Postprint of: Xiao Z., Xu X., Zhang H., Szczerbicki E., A new multi-process collaborative architecture for time series classification, KNOWLEDGE-BASED SYSTEMS, Vol. 220 (2021), 106934, DOI: 10.1016/j.knosys.2021.106934

A new multi-process collaborative architecture for time series classification

Zhiwen Xiao[a], Xin Xu[b], Haoxi Zhang[a,*] and Edward Szczerbicki[c]

[a]*Chengdu University of Information Technology, Chengdu*
[b] *China University of Mining and Technology, Xuzhou*
[c] *Gdansk University of Technology, Gdansk*

Abstract

Time series classification (TSC) is the problem of categorizing time series data by using machine learning techniques. Its applications vary from cybersecurity and health care to remote sensing and human activity recognition. In this paper, we propose a novel multi-process collaborative architecture for TSC. The propositioned method amalgamates multi-head convolutional neural networks and capsule mechanism. In addition to the discovery of the temporal relationship within time series data, our approach derives better feature extraction with different scaled capsule routings and enhances representation learning. Unlike the original CapsNet, our proposed approach does not need to reconstruct to increase the accuracy of the model. We examine our proposed method through a set of experiments running on the domain-agnostic TSC benchmark datasets from the UCR Time Series Archive. The results show that, compared to a number of recently developed and currently used algorithms, we achieve 36 best accuracies out of 128 datasets. The accuracy analysis of the proposed approach demonstrates its significance in TSC by offering very high classification confidence with the potential of making inroads into plentiful future applications.

Keywords: Time series classification, capsule networks, data mining, signal processing

1. Introduction

Tim series classification (TSC) is the problem of categorizing time series data by using machine learning methods [1]. Time series data are sequences of time-ordered values measuring certain processes [2]. In recent years, there has been an explosion in not only volume but also velocity and variety of time series data related to real-world applications ranging from cybersecurity [3], network optimization [4] and health care [5][6], to energy efficiency management [7] and human activity recognition [8]. With the significant increase in time

series data, TSC has become one of the most important and challenging problems in data science [9][10]. TSC differs greatly from traditional classification problems because the data values are ordered [11]. In fact, any classification problem, considering some notion of ordering inside its data, can be regarded as a TSC problem [12][13]. Researchers have proposed a great number of algorithms and methods to tackle this problem [11],

* Haoxi Zhang
Email addresses: xiao1994zw@163.com (Zhiwen Xiao), jhsu99@163.com (Xin Xu), haoxi@cuit.edu.cn(Haoxi Zhang), edward.szczerbicki@zie.pg.gda.pl (Edward Szczerbicki)

which can be generally divided into two categories: traditional methods and deep learning-based methods.

One of the most popular traditional TSC approaches is the use of the nearest neighbor (NN) classifier coupled with the Dynamic Time Warping (DTW) distance function [12]. In [14], researchers introduced an approach, named Elastic Ensemble (EE), which combines ensembles of the individual NN classifiers with different distance measures, which outperforms the individual classifiers. Similarly, in [15] an ensemble method, the Bag-of-SFA-Symbols (BOSS), was proposed and demonstrated to be very promising for TSC. BOSS combines the frequency histograms extracted from the Symbolic Fourier Approximation (SFA) discretization with the structure-based representation of the bag-of-words model. Recently, Bagnall et al. [16] significantly improved the TSC accuracy by constructing an ensemble of different classifiers over different time series representations, called COTE. Then, by leveraging a new hierarchical structure with probabilistic voting, including additional representation transformation domains as well as two new classifiers, Lines et al. [17] further improved COTE to be known as the Hierarchical Vote Collective of Transformation-Based Ensembles (HIVE-COTE), which is currently considered the state-of-the-art algorithm for TSC on the University of California, Riverside (UCR) time series classification and clustering repository [18]. However, HIVE-COTE has a notable predicament: its huge computation complexity, which makes it less practical to tackle real-time big data mining problems. A more detailed comprehensive review of topical methods for TSC can be found in [11].

Apart from using traditional methods, there is increasing interest in extending deep learning approaches for TSC [8][12][13]. Particularly, researchers have borrowed ideas from image recognition challenges and their solutions [18] to tackle TSC problems. For example, Zheng et al. [20] proposed a deep learning framework based on Convolutional Neural Networks (CNNs) for multivariate time series classification. Moreover, the Time LeNet [21] and Multi-scale Convolutional Neural Networks (MCNN) [22] are considered among the first architectures to be validated on a domain-agnostic TSC benchmark such as the UCR archive [1][18]. In

CNNs, there are convolutional layers, each convolutional layer consists of sliding filters for processing the temporal data, which allows the network to extract non-linear features that are time-invariant and suitable for classification. By cascading multiple layers, CNNs can automatically learn a hierarchical feature representation from raw data. Therefore, several studies suggest CNNs for classifying electrocardiogram (ECG) signals [23][24]. More recently, it has been shown that deeper CNN models coupled with residual connections such as ResNet can further improve the classification accuracy [12][25].

Even though CNN-based methods achieve state-of-the-art classification performance, they primarily have two drawbacks in common: they disregard the spatial relationship in input data, and need considerable amounts of data samples to achieve good performance. Capsule Networks (CapsNets), as one of the attempts to address the limitations of CNNs such as the loss of spatial information in the pooling layers, is contradictory to the spatial relationships between the learned entities, were proposed in [26]. CapsNets learn and capture the properties of an entity present in the input, in this case, a signal in addition to its existence, in the form of capsules. Currently, CapsNets achieve state-of-the-art performance on the Modified National Institute of Standards and Technology (MNIST) database and performs considerably better than a convolutional net at recognizing highly overlapping digits.

Following and expanding this promising new research direction, we propose in this paper the TSCaps, a 3-head neural network that combines the multi-head structure with the capsule mechanism to support the challenging task of TSC. The proposed new approach utilizes primarily the multi-head CNNs that extract the sufficient features, and the Capsule-based mechanism that safeguards different scaled capsule routings and representation learning in addition to the temporal relationships within time series data. The main contributions of this paper can be summarized as follows:

- In present literature, CapsNets are primarily investigated in the domain of image classification. In this paper, we introduce a novel 3-head Capsule-based network to tackle the TSC problem. The proposed new architecture allows ample feature extraction and representation learning in addition to the temporal relationships within time series data.
- Unlike the original CapsNets, our proposed approach does not require arduous reconstruction technique to augment the accuracy of the model (see Section 4.2).
- We not only investigate the influence and the role of each head of our 3-head structure, but also explore the overall performance of different combinations of heads. Such approach ensures that each head of our structure is contributing to our model, and, even more importantly, it provides insights into why and when the multi-head structure can be beneficial.

2. Introduction

2.1. Multi-head Convolutional Neural Network

Convolutional neural networks (CNNs) are designed to perform feature extraction and mapping of data that

are presented in the form of manifold arrays [27]. As presented in the previous section, CNNs hold immense promise to recognize patterns in time series comprising four core elements that exploit the essential attributes of natural signals, namely, local connections, shared weights, pooling mechanism, and multi-layer network structure. All of these elements establish well-defined means of feature extraction and mapping required for TSC. Computation units attain the local basic features of time series data in the lower network layers, while they learn higher-level representation and patterns of the data in the higher network layers. Moreover, in comparison with traditional feed-forward networks, such as fully connected neural networks, CNNs perform with much fewer connections, and they are easier to train [18].

The standard CNNs can be considered a one-head architecture. The multi-head CNNs [28] simply multiply this representation learning ability. With multiple heads, the CNNs can have different filter banks and different processing layers in each head. For instance, our proposed method utilizes a 3-head CNNs in the first layer, which has a number of 9, 7 and 5 filters at each head respectively. If necessary, we may even choose whether to have pooling or dropout layers for a certain head. By using multiple heads, the CNNs is empowered with the unique ability to combine various feature learning processes for tackling input series, which enriches the features extracted and thus enhances the final representation learning results.

