Class-Specific Attention (CSA) for Time-Series Classification

Yifan Hao, Huiping Cao, K. Selçuk Candan,
Jiefei Liu, Huiying Chen

Presenter: Huiping Cao

Department of Computer Science

New Mexico State University

Time series classification

- Time series data describe how variables evolve over time
  - Example of time series data: temperature, CPU utilization, accelerometer data, power voltage/frequency waveforms
- Classification
  - Examples: predict human/animal behavior, power disturbances

Notation

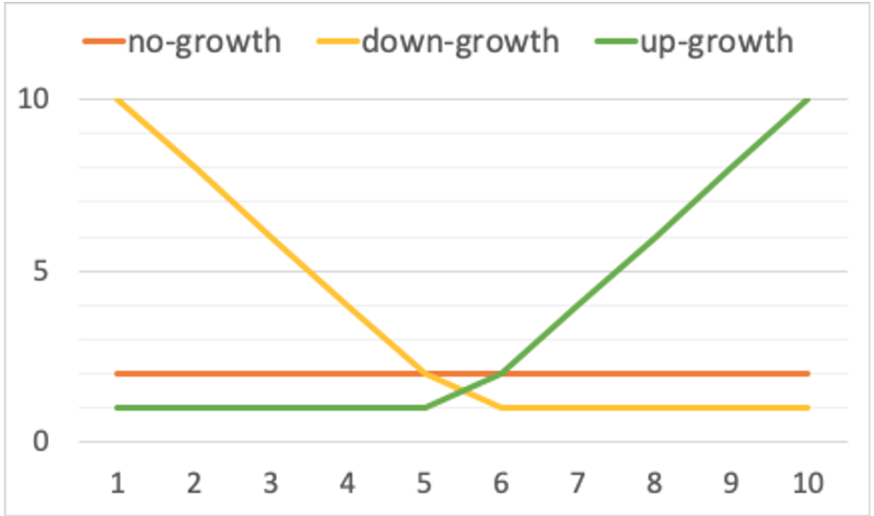
- One time series instance: x ∈ $R^{T \times V}$
  - T: # of timestamps (or length)
  - V: # of variables
- Time series dataset: X = {x_1, x_2, …, x_N} ∈ $R^{N \times T \times V}$
  - N: # of instances (or samples)
- Time series classification
  - Input: (X, Y) where $X \in R^{N \times T \times V}$ and $Y \in R^N$ each x_i corresponds to a label y_i
  - Output a model f: X-> Y (to make predictions for future X')

Existing Solutions

- Convolutional Neural Networks (CNNs) based models (e.g., [17, 19, 22])
- Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) based models (e.g., [5, 9, 13, 14])

- Introduction of attention mechanism [2]

Problems with Existing Models – Toy Data



No-growth: A company with stable stock prices

Down-growth:
A company with stock price decreasing.

Up-growth:
A company with stock price increasing.

A subsequence of a time series sequence can be more helpful to differentiate an instance from belonging to one class to belonging to other classes.

Challenges

To leverage **Class-Specific Features (CSF)** in neural network (NN)

