

# 针对时间序列分类（TSC）的深度模型

Deep Learning for Time Series Classification<sup>1</sup>

Xinyi Li

2019 年 11 月 21 日

---

<sup>1</sup>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

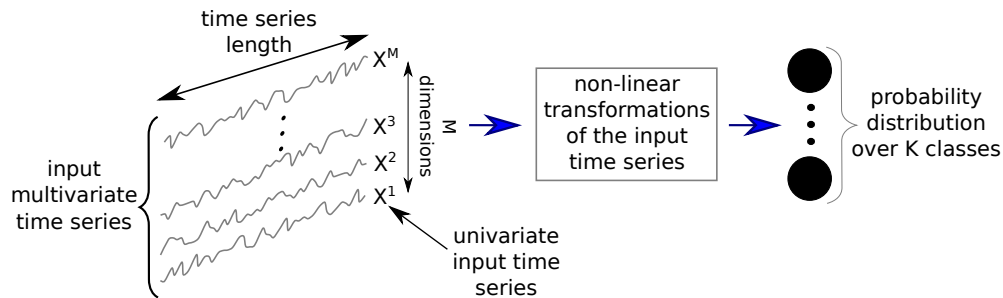
TWIESN

结论

Visualization Reasoning

TSC 社区生态

# Problem Description



# Different Learning Tasks

Univariate/Multi-variate/Panel

## Univariate

1. Forecasting (e.g. fbprophet)
2. Regression
3. Classification

## Multivariate

### 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<sup>2</sup>:

**SOTA** classic algorithm<sup>3</sup>: 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  $k$  shapelets (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*

---

<sup>2</sup>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.

<sup>3</sup>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<sup>4</sup>

ensemble of ensembles: ref to Flat-COTE

---

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

---

```
1: trainAccs =  $\emptyset$ ;  
2: for  $i \leftarrow 1$  to  $|classifiers|$  do  
3:   trainAccsi = loocv(train, classifiers[i])  
4:   classifiersi.buildClassifier(train)  
5: testPreds =  $\emptyset$   
6: for  $i \leftarrow 1$  to  $|test|$  do  
7:   votes =  $\emptyset$   
8:   bsfWeight = -1;  
9:   bsfClass = -1;  
10:  for  $c \leftarrow 1$  to  $|classifiers|$  do  
11:     $p = classifiers_c.classify(test_i)$   
12:    votesp = votesp + trainAccsc;  
13:    if votesp > bsfWeight then  
14:      bsfWeight = votesp  
15:      bsfClass = p  
16:    testPredsi = bsfClass  
17: return testPreds
```

---

---

<sup>4</sup><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

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

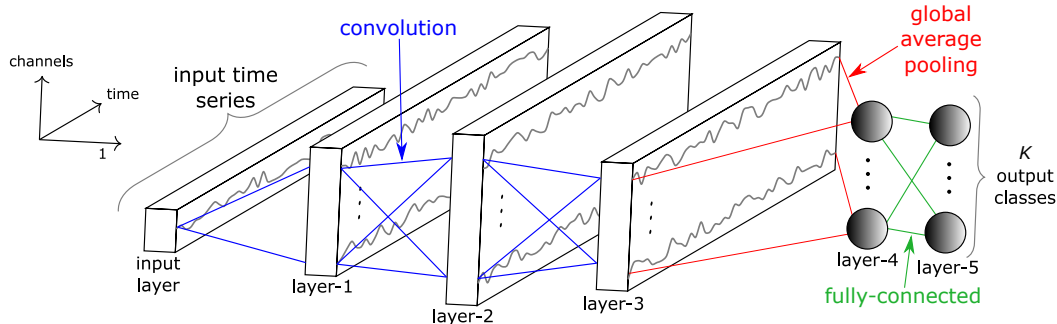
Objective loss: categorical cross entropy

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

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

# FCN<sup>5</sup>

## Fully Convolutional Neural Network



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

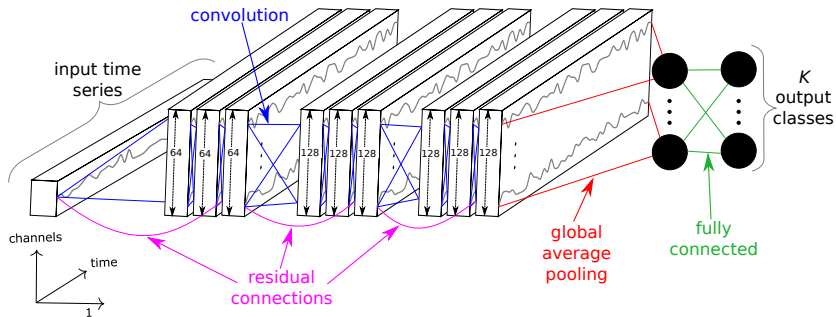
$$\mathbf{C}_t = \sigma(\omega * \mathbf{X}_{t-l/2:t+l/2} + b) | \forall t \in [1, T]$$