2.2. Capsules

The idea of a capsule is first introduced by Hinton and his colleagues [29] as an alternative to CNNs. A capsule is composed of a group of neurons, which deals with vectors instead of CNNs' scalar values. This exceptional characteristic enables a capsule to learn the features of an image in addition to its deformations and viewing conditions [30]. After being processed based on the type of the capsule employed, the features produced by a CNN are accepted as the input to a capsule. The output of the Capsule is made up of a set of activity vector values commonly called instantiation parameters. The capsule's activity vector carries various properties of a particular entity such as an object or an object part [26]. More specifically, the length of the activity vector represents the probability of the entity's existence, while the orientation of the activity vector holds the instantiation parameters of the entity. The instantiation parameters are used to represent equivariance of the capsule indicating its ability to recognize pose, deformation, velocity, texture, etc. The equivariance makes sure that the capsule takes into account spatial relationships of entities.

By using the algorithm called "routing by agreement" between different layers [29], active capsules at the lower level make predictions for the instantiation parameters of their higher-level capsules. When multiple predictions agree, a higher level capsule becomes active. This allows neural activities of capsules to vary according to varying viewpoints, instead of eliminating, which gives capsules the advantage over normalization methods. Because of that, they can handle multiple different affine transformations of different objects or object parts simultaneously. Furthermore, this unique property also makes capsules very effective

for tackling segmentation, which is another challenging problem in computer vision [26].

Motivated by the above promising findings, in this paper we propose to apply the multi-head structure with the capsule mechanism to support the task of TSC. Instead of simply assembling capsule modules, the capsule block in our proposed architecture is completely re-designed in order to suit time series problems and achieve state-of-the-art performance.

3. Methodology

In this section, we present the methodology of our proposed Capsule-based neural architecture for TSC, including the overview of the proposed approach, followed by the mathematical formulation of multi-head convolution, capsule activation, and routing. Then, we present our dynamic routing algorithm. At the end of this section, the classifier and the training procedure are introduced.

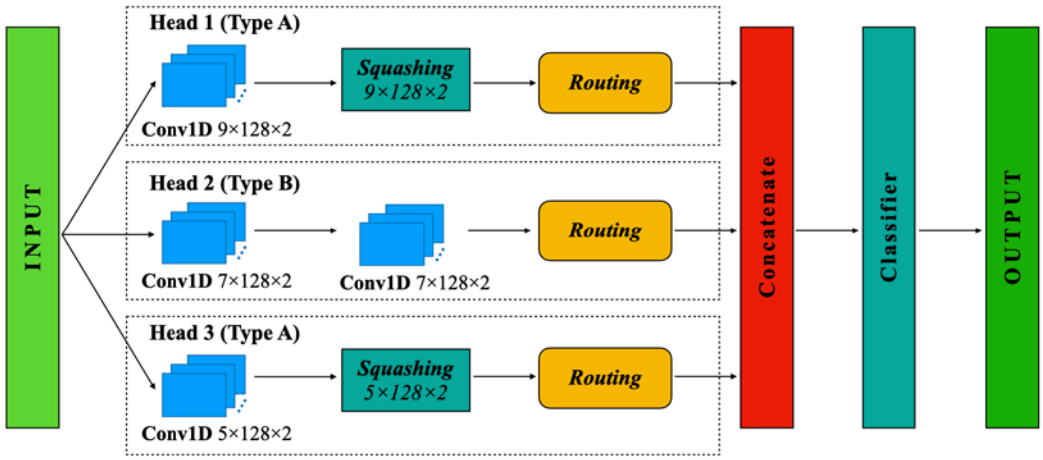

Fig. 1. Schematic diagram of the proposed TSCaps approach.

3.1. Overview

The architecture of the proposed method for TSC is shown in Fig. 1. A three-head structure with two types of heads is chosen for the TSCaps in order to make the method more robust and effective (see the comparison and ablation study in section 4.2). The most general and sketchy overview of the approach is as follows. Given an input series, we first use a 3-head CNN to extract its features. Then, these features are sent to capsules for further representation learning in addition to the temporal relationships among small segments within the time series. Finally, the Classifier produces the output of the network, which is the category

prediction of the input.

More specifically, the TSCaps is designed to tackle feature extraction and learning from the input data. The input data $D = \{(X_1, y_1),\ (X_2, y_2), \ldots, (X_n, y_n)\}$ is a dataset containing a collection of pairs (X_i, y_i) where X_i is the time series $X_i = [x_i^1,\ x_i^2, \ldots, x_i^m]$ consisting of m ordered values with y_i as its class label. By using convolution operation, the features of the input data can be extracted; these features are then vectorized through squashing and fed into the routing process. At the end, the Classifier utilizes the softmax function to carry out the mapping from the space of possible inputs to a probability distribution over the class labels, and produces the output of the method, which is the predicted label y_i' by given the input X_i.

3.2. The Three Heads

The three heads are the core of the proposed method, which learns the representation of data in addition to the temporal relationships among small segments within the time series. The three heads can be further divided into two types. Type A is under a normal CapsNet structure that consists of convolution module, squashing module, and routing module. In Type B there is no squashing module; instead, we use another convolution module to learn higher level representation of the input data (Fig. 1.). The key components of the three heads are the Convolution module, the Squashing module, and the Routing module.

3.2.1. The Convolution module

Aiming to extract various features, the convolution module is designed to deal with the initial input data. In the convolution module, the input data is convolved with a set of convolutional filter banks (to be learned in the training process). The output of the convolutional operators enhanced by a bias (to be learned) is put through the activation function to form the feature map for the next layer/module. Formally, given the input data X, the i^{th} feature map of h^{th} head of the multi-head CNN is also a matrix, denoted as v_i^h , and it is given by:

$$v_i^h = f_{leaky\_relu}\left(f_{BN}\left(f_{conv}^h(X)\right)\right), \quad \forall h \in \{1,2,3\} \tag{1}$$

where f_{leaky_relu} is the activation function that can retain some useful negative values, defined as:

$$f_{leaky\_relu}(x) = \begin{cases} \alpha * x, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{2}$$

where α is a coefficient for retaining negative values (we set $\alpha = 0.1$ in our experiments), f_{BN} is the batch normalization function that accelerates the training and enhances the classification accuracy. Moreover, f_{conv}^{h} is the convolution function of the h^{th} head in the multi-head convolution layer, as presented in (3):

$$f_{conv}^{h}(X) = b_i^h + \sum_k \sum_{p=0}^{n^h} W_{i\,k}^{p,h} X \qquad \forall h \in \{1,2,3\} \tag{3}$$

where b_i^h is the bias for this particular feature map, k is the index of the feature maps at the convolution module, n^h is the size of the filter bank of the h^{th} head, and $W_{i\,k}^{p,h}$ is the value at the position p of the convolution filter bank connected to the current feature map. After the 3-head convolution process is performed, a number of various features are acquired and sent to the next module.