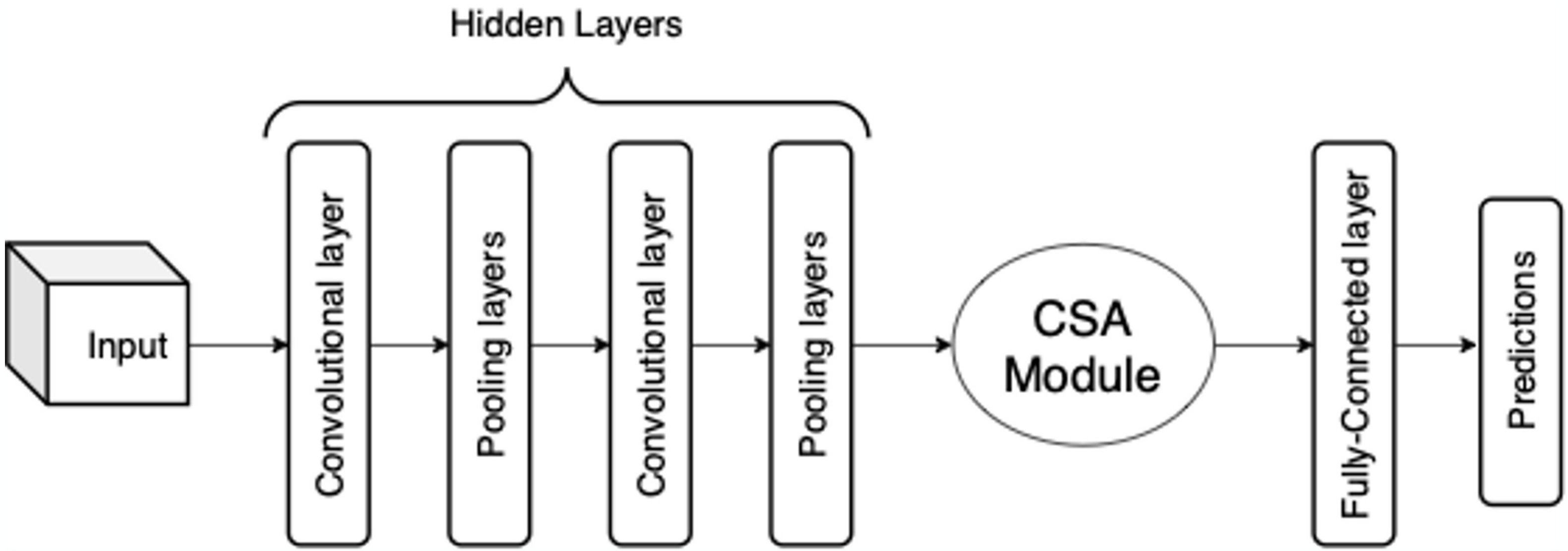
1. How to accurately identify CSF information
   a. CSF are used to separate one class from the rest of classes
   b. A general significant features may not be a good CSF.
2. How to leverage label information
   a. The label information is not well studied in most NN models
   b. Label information is not available during testing

Class-Specific Attention (CSA) Module

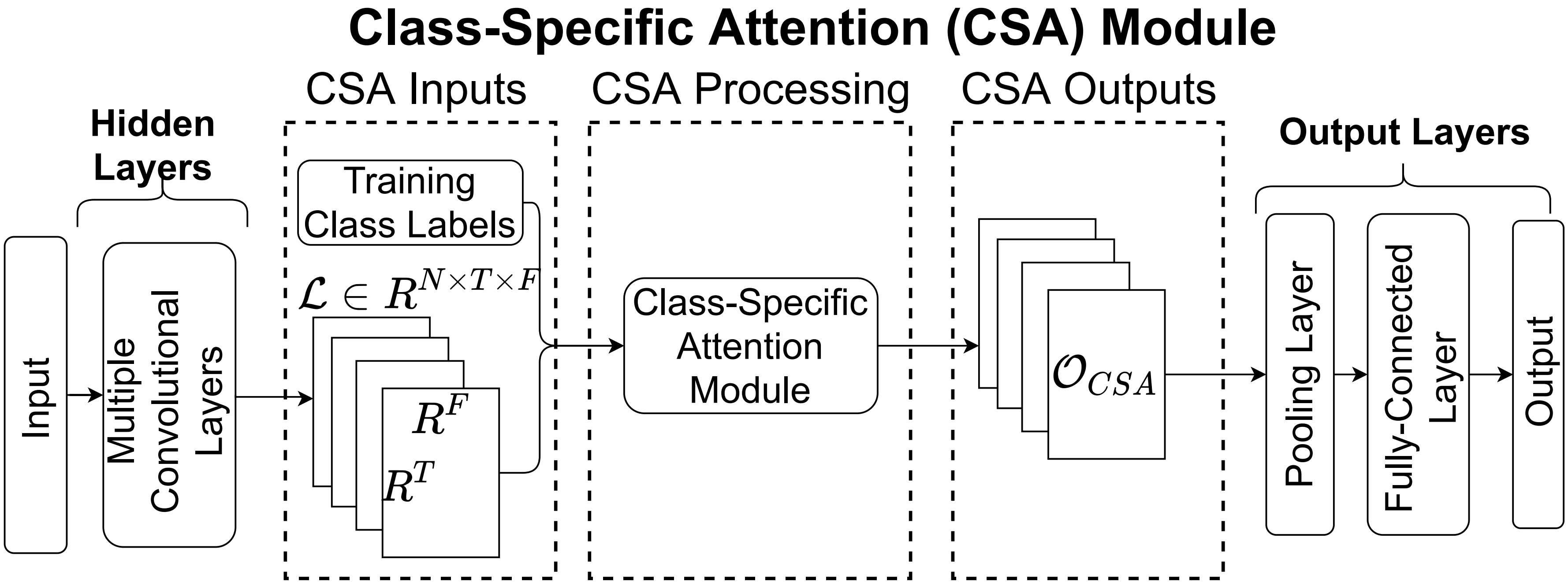
- Propose **a CSA module** which can be embedded in most NN models without post-processing and retraining