<sup>5</sup> John Cristian Borges Gamboa. "Deep learning for time-series analysis". In: *arXiv preprint arXiv:1701.01887* (2017).



# ResNet<sup>6</sup>

## Residual Network



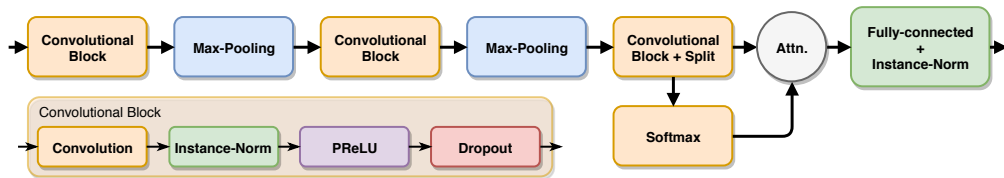
<sup>6</sup>Zhiguang Wang, Weizhong Yan, and Tim Oates. "Time series classification from scratch with deep neural networks: A strong baseline". In: *2017 international joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 1578–1585.

# Encoder<sup>7</sup>

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



<sup>7</sup>Joan Serrà, Santiago Pascual, and Alexandros Karatzoglou. “Towards a Universal Neural Network Encoder for Time Series.”. In: *CCIA*. 2018, pp. 120–129.

# MCNN<sup>8</sup>

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

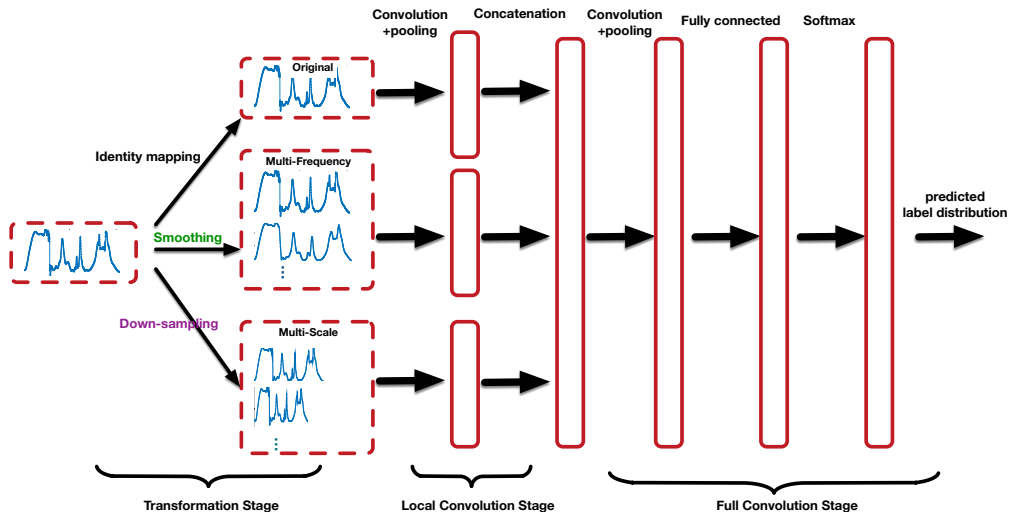
## Transformations (parallel)

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

<sup>8</sup>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<sup>9</sup>

Inspired by Le-Net<sup>10</sup>, like CNN: 2 Conv + FC + final softmax

Compared to FCN:

GAP  $\rightarrow$  FC

Local max pooling

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

Still #parameters  $\uparrow$  #invariance  $\downarrow$

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  $l$

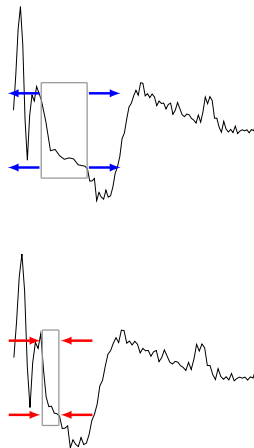
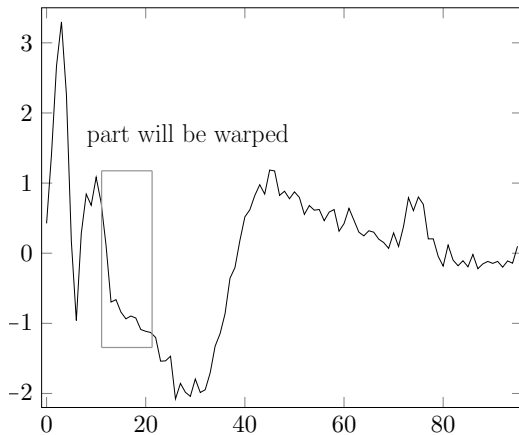
1. dilate ( $\times 2$ )  $\rightarrow 2l$
2. squeeze ( $\times \frac{1}{2}$ )  $\rightarrow \frac{1}{2}l$

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

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

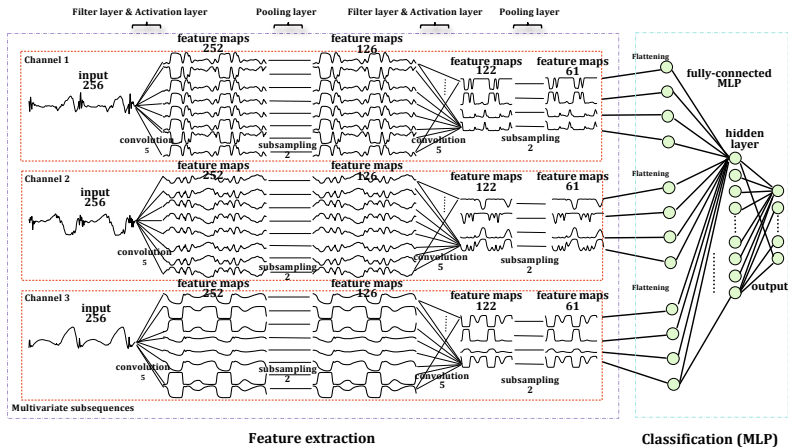
# Window Warping (WW)

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



# MCDCNN<sup>11</sup>

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



<sup>11</sup>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<sup>12</sup>

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 MCD CNN)
- ▶ Conv directly fully connected to final layer (modify FCN: GAP  $\rightarrow$  FC)

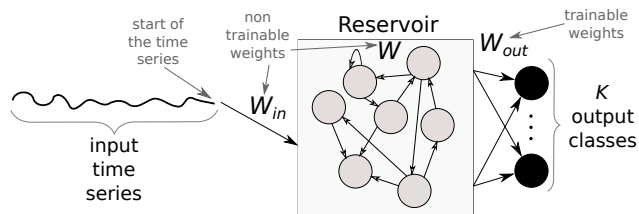
---

<sup>12</sup>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)<sup>13</sup>

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  $t$  :  $I(t) \in R^{N_r}$ ,  $N_r = \# \text{neurons inside reservoir}$

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

Finally

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

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

# TWIESN<sup>14</sup>

## Time Warping Invariant Echo State Network

ESN originally proposed for time series forecasting → 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

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

---

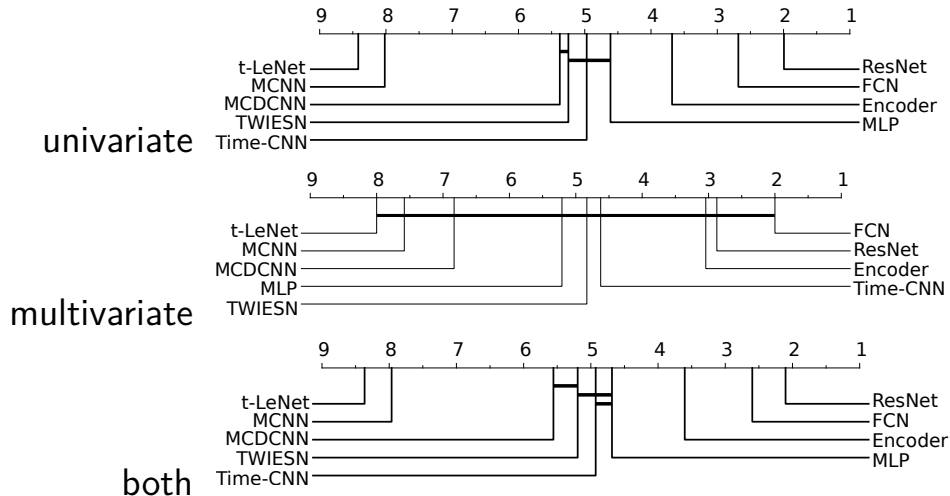
<sup>14</sup>Pattreeya Tanisaro and Gunther Heidemann. "Time series classification using time warping invariant echo state networks". In: *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2016, pp. 831–836.