3.2.2. The Squashing module

Following the convolution module, the Squashing module is developed to receive the feature maps extracted from the convolution module, and to transfer them into vectors that capsules employed. It utilizes a non-linear "squashing" function to carry out this transfer. Formally, the output of the h^{th} head convolution module is denoted as v^h, and the transfer is made by

$$s^h = f_{squash}^{h}\left(f_{reshape}^{h}(v^h) \right) \qquad \forall h \in \{1,2,3\} \tag{4}$$

where s^h is the output of this module for the h^{th} head, $f_{reshape}^{h}$ is the reshaping function that ensures the data matches the required shape by the Capsule, and f_{squash}^{h} is defined as:

$$f_{squash}(x) = \frac{||x||^2}{1+||x||^2} \frac{x}{||x||} . \tag{5}$$

3.2.3. The Routing module

It is the module that learns the features of data and takes into account the temporal relationships by using

the “routing by agreement” algorithm (Algorithm 1). In this module, the total input to a capsule is a weighted sum over all “prediction vectors”. For instance, the total input s_j^h to capsule j is computed as:

$$S_j^h = \sum_i k_{ij}^h \hat{c}_{j|i}^h , \quad \hat{c}_{j|i}^h = W_{ij}^h c_i^h \quad \forall h \in \{1,2,3\} \tag{6}$$

where $\hat{c}_{j|i}^h$ is the “prediction vectors” from the capsules in the previous module and is calculated by multiplying c_i^h , the output of capsule i in the previous module, by the weight matrix W_{ij}^h, and the k_{ij}^h are coupling coefficients that are learned via the iterative routing process.

The coupling coefficients between all capsules in this module and capsule i in the previous module are given by a “routing softmax” defined as:

$$k_{ij}^h = \frac{e^{b_{ij}^h}}{\sum_l e^{b_{il}^h}} \qquad \forall h \in \{1,2,3\} \tag{7}$$

where b_{ij}^h are initial logits representing the log prior probabilities that capsule i should be coupled to capsule j, and the l represents the other capsules link with capsule i but capsule j in this module.

The initial logits can be learned through the training, so that the coupling coefficients can be refined iteratively by measuring the agreement between the current output c_j^h and the prediction $\hat{c}_{j|i}^h$ produced by capsule i from the previous module. The output of a capsule is calculated via “squashing” its total input, as defined in (5). The agreement is just the product $d_{ij}^h = c_j^h \cdot \hat{c}_{j|i}^h$, and it is added to the initial logits b_{ij}^h before computing new values for the coupling coefficients. The vector output of the Routing module is carried out through the “routing by agreement” algorithm that is presented in Algorithm 1.

Finally, we get three sets of output as each head produces one output. These outputs are vectors in different scales and carry various features. In order to take advantage of the variety of representations, we utilize a concatenation process to put them into one piece while keeping critical features learned which are defined as:

$$V_j = f_{coe\_concat}\left(\left[\zeta c_j^1, \eta c_j^2, \mu c_j^3 \right] \right) \tag{8}$$

where ζ, η, μ are coefficients of the outputs from the three heads respectively. The V_j is then sent to the

Alogrithm 1: Routing by agreement algorithm

1 **procedure** $Routing(s^h, r^h) \quad \forall h \in \{1,2,3\}$
2 initial weight matrix W_{ij}^h
3 get $\hat{c}_{j|i}^h = \sum W_{ij}^h s^h$
4 $b_{ij}^h = 0$
5 **for** r^h iterations **do**
6 get k_{ij}^h by using (7)
7 get $\hat{c}_{j|i}^h$ by using (6)
8 get S_j^h by using (6)
9 get $c_j^h = f_{squash}(S_j^h)$ computes (5)

classifier for final predictions. We set $\zeta = 1,\ \eta = 1,\ \mu = 1$ in our experiments.

3.3. Classifier

The Classifier carries out the mapping from its inputs to a probability distribution over the class labels by giving the predicted class labels y′ as the output of our proposed method. Particularly, in a capsule, the instantiation vector's length represents the probability of the entity's existence, which is defined as:

$$p(y_j' = L_j \mid V_j; \theta) = \frac{e^{\theta_j ||V_j||}}{\sum_{k=1}^{N} e^{\theta_k ||V_j||}} \tag{9}$$

where θ represents the parameters of the activity vector V_j, and N is the number of classes that are denoted by $L_j = (1,2,\dots,N)$. Therefore, it is anticipated that the correct capsule in TSCaps has a long instantiation vector. To predict multiple classes, the separate margin loss is used for each capsule linked to a particular class k:

$$loss = \frac{1}{n}\sum_{j=1}^{n} (y_j \max(0, m^+ - ||V_j||) + \lambda (1 - y_j) \max(0, ||V_j|| - m^-)) \tag{10}$$

where y_j is the truth label of j^{th} class, and λ is a coefficient for the margin loss. We set $\lambda = 0.5$, $m^+ = 0.9$ and $m^- = 0.1$ in our training. The whole training process is summarized in Algorithm 2.

Algorithm 2: TSCaps Optimization

Input: labeled time series dataset: D = {X, Y}
Output: predicted label y_j' of the input
// Initialization
Initialize the parameters θ
Normalize the dataset
Divide the dataset into certain sets:
    training dataset: $D_{train} = \{X^{train}, Y^{train}\}$
    validation dataset: $D_{val} = \{X^{val}, Y^{val}\}$
    testing dataset: $D_{test} = \{X^{test}, Y^{test}\}$
// Training on training and validation datasets
for epoch = 1, M **do**
  for n = 1, N **do**
    get the input data $X_j \in D_{train}$
    feedforward the X_j and get the activity vector V_j
    // prediction
    get the predicted label y_j' by computing (9)
    get the loss by computing (10)
    perform a gradient decent step on (loss | θ)
  end for
  if (epoch % 2 == 0) **then**
    validate the model using D_{val}
    save θ
  end if
end for
// Testing
Use the trained network to predict the labels of D_{test}

4. Experiments and Results

In this section, we first introduce the experimental setup and dataset description, then explore relations of each head of multi-head capsules through ablation study, and finally analyze and compare our algorithms with others.

4.1. Dataset Description and Experiment Settings

The UCR 2018 archive is one of the most popular time series repositories with 128 datasets of different lengths in various application domains. In order to ensure verified fairness of the proposed approach for time-series data with various lengths, the UCR 2018 archive is divided into 4 categories (i.e. ‘short’, ‘medium’, ‘long’, and ‘vary’) according to the length of each dataset. To be specific, the 128 datasets consist of 41

'short' , 32 'medium', 44 'long', and 11 'vary' datasets (more details are shown in Table 1 and Table 2), and 'Total' represents the whole UCR 2018 datasets archive. In our partition ‘short’ refers the length of the dataset that is below 200, ‘medium’ ranges from 200 to 500, ‘long’ is over 500, and ‘vary’ is for dataset with indefinite length. All experiments are run on a desktop with a Nvidia GTX 1080Ti GPU with 11 GB plus another Nvidia GTX 1070Ti GPU with 8GB, and an AMD R5 1400 CPU with 16G RAM under the Ubuntu 18.04 OS.

Table 1: The details of ‘short’ and ‘medium’ datasets.