- One model can learn class-specific features and make classifications

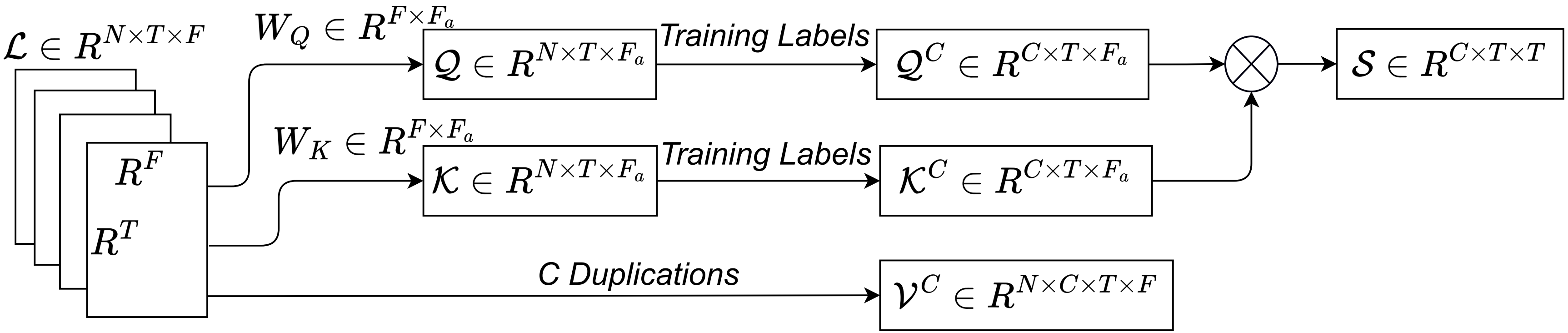
Class-Specific Attention (CSA) Module



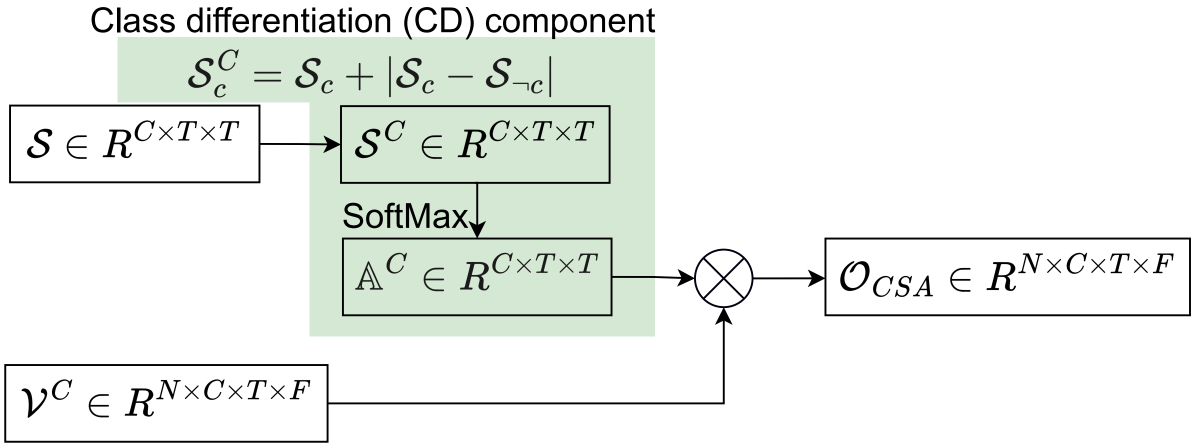
Improved: Class-Specific Attention (CSA) Module



Improved: Class-Specific Attention (CSA) Module



Improved: Class-Specific Attention (CSA) Module



Major design differences

- Class specific features
  - CSA is designed to learn the class-specific features in the training stage and preserve these features in the *hidden query space*.
  - The design of the query space can be directly utilized in the testing stage without knowing the class labels of testing instances.
- Attention value calculation
  - Typical attention mechanisms calculate attention values by using the similarity between the *key features* and the *query features*
  - In contrast, attention calculation in the CSA module differentiates the importance of *class-specific features* from *other features*.