# 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)



## For Different Datasets

Themes (#)	MLP	FCN	ResNet	Encoder	MCNN	t-LeNet	MCDCNN	Time-CNN	TWIESN
DEVICE (6)	0.0	50.0	<b>83.3</b>	0.0	0.0	0.0	0.0	0.0	0.0
ECG (7)	14.3	<b>71.4</b>	28.6	42.9	0.0	0.0	14.3	0.0	0.0
IMAGE (29)	6.9	34.5	<b>48.3</b>	10.3	0.0	0.0	6.9	10.3	0.0
MOTION (14)	14.3	28.6	<b>71.4</b>	21.4	0.0	0.0	0.0	0.0	0.0
SENSOR (16)	6.2	37.5	<b>75.0</b>	31.2	6.2	6.2	6.2	0.0	12.5
SIMULATED (6)	0.0	33.3	<b>100.0</b>	33.3	0.0	0.0	0.0	0.0	0.0
SPECTRO (7)	14.3	14.3	<b>71.4</b>	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	<b>2.43</b>	2.79	8.21	8.0	3.07	3.64	5.5
81-250	4.16	<b>1.63</b>	1.79	3.42	7.89	8.32	5.26	4.47	5.53
251-450	3.91	2.73	<b>1.64</b>	3.32	8.05	8.36	6.0	4.68	4.91
451-700	4.85	2.69	<b>1.92</b>	3.85	7.08	7.08	5.62	4.92	4.31
701-1000	4.6	1.9	<b>1.6</b>	3.8	7.4	8.5	5.2	6.0	4.5
>1000	3.29	2.71	<b>1.43</b>	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	<b>1.67</b>	4.13	7.67	7.73	6.1	4.37	4.77
100-399	4.85	2.76	<b>2.06</b>	3.24	7.71	8.12	4.59	4.97	4.5
400-799	3.62	2.38	<b>1.75</b>	3.5	8.0	8.62	4.38	5.0	5.88
>799	3.85	2.85	<b>1.62</b>	2.08	7.92	8.69	4.62	4.85	6.92

# Class Activation Map (CAM)<sup>15</sup>

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  $c$

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

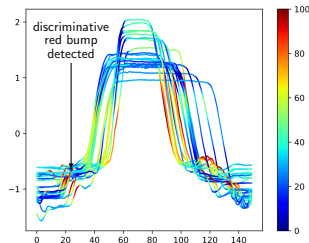
constructed a new time series for further observation and explanation.

---

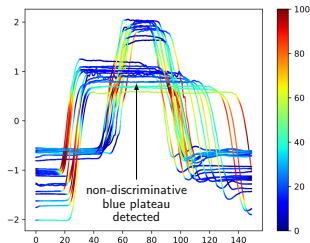
<sup>15</sup>Bolei Zhou et al. "Learning deep features for discriminative localization". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2921–2929.

# Find discriminative Region

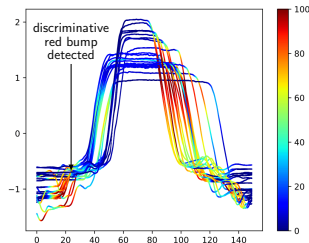
Example: FCN/ResNet on GunPoint dataset (achieve 100% accuracy)



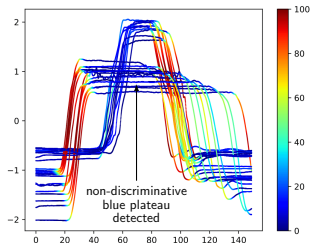
(a) FCN on GunPoint: Class-1



(b) FCN on GunPoint: Class-2



(c) ResNet on GunPoint: Class-1



(d) ResNet on GunPoint: Class-2



# Libraries/Implements/Community

sktime<sup>16</sup> & 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

---

<sup>16</sup>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](https://github.com/li-xin-yi/deep_time_series_share_slide)