Scale	Dataset	Train	Test	Class	Length	Type
Short	SmoothSubspace	150	150	3	15	Simulated
	ItalyPowerDemand	67	1029	2	24	Sensor
	Chinatown	20	345	2	24	Traffic
	MelbournePedestrian	1194	2439	10	24	Traffic
	Crop	7200	16800	24	46	Image
	SyntheticControl	300	300	6	60	Simulated
	SonyAIBORobotSur.2	27	953	2	65	Sensor
	SonyAIBORobotSur.1	20	601	2	70	Sensor
	DistalPhalanxO.A.G	400	139	3	80	Image
	DistalPhalanxO.C.	600	276	2	80	Image
	DistalPhalanxTW	400	139	6	80	Image
	MiddlePhalanxO.A.G.	400	154	3	80	Image
	MiddlePhalanxO.C.	600	291	2	80	Image
	MiddlePhalanxTW	399	154	6	80	Image
	PhalangesO.C.	1800	858	2	80	Image
	ProximalPhalanxO.A.G.	400	205	3	80	Image
	ProximalPhalanxO.C.	600	291	2	80	Image
	ProximalPhalanxTW	400	205	6	80	Image
	TwoLeadECG	23	1139	2	82	ECG
	MoteStrain	20	1252	2	84	Sensor
	ECG200	100	100	2	96	ECG
	ElectricDevices	8926	7711	7	96	Device
	MedicalImages	381	760	10	99	Image
	CBF	30	900	3	128	Simulated
	SwedishLeaf	500	625	15	128	Image
	TwoPatterns	1000	4000	4	128	Simulated
	BME	30	150	3	128	Simulated
	FaceAll	560	1690	14	131	Image
	FacesUCR	200	2050	14	131	Image
	ECGFiveDays	23	861	2	136	ECG
	ECG5000	500	4500	5	140	ECG
	Plane	105	105	7	144	Sensor
	PowerCons	180	180	2	144	Power
	GunPoint	50	150	2	150	Motion
	GunPointAgeSpan	135	316	2	150	Motion
	GunPointMaleV.F.	135	316	2	150	Motion
	GunPointOldV.Y.	136	315	2	150	Motion
	UMD	36	144	3	150	Simulated
	Wafer	1000	6164	2	152	Sensor
	ChlorineCon.	467	3840	3	166	Sensor
	Adiac	390	391	37	176	Image
Medium	Fungi	18	186	18	201	HRM
	Wine	57	54	2	234	Spectro
	Strawberry	613	370	2	235	Spectro
	ArrowHead	36	175	3	251	Image
	InsectWingbeatS.	220	1980	11	256	Sensor
	FiftyWords	450	455	50	270	Image
	WordSynonyms	267	638	25	270	Image
	Trace	100	100	4	275	Sensor
	ToeSegmentation1	40	228	2	277	Motion
	Coffee	28	28	2	286	Spectro
	DodgerLoopDay	78	80	7	288	Sensor
	DodgerLoopGame	20	138	2	288	Sensor
	DodgerLoopWeekend	20	138	2	288	Sensor
	CricketX	390	390	12	300	Motion
	CricketY	390	390	12	300	Motion
	CricketZ	390	390	12	300	Motion
	FreezerRegularTrain	150	2850	2	301	Sensor
	FreezerSmallTrain	28	2850	2	301	Sensor
	UWaveGestureL.X	896	3582	8	315	Motion
	UWaveGestureL.Y	896	3582	8	315	Motion
	UWaveGestureL.Z	896	3582	8	315	Motion
	Lightning7	70	73	7	319	Sensor
	ToeSegmentation2	36	130	2	343	Motion
	DiatomSizeRe.	16	306	4	345	Image
	FaceFour	24	88	4	350	Image
	Symbols	25	995	6	398	Image
	Yoga	300	3000	2	426	Image
	OSULeaf	200	242	6	427	Image
	Ham	109	105	2	431	Spectro
	Meat	60	60	3	448	Spectro
	Fish	175	175	7	463	Image
	Beef	30	30	5	470	Spectro

Table 2: The details of ‘long’ and ‘vary’ datasets.

Scale	Dataset	Train	Test	Class	Length	Type
Long	FordA	3601	1320	2	500	Sensor
	FordB	3636	810	2	500	Sensor
	ShapeletSim	20	180	2	500	Simulated
	BeetleFly	20	20	2	512	Image
	BirdChicken	20	20	2	512	Image
	Earthquakes	322	139	2	512	Sensor
	Herring	64	64	2	512	Image
	ShapesAll	600	600	60	512	Image
	OliveOil	30	30	4	570	Spectro
	Car	60	60	4	577	Sensor
	InsectEPGRegularT.	62	249	3	601	EPG
	InsectEPGSmallT.	17	249	3	601	EPG
	Lightning2	60	61	2	637	Sensor
	Computers	250	250	2	720	Device
	LargeKitchenApp.	375	375	3	720	Device
	RefrigerationDevices	375	375	3	720	Device
	ScreenType	375	375	3	720	Device
	SmallKitchenApp.	375	375	3	720	Device
	NonInvasiveFetalECG.	1800	1965	42	750	ECG
	NonInvasiveFetalECG.	1800	1965	42	750	ECG
	Worms	181	77	5	900	Motion
	WormsTwoClass	181	77	2	900	Motion
	UWaveGestureL.All	896	3582	8	945	Motion
	Mallat	55	2345	8	1024	Simulated
	Phoneme	214	1896	39	1024	Sensor
	StarLightCurves	1000	8236	3	1024	Sensor
	MixedShapesRegularT.	500	2425	5	1024	Image
	MixedShapesSmallT.	100	2425	5	1024	Image
	Haptics	155	308	5	1092	Motion
	EOGHorizontalSignal	362	362	12	1250	EOG
	EOGVerticalSignal	362	362	12	1250	EOG
	ACSF1	100	100	10	1460	Device
	SemgHandG.Ch2	300	600	2	1500	Spectrum
	SemgHandM.Ch2	450	450	6	1500	Spectrum
	SemgHandS.Ch2	450	450	5	1500	Spectrum
	CinCECGTorso	40	1380	4	1639	Sensor
	EthanolLevel	504	500	4	1751	Spectro
	InlineSkate	100	550	7	1882	Motion
	HouseTwenty	40	119	2	2000	Device
	PigAirwayPre.	104	208	52	2000	Hemodynamics
	PigArtPre.	104	208	52	2000	Hemodynamics
	PigCVP	104	208	52	2000	Hemodynamics
	HandOutlines	1000	370	2	2709	Image
	Rock	20	50	4	2844	Spectrum
Vary	AllGestureWiimoteX	300	700	10	Vary	Sensor
	AllGestureWiimoteY	300	700	10	Vary	Sensor
	AllGestureWiimoteZ	300	700	10	Vary	Sensor
	GestureMidAirD1	208	130	26	Vary	Trajectory
	GestureMidAirD2	208	130	26	Vary	Trajectory
	GestureMidAirD3	208	130	26	Vary	Trajectory
	GesturePebbleZ1	132	172	6	Vary	Sensor
	GesturePebbleZ2	146	158	6	Vary	Sensor
	PickupGestureW.Z	50	50	10	Vary	Sensor
	PLAID	537	537	11	Vary	Device
	ShakeGestureW.Z	50	50	10	Vary	Sensor

Table 3: Results on various structure in ablation study.

Scale	Dataset	*Head 1*	*Head 2*	*Head 3*	*Head 1&2*	*Head 1&3*	*Head 2&3*	*Ours without Caps*	*Ours with Recon*	*Ours*
Short	ECG200	0.9100	0.9000	0.9000	0.9200	0.9100	0.92	0.63	**0.93**	**0.93**
	ECG 5000	0.944222	0.938444	0.590222	0.946444	0.945222	0.944444	0.589333	**0.948444**	**0.948444**
	ChlorineCon.	0.849479	0.816146	0.816416	0.863281	0.850260	0.849479	0.409375	**0.87474**	**0.87474**
MeanACC		0.901234	0.884863	0.768879	0.909908	0.901827	0.904641	0.542903	**0.917728**	**0.917728**
Medium	Strawberry	0.97297	0.959459	0.945946	0.981081	0.981081	0.978378	0.810811	**0.986486**	**0.986486**
	ArrowHead	0.845714	0.834286	0.828571	0.857143	0.851429	0.845714	0.668571	**0.868571**	**0.868571**
	DodgerLoopW.	0.55	0.5375	0.5	0.65	0.6375	0.6125	0.2	0.7	**0.7625**
MeanACC		0.789561	0.777082	0.758172	0.829408	0.823337	0.812197	0.559794	0.851686	**0.872519**
Long	OliveOil	0.9	0.866667	0.866667	0.933333	0.933333	0.9	0.7	**0.966667**	**0.966667**
	SemgH.G	0.873333	0.85	0.831667	0.906667	0.896667	0.893333	0.313333	**0.916667**	**0.916667**
	Rock	0.76	0.76	0.72	0.84	0.84	0.8	0.34	**0.86**	**0.86**
MeanACC		0.844444	0.825556	0.806111	0.893333	0.89	0.864444	0.451111	**0.914445**	**0.914445**
Vary	AllGestruerW.X	0.701492	0.697142	0.694286	0.717143	0.714286	0.714286	0.285714	**0.742857**	**0.742857**
	GestureMidAirD1	0.669231	0.653846	0.646154	0.715385	0.7	0.692308	0.430769	**0.723077**	**0.723077**
	PickupGestureW.Z	0.78	0.76	0.72	**0.8**	**0.8**	0.76	0.22	**0.8**	**0.8**
MeanACC		0.716908	0.703663	0.686813	0.744176	0.738095	0.722198	0.312161	**0.755311**	**0.755311**

* MeanACC – mean accuracy

4.2. Ablation Study

To investigate effects and performance of different structures and types for the heads of our proposed approach, we employ an ablation study in our experiments on 12 datasets, including 3 ‘short’ datasets, 3 ‘medium’ datasets, 3 ’long’ datasets, and 3 ‘vary’ datasets (see Table 3).