Experiments: Effectiveness on UTS datasets

1. Datasets: 40 benchmark datasets (28 MTS and 12 UTS)
2. Baseline models
   a. Fully Convolutional Networks (FCN) [17]
   b. Multivariate Long Short-Term Memory (MLSTM) [9]
   c. MLSTM-FCN [11]
   d. Convolutional Neural Networks with Attention (CNN-ATN) [6]
   e. TapNet (with Class-Specific features) [20]
3. Measurements
   a. Improvements that Model (A) has over Model (B)

$$AI(A,B) = \frac{Acc_A - Acc_B}{Acc_B}$$

Experiments: Effectiveness on MTS datasets

Datasets	AI_{FCN}	AI_{MLSTM}	$AI_{MLSTM\text{-}FCN}$	AI_{TapNet}	$AI_{CNN\text{-}ATN}$
ArtWordRec	**0.204**	**4.902**	**1.029**	**0.816**	**0.407**
BasicMotions	-0.207	**0.948**	**0.207**	-0.600	**1.833**
CharTraj	0.000	**0.616**	**0.403**	0.000	0.000
Cricket	0.000	**0.568**	**2.050**	-1.895	**2.228**
DuckDuckGeese	**3.514**	**3.274**	**1.662**	-6.630	-
EigenWorms	**1.471**	0.000	**9.125**	-3.333	-
Epilepsy	**6.045**	**3.003**	**1.157**	-5.517	**0.421**
EthanolConc	**5.449**	**1.170**	**5.145**	**13.043**	-1.923
FaceDetection	0.000	**0.692**	**1.079**	**1.423**	-
FingerMovements	**1.235**	0.000	0.000	**3.583**	**2.990**
HandMovement	**4.274**	**11.892**	**5.579**	**12.069**	**4.739**
Handwriting	**1.408**	**7.692**	**4.965**	**40.404**	**0.676**
Heartbeat	**0.739**	**0.244**	**0.741**	-2.139	0.000
InsectWingbeat	**27.778**	0.000	**17.241**	-	-
JapaneseVowels	**0.907**	**1.493**	**0.215**	**0.405**	0.000

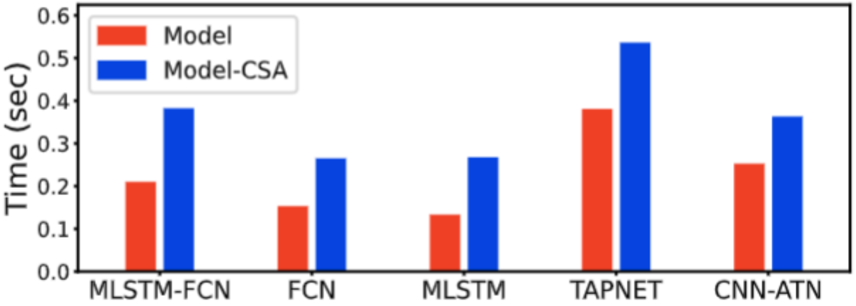
Datasets	AI_{FCN}	AI_{MLSTM}	$AI_{MLSTM\text{-}FCN}$	AI_{TapNet}	$AI_{CNN\text{-}ATN}$
LSST	**2.703**	**0.379**	**5.283**	**42.667**	**1.506**
Libras	**0.443**	**28.261**	-0.442	**3.797**	0.000
MotorImagery	**1.250**	**2.632**	0.000	-3.459	-
NATOPS	**1.354**	**3.535**	**1.814**	0.000	**0.423**
PEMS-SF	**1.720**	**22.222**	**1.724**	0.000	-
PenDigits	-0.203	0.000	0.000	0.000	0.000
Phoneme	**4.545**	**6.000**	**18.750**	**31.707**	-
RacketSports	**3.476**	**1.542**	**4.416**	**15.909**	0.000
SelfRegSCP1	**0.229**	**0.668**	**0.452**	-13.587	**5.783**
SelfRegSCP2	**3.169**	**4.196**	0.000	**2.076**	**2.198**
SpokenArab	**0.821**	0.000	**0.407**	0.000	0.000
StandWalkJump	0.000	-5.780	-3.271	**15.748**	-
UWaveGesture	-0.962	**0.737**	-0.260	-0.907	-0.477
Wins	**21/28**	**22/28**	**21/28**	**13/27**	**11/20**
Average	**2.549**	**3.603**	**2.838**	**5.392**	**1.040**

