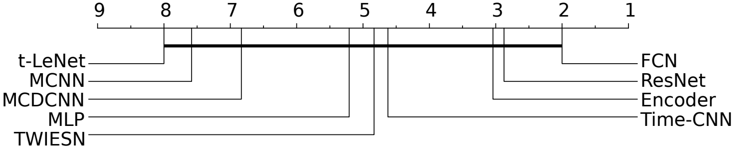
Overall Performance

Critical Difference Diagram (dataset: UCR/UEA)

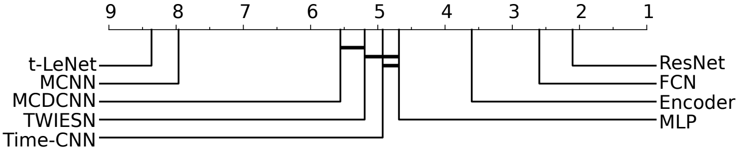
univariate



multivariate



both



For Different Datasets

Themes (#)	MLP	FCN	ResNet	Encoder	MCNN	t-LeNet	MCDCNN	Time-CNN	TWIESN
DEVICE (6)	0.0	50.0	**83.3**	0.0	0.0	0.0	0.0	0.0	0.0
ECG (7)	14.3	**71.4**	28.6	42.9	0.0	0.0	14.3	0.0	0.0
IMAGE (29)	6.9	34.5	**48.3**	10.3	0.0	0.0	6.9	10.3	0.0
MOTION (14)	14.3	28.6	**71.4**	21.4	0.0	0.0	0.0	0.0	0.0
SENSOR (16)	6.2	37.5	**75.0**	31.2	6.2	6.2	6.2	0.0	12.5
SIMULATED (6)	0.0	33.3	**100.0**	33.3	0.0	0.0	0.0	0.0	0.0
SPECTRO (7)	14.3	14.3	**71.4**	0.0	0.0	0.0	0.0	28.6	28.6

Length	MLP	FCN	ResNet	Encoder	MCNN	t-LeNet	MCDCNN	Time-CNN	TWIESN
<81	5.43	3.36	**2.43**	2.79	8.21	8.0	3.07	3.64	5.5
81-250	4.16	**1.63**	1.79	3.42	7.89	8.32	5.26	4.47	5.53
251-450	3.91	2.73	**1.64**	3.32	8.05	8.36	6.0	4.68	4.91
451-700	4.85	2.69	**1.92**	3.85	7.08	7.08	5.62	4.92	4.31
701-1000	4.6	1.9	**1.6**	3.8	7.4	8.5	5.2	6.0	4.5
>1000	3.29	2.71	**1.43**	3.43	7.29	8.43	4.86	5.71	6.0

Train size	MLP	FCN	ResNet	Encoder	MCNN	t-LeNet	MCDCNN	Time-CNN	TWIESN
<100	4.3	2.03	**1.67**	4.13	7.67	7.73	6.1	4.37	4.77
100-399	4.85	2.76	**2.06**	3.24	7.71	8.12	4.59	4.97	4.5
400-799	3.62	2.38	**1.75**	3.5	8.0	8.62	4.38	5.0	5.88
>799	3.85	2.85	**1.62**	2.08	7.92	8.69	4.62	4.85	6.92

Class Activation Map (CAM)[15]

GAP layer + softmax

univariate time series $A_m(t)$ produced by m-th filter in Last Conv, input to neuron of class c

$$
\begin{aligned}
z_c &= \sum_m {w_m}^c \sum_t A_m(t) \\
&= \sum_m \sum_t {w_m}^c A_m(t)
\end{aligned}
$$

Therefore, CAM_c that explains the classification as label c

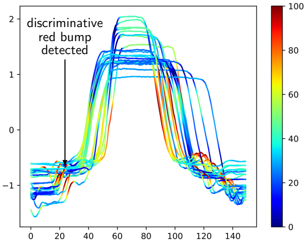
$$
CAM_c(t) = \sum_m {w_m}^c A_m(t)
$$

constructed a new time series for further observation and explaination.

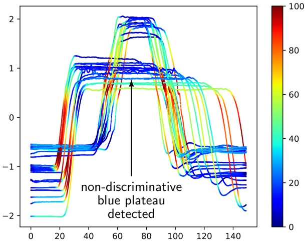
[15] Bolei Zhou et al. "Learning deep features for discriminative localization". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2921–2929.

Find discriminative Region

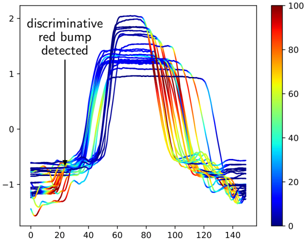
Example: FCN/ResNet on GunPoint dataset (achieve 100% accurancy)



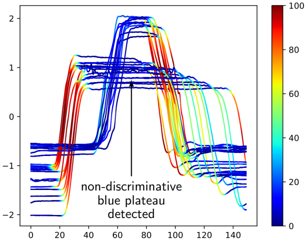
(a) FCN on GunPoint: Class-1



(b) FCN on GunPoint: Class-2



(c) ResNet on GunPoint: Class-1



(d) ResNet on GunPoint: Class-2

Libraries/Implements/Community

sktime[16] & its extensions

Sktime

- based on classic models (shallow)
- `scikit-learn` interface compatible

Sktime-dl

- use `Keras` to implement all 9 deep models above
- 暂时不能直接安装（MacOS）

UEA & UCR Time Series Classification Repository

- 128 TSC datasets + 30 MTS datasets
- Collect a bunch of **classic** algorithms

---

[16] Markus Löning et al. *sktime: A Unified Interface for Machine Learning with Time Series*. 2019. eprint: `arXiv:1909.07872`.

Thanks

All codes, slides and papers available

li-xin-yi/deep_time_series_share_slide