First, we compare our proposed network structure (*Ours*) with pure multi-head CNNs (*Ours without Caps*), which verifies contributions of capsules. Then we add to the comparison analysis the original CapsNets structure (*Ours with Recon*) that utilizes a reconstruction module to enhance its performance. Table 3 shows that compared with *Ours without Caps*, *Ours* achieves noticeable better performance on every dataset demonstrating the effects of the capsule mechanism. For instance, the test accuracies on ECG200 dataset of these two structures are 0.63 and 0.93, respectively. Moreover, when employing the reconstruction module (*Ours with Recon*), the network doesn’t perform better: it even underperforms *Ours* by 0.06 on the DodgerLoopW dataset.

Next, if focusing on a single head network structure, it can be seen (Table 3) that the *Head 1* outperforms other two single heads on each dataset. For example, the accuracies of the three single heads on ECG500

dataset are 0.944222, 0.9384444, and 0.590222 respectively. The *Head 3* performs worst among these three single heads because the performance of a single head heavily depends on the scale of its size. The larger of the head's scale the more shapelets and features can be extracted from the given input data.

Additionally, we find that the performance of multiple-head structure is always better than the single head structure. Specifically, the three-head structure beats the two-head ones, while the two-head network beats the single-head approaches. We also find that combining *Head 1* and *Head 2* achieves the best performance, which indicates that *Head 2* (with two Convolution Modules, i.e. no Squashing Module) is effective and beneficial for the proposed approach.

Through comparisons between various network structures, we find that the multi-head configurations can take advantages of the variety of their head to extract diverse features from input data, resulting in a more robust and accurate model.

Finally, the computational complexity is compared between *Ours* and *Ours with Recon*, where *Ours with Recon* is composed of three layers, i.e. a fully-connected layer with 128 channels, a fully-connected layer with 256 channels, and a fully-connected layer with the number of channels equal to the length of a given dataset when the two approaches achieve similar performance. The latter approach is certain to cost the larger amount of computing resources due to its extra reconstruction module. For instance, the parameters of *Ours* and *Ours with Recon* are 11.8684M and 12.2932M on SemgH.G dataset, respectively (see Table 3). Consequently, the computational time costs of *Ours* are around 2 times less than *Ours with Recon* on different datasets, e.g. the test time cost on SemgH.G dataset are 14.3279s and 28.538s on CPU, respectively. At the same time, their accuracies on each dataset are almost the same. Therefore we conclude that our proposed approach makes full use of the variety of its heads to extract diverse features, and achieves the best accuracy without the reconstruction module, which largely reduces the complexity of our approach while still ensure its high performance.

Table 4: Statistical results obtained by various algorithms.

Scale	Ranks	Existing SOTA [25]	TS-CHIEF	Vanilla:ResNet Transformer	ResNet-Trans1	ResNet-Trans2	ResNet-Trans3	ResNet 50 SC	ResNet 152 SC	Inception--Time	ROCKET	Ours
Total	Win	**19**	2	6	3	7	12	3	5	10	10	11
	Tie	13	8	**31**	23	21	22	6	6	11	12	25
	Lose	53	78	91	102	100	94	119	117	107	106	92
	Best	32	10	**37**	26	28	34	9	11	21	22	36
	AVG_rank	6.222656	7.730496	**4.238281**	5.261719	5.8125	5.723656	7.312500	7.359375	5.785156	5.613281	4.941406
Short	Best	13	4	**18**	15	15	15	4	4	7	7	17
	AVG_rank	5.902439	7.780488	**4.182927**	4.878049	5.731707	5.317073	7.060975	7.146341	6.634146	6.341463	5.024390
Medium	Best	8	4	7	6	4	**11**	5	2	7	7	10
	AVG_rank	5.296875	6.390625	5.093750	**4.828125**	6.781250	5.937500	7.406350	8.078125	5.937500	4.921875	5.328125
Long	Best	**11**	2	9	4	6	6	0	3	3	6	8
	AVG_rank	6.125000	7.965909	**3.909091**	5.886364	5.522727	6.102273	7.522727	7.284091	5.545455	5.534091	4.602273
Vary	Best	0	0	3	1	3	2	0	2	**4**	2	1
	AVG_rank	10.500000	10.500000	3.272727	5.454545	4.454545	5.090909	3.136364	7.136364	**3.136336**	5.227273	4.863636

4.3. Experimental Analysis

To evaluate the performance of our proposed approach, we select for the comparison process seven best existing approaches that claim the state-of-the-art results as presented in the highly cited paper [11] and the most recent arXiv preprint 2020 [25] (see Table 4). Following the standard approach most researchers take, we use 'win', 'tie', 'lose' and the average ranking (AVG_rank) to rank algorithms taking part in the experimental evaluation process (please refer to the APPENDIX for detailed scores of each algorithm on each dataset). The 'win', 'tie' and 'lose' index represents the number of datasets that an approach performs better than, equivalent to, or worse than others, respectively. The 'best' cases are the sum of 'win' and 'tie' scores. The average ranking scores are defined according to the average Geo-ranking approach, measuring the average difference between the accuracies of a model and the best accuracies among all models. We calculate the mean accuracy by averaging the measures over 30 runs on each test set.

Table 4 shows the statistical results achieved by nominated algorithms on selected 44 datasets in the UCR 2018 archive. For each dataset, the existing SOTA represents the best algorithm on that dataset [25], including DTW [12], BOSS [15], COTE [16], and EE [14]. It should be noted, that both SOTA and TS-CHIEF algorithms [32] don't consider the last 43 of 128 datasets (detailes can be found in [25] [32]).

As it can be observed in Table 4, Vanilla:ResNet-Transformer attains the first position in the 'best' and AVG_rank evaluations. Our is a close second, only one score less in the 'best' cases, i.e. 36 'best' scores. To be specific, our algorithm wins in 11 cases and performs no worse than any other algorithm in 25 cases.

Similarly, our proposed approach also follows the latter in the AVG_rank metric. Also, ResNet-Tran1 and ResNet-Trans3 take the third place in the AVG_rank and 'best' metrics, respectively. Table 4 additionally demonstrates that it is hard for TS-CHIEF [32] to extract efficient features from a variety of datasets despite its combination of heterogeneous and integrated embedding forest, and it fails in the competition with scoring only 2 'win' values.

To further investigate the performance of our proposed approach, we compare it with other algorithms using the scores of 'best' and AVG_rank on 'short', 'medium', 'long', and 'vary' datasets. Table 4 illustrates that Vanilla:ResNet- Transformer is still the best among algorithms on 'short' datasets in terms of 'best' and AVG_rank values. Our proposed approach gains the second and third position in the 'best' and AVG_rank evaluations, respectively. ResNet-Trans1 takes the second and third position in the 'best' and AVG_rank evaluations, respectively. TS-CHIEF is undoubtedly the worst performer.