Experiments: Effectiveness on UTS datasets

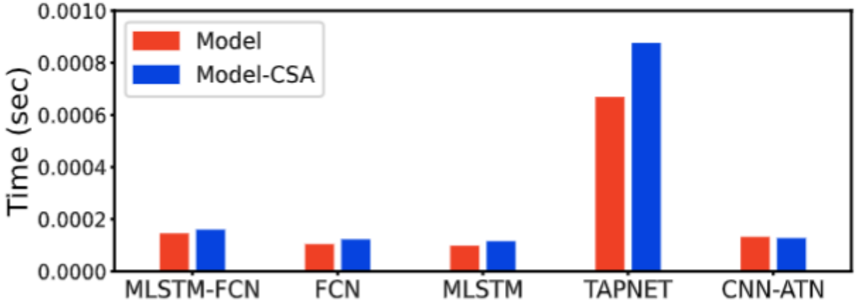
Datasets	AI_{FCN}	AI_{LSTM}	$AI_{LSTM\text{-}FCN}$	AI_{TapNet}	$AI_{CNN\text{-}ATN}$
MedicalImages	**0.253**	**11.111**	**0.509**	**0.267**	**1.044**
MelPedestrian	**0.909**	**8.434**	**0.251**	**3.788**	**1.412**
MidPhalanxoutGrp	**5.298**	**35.514**	-0.323	**0.608**	**2.096**
MidPhalanxoutCor	**0.246**	0.000	**0.247**	**0.474**	**0.242**
MidPhalanxtw	0.000	**20.961**	**3.200**	-1.294	**0.370**
OsuLeaf	**0.828**	0.000	**1.245**	**1.029**	**2.004**
PhalangesCor	**0.249**	0.000	**1.003**	**2.332**	**1.985**
Powercons	**0.432**	**1.277**	**0.648**	**1.695**	**1.089**
ProximalPhaGrp	**0.955**	**25.085**	**0.238**	-0.234	**0.235**
ProximalPhaCor	**2.128**	**4.348**	**3.118**	**0.671**	-1.089
ProximalPhaTw	-2.151	**12.262**	**3.261**	**0.756**	**1.222**
RefrigerationDev	-0.769	**13.725**	**1.515**	0.000	-2.405
Wins	**9/12**	**9/12**	**11/12**	**9/12**	**10/12**
Average	**0.644**	**11.251**	**1.175**	**0.776**	**0.631**

BE BOLD. Shape the Future.

Experiments: Effectiveness on UTS datasets



(a) Training time per iteration



(b) Testing time per instance

Conclusions

- We present **a class-specific attention (CSA) module** to improve the classification performance of neural network models for time series classification.
- CSA identifies class-specific features **leveraging training labels**, while avoiding the need to access label information during testing phase.
- This is the **first attention design that leverages the class label** information in the hidden layers to generate class-specific features.
- The CSA module was embedded to **five state-of-the-art** time series classification NN models and tested on **40 benchmark datasets** to demonstrate its superiority.

Q & A

Feel free to check our paper [1] if you have any questions.

References

- [1] Yifan Hao, Huiping Cao, K. Selcuk Candan, Jiefei Liu, Huiying Chen: Class-Specific Attention (CSA) for Time-Series Classification. Peer reviewed. Published in the The SIGKDD Workshop on Mining and Learning from Time Series. https://kdd-milets.github.io/milets2022/papers/MILETS_2022_paper_2604.pdf
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine TranslationbyJointlyLearningtoAlignandTranslate.InICLR. http://arxiv.org/ abs/1409.0473
- [5] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Ben- gio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. CoRR abs/1409.1259 (2014). http://arxiv.org/abs/1409.1259
- [6] Yifan Hao and Huiping Cao. 2020. A New Attention Mechanism to Classify Multivariate Time Series. In IJCAI. 1999–2005. https://doi.org/10.24963/ijcai.2020/277
- [9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. NeuralComputation9,8(1997),1735–1780. https://doi.org/10.1162/neco.1997.9. 8.1735
- [11] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford.2019. Multivariate LSTM-FCNs for time series classification. Neural Networks 116 (2019),237–245. https://doi.org/10.1016/j.neunet.2019.04.014