When focusing on 'medium' datasets, one can find that ResNet-Trans3 achieves the best performance in terms of the highest 'best' scores of 11. Our procedure follows closely the latter and obtains 10 'best' scores. On the other hand, in terms of AVG_rank metric, ResNet-Trans1 achieves the lowest AVG_rank scores of 4.828125. ROCKET [31] makes use of a linear classifiers using random convolutional kernels to attain the second position. However, compared with the performance on 'short' datasets, Vanilla:ResNet-Transformer and Ours both perform poorly and they slipped in relative rankings. For example, Vanilla:ResNet-Transformer moves from the first to the third position. The reason may be behind their structure that is less sensitive to 'medium' length signal information. In addition, ResNet 152 SC [32] that relies on complex residual structure fails in the competition.

Considering 'long' datasets, Vanilla:ResNet-Transformer and our proposed approach have significantly improved in the AVG_rank performance metric compared with the performance on 'medium' datasets. Their positions are found as the first and second, namely 3.909091 and 4.602273 AVG_rank scores, respectively. This is because the former takes advantage of its transformer structure to relate different position of 'long' sequences, while the latter (Ours) fusions different scaled features through the multi-head capsule structure. ResNet-Tran2 and ROCKET are behind in terms of the AVG_rank performance evaluation.

Furthermore, when paying more attention to 'vary' datasets compared with the performance on 'long' datasets, Vanilla:ResNet-Transformer and Ours are both down in terms of the AVG_rank values. On the contrary, InceptionTime makes use of the inception structure to mine sufficient features from 'vary' datasets, ensuring the best performance in the 'best' and AVG_rank cases.

Lastly, we also visualize the methods' comparison employing the critical difference diagram proposed by

Demšar [33]. The diagram shows a thick horizontal line when a group of classifiers are not-significantly different in terms of accuracy, and a given classifier is better the closer to the right hand site of the thick line it is located (has smaller scaler). Fig. 3 illustrates the comparison results.

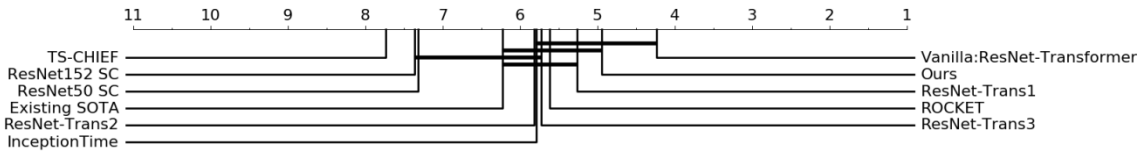

Fig. 3. Critical difference diagram showing pairwise statistical difference comparison of state-of-the-art classifiers on 128 UCR datasets.

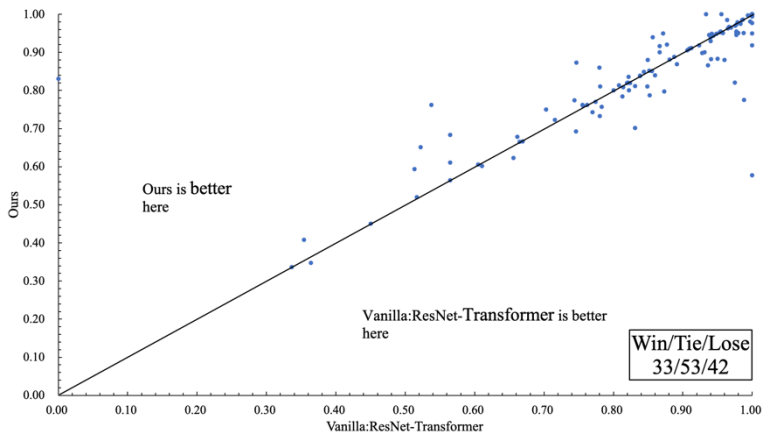

Fig. 3. Accuracy plot showing the performance difference between Vanilla:ResNet-Transformer and Ours.

Finally, to further visualize the difference between Vanilla:ResNet-Transformer and Ours, Fig. 3 depicts the accuracy plot of Ours against Vanilla:ResNet- Transformer for each of the whole 128 UCR datasets. The results show that Ours gains 'win'/'tie'/ 'loss' in 33/55/42 cases respectively, with p-value well over 0.5 (about 0.9451). Meanwhile, the mean accuracy (MeanACC) of Ours is 0.0013 higher than that of Vanilla:ResNet-Transformer. This indicates that there is no significant performance difference between them. It can be stated, that the performance of Our proposed approach is the same as Vanilla:ResNet- Transformer, both of which have huge potential to deal with a variety of datasets. Additionaly, Ours compared with InceptionTime (see

Fig. 4) obtains 'win'/'tie'/'loss' in 62/11/55 cases on whole UCR datasets, Fig. 5 depicts that Ours also achieves 'win'/'tie'/'loss' of 68/10/50 cases compared with ROCKET.

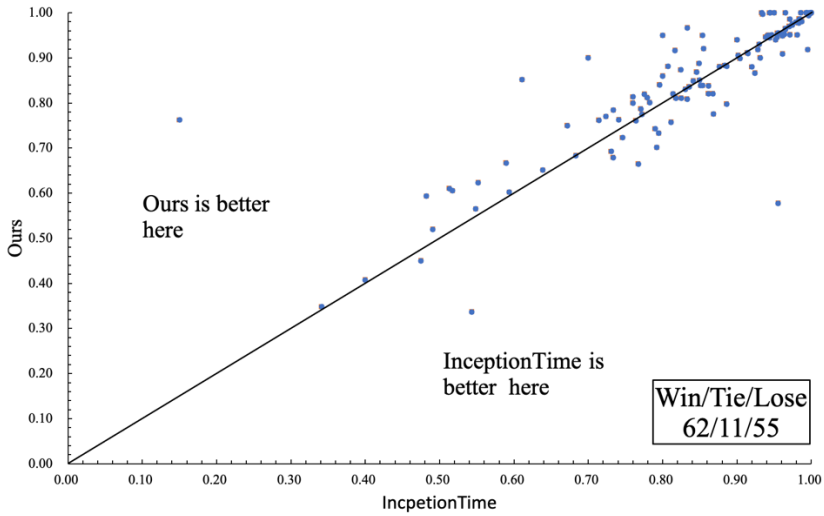

Fig. 4. Accuracy plot showing the performance difference between InceptionTime and Ours.

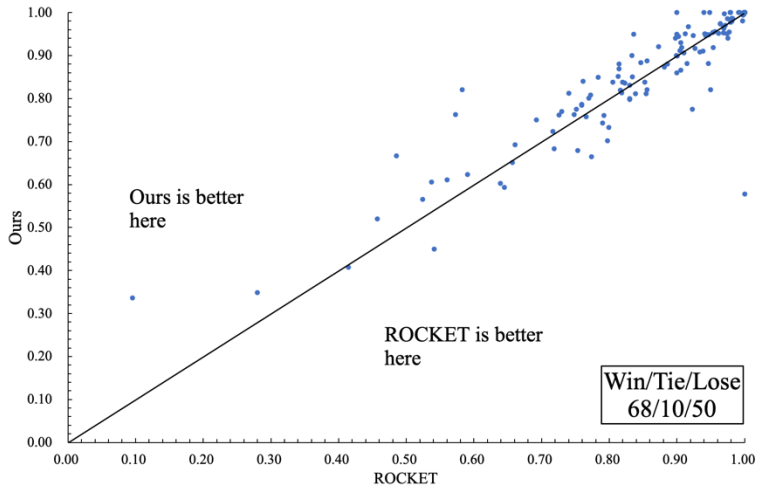

Fig. 5. Accuracy plot showing the performance difference between ROCKET and Ours.

5. Conclusions

In this paper, we propose the Capsule-based neural structure for TSC. The proposed method takes advantages of multi-head convolutional neural networks and capsule mechanism integration, to achieve better feature extraction and different scaled capsule routings and representation learning in addition to the temporal relationships discovery within time-series data. As our proposed architecture is able to explore sufficient shapelets hidden in the data, we do not need to employ the reconstruction technique to enhance the accuracies of the model. Therefore, unlike the original CapNets, our approach is more computing friendly. We compare our proposed method with the current state-of-the-art approaches by using the whole URC dataset. The comparison results show that our proposed procedure achieves very reasonable performance by wining 11 classification tasks and drawing in 25, and that it provides the highest average accuracy over all 128 tested datasets. The accuracy analysis of the proposed approach demonstrates its significance in TSC by offering very high classification confidence with the potential of making inroads into plentiful future applications.

Our future work will involve exploring ways to reduce the complexity of our proposed approach and make it more practical. We believe that detailed study of elaborately hand-crafted features and automatically learned features needs to be performed first. Then, we plan to distill the prior knowledge encoded in these features and introduce such knowledge into neural networks to enhance the model with long-term dependencies that are hard to learn with a limited dataset.

Acknowledgements

This work was supported by the Sichuan Science and Technology Program under Grant 2019YFH0185.

References

[1] Clark, H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P. Muller and F. Petitjean. InceptionTime: finding AlexNet for time series classification. *Data Min. Knowl. Disc.*, vol. 34, pp. 1936-1962, 2020.

[2] O. Triebe, N. Laptev and R. Rajagopal. AR-Net: a simple auto-regressive neural network for time-series. *arXiv preprint arXiv:1911.12436*, 2019.

[3] F. Wei, Z. Wan, H. He and X. Lin. Ultrafast active response strategy against malfunction attack on fault current limiter. *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2722-2733, 2020.

[4] L. Wang, H. He and Z. Zeng. Intermittent Stabilization of Fuzzy Competitive Neural Networks with Reaction Diffusions, in IEEE Transactions on Fuzzy Systems, doi: 10.1109/TFUZZ.2020.2999041.

[5] G. Forestier, F. Petitjean, P. Senin, F. Despinoy, A. Huaulmé, H. I. Fawaz, J. Weber, L. Idoumghar, P.-A. Muller and P. Jennin. Surgical motion analysis using discriminative interpretable patterns. *Artif. Intell. Med.*, vol. 91, pp. 3-11, 2018.

[6] H. I. Fawaz, G. Forestier, J. Weber, F. Petitjean, L. Idoumghar and P.-A. Muller. Automatic alignment of surgical videos using kinematic data. In *Proc. AJME 2019*, pp. 104-113, Poznan, Poland, 2019.

[7] F. Wei, Z. Wan, H. He, X. Lin and Y. Li. A Novel Scheduling Strategy for Controllable Loads With Power-Efficiency Characteristics, in IEEE Transactions on Smart Grid, vol. 11, no. 3, pp. 2151-2161, May 2020, doi: 10.1109/TSG.2019.2948370.

[8] H. Zhang, Z. Xiao, J. Wang, F. Li and E. Szczerbicki. A novel IoT-perceptive human activity recognition (HAR) approach using multi-head convolutional attention. *IEEE Internet Things J.*, vol. 2, no. 7, pp. 1072-1080, 2020.

[9] Q. Yang and X. Wu. 10 challenging problems in data mining research. *Int. J. Inf. Tech. Decis.*, vol. 5, no. 4, pp. 597-604, 2006.

[10] P. Esling and C. Agon. Time-series data mining. *ACM Comput. Surv.*, vol. 1, no. 45, pp. 1-34, 2012.

[11] A. Bagnal, J. Lines, A. Bostrom, J. Large and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Disc.*, vol 31, pp. 606-660, 2017.

[12] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar and P.-A. Muller. Deep learning for time series classification: a review. *Data Min. Knowl. Disc.*, vol. 33, pp. 917-963, 2019.

[13] J. C. B. Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

[14] J. Lines and A. Bagnall. Time series classification with ensembles of elastic distance measures. *Data Min. Knowl. Disc.*, vol. 29, pp. 565-592, 2015.

[15] P. Schäfer. The BOSS is concerned with time series classification in the presence of noise. *Data Min. Knowl. Disc.*, vol. 29, pp. 1505-1530, 2015.

[16] A. Bagnall, J. Lines, J. Hills and A. Bostrom. Time-series classification with COTE: the collective of transformation-based ensembles. *IEEE Trans. Knowl. Data En.*, vol. 27, no. 9, pp. 2522-2535, 2015.

[17] J. Lines, S. Taylor, and A. Bagnall. Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. *ACM Trans. Knowl. Discov. D.*, vol. 12, no. 5, 2018

[18] H. A. Dau, A. Bagnall, K. Kamgar, C. -C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatanna and E. Keogh. The UCR time series archive. *arXiv preprint arXiv*: *1810.07758v2*, 2019.

[19] Y. Lecun, B. E. Boser, J. Denker and D. Henderson. Handwritten digit recognition with a back-propagation network. In *Proc. NeurIPS 1997*, Denver, USA, 1997.

[20] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao. Time series classification using multi-channels deep convolutional neural networks. In *Proc. WAIM 2014*, pp. 298-310, Macau, China, 2014.

[21] A. L. Guennec, S. Malinowski and R. Tavenard. Data augmentation for time series classification using convolutional neural networks. 2016.

[22] Z. Cui, W. Chen, and Y. Chen. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*, 2016.

[23] D. Li, J. Zhang, Q. Zhang and X. Wei. Classification of ECG signals based on 1D convolution neural network. In Proc. *IEEE 19th Healthcom 2017*, pp. 1-6, Dalian, China, 2017.

[24] U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, M. Adam, A. Gertych and R. S. Tan. A deep convolutional neural network model to classify heartbeats. Comput. Biol. Med., vol. 89, pp. 389-396, 2017.

[25] S. H. Huang, L. Xu, and C. Jiang. Residual attention net for superior cross-domain time sequence modeling. *arXiv preprint arXiv:2001.04077*, 2020.

[26] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Proc. Adv. Neural Inf. Process. Syst.* (*ANIPS 2017*), pp. 3856-3866. 2017.

[27] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[28] S. Ö. Arık, H. Jun and G. Diamos, Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Proc. Let.*, vol. 26, no. 1, pp. 94-98, 2019.

[29] G.E. Hinton, A. Krizhevsky and S.D. Wang. Transforming auto-encoders. In *Proc. ICANN 2011*, pp. 44-51, Berlin, Germany, 2011.

[30] M. K. Patrick, A. F. Adekoya, A. A. Mighty and B. Y. Edward. Capsule networks–a survey. *J. King Saud. Univ. Sci.*, vol 25, 2019.

[31] A. Dempster and F. Petitjean and G. I. Webb. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Knowl. Disc., vol. 34, pp. 1454-1495, 2020.

[32] A. Shifaz, C. Pelletier, F. Petitjean and G. I. Webb. TS-CHIEF: A scalable and accurate forest algorithm for time series classification. arXiv preprint arXiv: 1906.1032, 2019..

[33] J. Demšar. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res., vol. 7, pp. 1-30, 2006.

[34] M. Wenninger, S. P. Bayerl, J. Schmidt and K. Riehammer. Timage – a robust time series classification pipeline. *arXiv preprint arXiv*: 1909.91491v1, 2019.

[35] M. Dehghani, S. Gouws, O. Vinyals and Ł. Kaiser. Universal transformers. arXiv preprint arXiv: 1807.03819.

Reviewers' Comments

Reviewer #1: *This paper introduces a novel 3-head Capsule-based network to tackle the TSC problem. The proposed new architecture allows feature extraction and representation learning in addition to the temporal relationships within time series data. By removing the reconstruction module, the approach is much less complex than traditional capsule networks. An ablation study is included. The results are presented on a small subset of the UCR archive which makes the analysis not very accurate.*

Another problem is not including one of the most recent state-of-the-art neural networks called InceptionTime (that the authors do cite but not include in the comparison: https://link.springer.com/article/10.1007/s10618-020-00710-y).