- [13] Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to Construct Deep Recurrent Neural Networks. In 2nd International Confer- enceonLearningRepresentations(ICLR). http://arxiv.org/abs/1312.6026
- [14] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. 2017. A Dual-Stage Attention-Based Recurrent Neural Network for Time SeriesPrediction.InIJCAI.2627–2633. https://doi.org/10.24963/ijcai.2017/366
- [17] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In International Joint Conference on Neural Networks (IJCNN). 1578–1585. https://doi.org/10.1109/ IJCNN.2017.7966039
- [19] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. 2015. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition. In IJCAI. 3995–4001. http: //ijcai.org/Abstract/15/561
- [20] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. 2020. TapNet: Multivariate Time Series Classification with Attentional Prototypical Network. In The Thirty-Fourth Conference on Artificial Intelligence, AAAI 2020, February 7-12,2020.6845–6852. https://aaai.org/ojs/index.php/AAAI/article/view/6165
- [22] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao. 2016. Exploiting multi-channels deep convolutional neural networks for multivariate time series classification.FrontiersComput.Sci.10,1(2016),96–112. https://doi.org/10.1007/ s11704- 015- 4478- 2

Backup slides

Symbol	Meaning
N	# of instances in a TS dataset
C	# of distinct classes in a dataset
V	# of variables in a TS dataset
T	# of time points of one time series in X
$\mathcal{L}$	Feature matrix from hidden layers
F	# of features at each time point in X
F_a	# of features in the hidden key and query spaces
B	# of instances in one batch

Table 1: Symbols used in this paper

Algorithm 1 *CSA_Calculation*

Input: $\mathcal{L} \in R^{N \times T \times F}$: feature tensor from the hidden layers
Output: $O_{CSA} \in R^{N \times C \times T \times F}$: feature tensor adjusted using class-specific attention

$\mathcal{K} = \mathcal{L} \cdot W_K$ where $W_K \in R^{F \times F_a}$
$\mathcal{Q} = \mathcal{L} \cdot W_Q$ where $W_Q \in R^{F \times F_a}$
Initialize $\mathcal{K}^C \in R^{C \times T \times F}$, $\mathcal{Q}^C \in R^{C \times T \times F}$
for each class label c **do**
  $\mathcal{L}_c$ = all the instances belonging to class c.
  $\mathcal{K}^C[c, :, :] = average(\mathcal{K}[\mathcal{L}_c, :, :])$ on the first dimension
  $\mathcal{Q}^C[c, :, :] = average(\mathcal{Q}[\mathcal{L}_c, :, :])$ on the first dimension
end for
$\mathcal{S} = \mathcal{K}^C \cdot (\mathcal{Q}^C)^T$ where $\mathcal{S} \in R^{C \times T \times T}$
Initialize $\mathcal{S}^C \in R^{C \times T \times T}$
for each class label c **do**
  $\mathcal{S}_c = \mathcal{S}[c, :, :]$
  $\mathcal{S}_{\neg c} = \frac{SUM(\mathcal{S}) - \mathcal{S}_c}{C-1}$
  $\mathcal{S}^C[c, :, :] = \mathcal{S}_c + abs(\mathcal{S}_c - \mathcal{S}_{\neg c})$
end for
$\mathbb{A}^C = SoftMax(\mathcal{S}^C) \in R^{C \times T \times T}$
$\mathcal{V} = \mathcal{L} \cdot W_V$, where $W_V \in R^{F \times F}$
$\mathcal{V}^C$ = shape C copies of $\mathcal{V}$ to $R^{N \times C \times T \times F}$
$\mathcal{L}^C$ = shape C copies of $\mathcal{L}$ to $R^{N \times C \times T \times F}$
$O_{CSA} = \mathcal{L}^C + \sigma \times (\mathbb{A}^C \cdot \mathcal{V}^C)$
return O_{CSA}

Testing stage

- The class-specific features OCSA for a testing instance are calculated by directly utilizing the global class-specific attention A^{C} and the value features of this instance.
    - Note that the testing instance does not need any class label to leverage class-specific attention.
- Next, these class-specific features O$_{CSA}$ are passed to the specially designed fully connected layer described above to make class predictions.