Reference 31 should be removed from the paper and from the list of compared methods. The approach uses the test loss when training. See: https://github.com/titu1994/LSTM-FCN/issues/7

You should also not compare approaches based on mean accuracy as this is not informative at all. You should try and stick with a unified method for comparing multiple classifiers over multiple datasets. When comparing only two classifiers, you should use a pairwise accuracy plot. Finally, the paper could benefit from re-writing with better english phrases and include the most recent TSC approaches such as: TS-CHIEF, ROCKET and InceptionTime.

To summarize, I suggest that the authors take their time into re-writing the paper, including all state-of-the-art approaches and finish experiments on the whole archive instead of choosing subsets.

Reviewer #2: *The authors propose an interesting neural topology for time series classification. This network is composed of three CNNs (3-heads) with a capsule mechanism. The proposed method is evaluated on 44 standard datasets. The approach is interesting and clear enough for me and the results are convicting. However, I have some specific comments:*

- Table 2 is too small for reading. Please increase the font size. Moreover, the average value on the bottom of the table is missing. Add this value. Moreover, I don't see the significance to report the results with the six decimals. I think that three values are OK.

- I don't think that the computational complexity experiment gives some useful information. Moreover, the units are not clear for me. It is in second? I suggest to remove this experiment or describe it better.

- Table 4 is also too small for reading. Please increase the font size. Moreover, the evaluation metrics are not obvious for me. Please justify better them including the references. I prefer, to have the table A1 from Appendix instead.

Appendix

Table A1: The experiment results on 128 UCR datasets.

Dataset	Existing SOTA [25]	TS-CHIEF	Vanilla:ResNet-Transformer	ResNet-Trans1	ResNet-Trans2	ResNet-Trans3	ResNet 50 SC	ResNet 152 SC	Inception--Time	ROCKET	Ours
Adiac	**0.8570**	0.7980	0.843990	0.849105	0.849105	0.849105	0.844000	0.823000	0.841432	0.783376	0.849105
ArrowHead	0.8800	0.8327	0.891429	0.891429	0.891429	**0.897143**	0.885700	0.862900	0.845714	0.814286	0.868571
Beef	**0.9000**	0.7061	0.866667	0.866667	0.866667	0.866667	0.733300	0.766700	0.700000	0.833333	**0.900000**
BeetleFly	0.9500	0.9136	**1.000000**	0.900000	**1.000000**	0.700000	0.900000	0.900000	0.800000	0.900000	0.950000
BirdChicken	0.9500	0.9091	**1.000000**	0.950000	0.950000	**1.000000**	0.900000	**1.000000**	0.950000	0.900000	**1.000000**
Car	0.9330	0.8545	**0.950000**	0.883333	0.866667	0.300000	0.883300	0.900000	0.883333	0.846667	0.883333
CBF	**1.0000**	0.9979	**1.000000**	0.997778	**1.000000**	**1.000000**	0.904400	0.991100	0.998889	**1.000000**	**1.000000**
ChlorineCon.	0.8720	0.7167	0.849479	0.863281	0.409375	0.861719	0.784400	0.785200	0.876563	0.814531	**0.880435**
CinCECGTorso	**0.9949**	0.9832	0.871739	0.656522	0.890580	0.310870	0.891300	0.858000	0.853623	0.836159	0.950000
Coffee	**1.0000**	**1.0000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
Computers	0.8480	0.7051	0.860000	0.844000	**0.908000**	0.840000	0.740000	0.696000	0.796000	0.761200	0.840000
CricketX	0.8210	0.8138	0.838462	0.800000	0.810256	0.800000	0.735900	0.733300	**0.853846**	0.819487	0.838462
CricketY	0.8256	0.8019	0.838462	0.820513	0.825641	0.807692	0.735900	0.753800	**0.851282**	0.852308	0.838462
CricketZ	0.8154	0.8340	0.820513	0.805128	0.128205	0.100000	0.728200	0.761500	**0.861538**	0.855897	0.820513
DiatomSizeRe.	0.9670	0.9730	0.993464	0.996732	0.379085	**0.996732**	0.937900	0.934600	0.934641	0.969935	**0.996732**
DistalPhalanxO.A.G	**0.8350**	0.7462	0.812950	0.776978	0.467626	0.776978	0.784200	0.784200	0.733813	0.758993	0.784173
DistalPhalanxO.C.	0.8200	0.7823	**0.822464**	**0.822464**	**0.822464**	0.793478	0.815200	0.808000	0.782609	0.769565	0.800725
DistalPhalanxTW	0.6120	0.6704	0.564935	0.577922	0.551948	0.623377	**0.719400**	0.683500	0.683453	0.718705	0.683453
Earthquakes	**0.8010**	0.7482	0.755396	0.755396	0.762590	0.755396	0.777000	0.798600	0.741007	0.748201	0.762590
ECG200	0.9200	0.8618	0.940000	**0.950000**	0.940000	0.930000	0.870000	0.940000	0.930000	0.906000	0.930000
ECG5000	0.9482	0.9454	0.941556	0.943556	0.944222	0.940444	0.945800	0.944200	0.940889	0.947156	**0.948444**
ECGFiveDays	**1.0000**	**1.0000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.816500	0.902400	**1.000000**	**1.000000**	**1.000000**
ElectricDevices	**0.7993**	0.7553	0.774219	0.771625	0.757489	0.766178	0.731300	0.729700	0.723901	0.729413	0.769784
FaceAll	0.9290	0.8414	0.881065	0.848521	**0.949704**	0.252071	0.749700	0.782800	0.807101	0.946509	0.881065
FaceFour	**1.0000**	**1.0000**	0.954545	0.965909	0.977273	0.215909	0.727300	0.909100	0.954545	0.977273	0.954545
FacesUCR	0.9580	0.9663	0.957561	0.947805	0.926829	0.951220	0.777100	0.856600	**0.971220**	0.961415	0.951220
FiftyWords	0.8110	**0.8450**	—	—	—	—	0.798700	0.786800	0.830769	0.830330	0.830330
Fish	0.9890	0.9943	**1.000000**	0.977143	0.960000	0.994286	0.977100	0.977100	0.982857	0.979429	0.977143
FordA	**0.9727**	0.9410	0.948485	0.946212	0.517424	0.940909	0.938600	0.923500	0.961364	0.944394	0.948485
FordB	**0.9173**	0.8296	0.838272	0.830864	0.838272	0.823457	0.822200	0.807400	0.861728	0.805062	0.838272
GunPoint	**1.0000**	**1.0000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.993300	**1.000000**	**1.000000**	**1.000000**
Ham	0.7810	0.7152	0.761905	0.780952	0.619048	0.514286	**0.790500**	0.742900	0.714286	0.725714	0.761905
HandOutlines	0.9487	0.9322	0.937838	0.948649	0.835135	0.945946	0.951400	0.929700	**0.954054**	0.942432	0.945946
Haptics	0.5510	0.5168	0.564935	0.545455	**0.600649**	0.194805	0.516200	0.496800	0.548701	0.524026	0.564935
Herring	0.7030	0.5881	0.703125	0.734375	0.656250	0.703125	0.687500	0.656300	0.671875	0.692188	**0.750000**
InlineSkate	**0.6127**	0.5269	0.516364	0.494545	0.494545	0.165455	0.403600	0.430900	0.490909	0.456909	0.520000
InsectWingbeatS.	0.6525	0.6429	0.522222	0.642424	0.535859	0.536364	0.617700	0.621200	0.638889	**0.656818**	0.651010
ItalyPowerDemand	0.9700	0.9703	0.965015	0.969874	0.962099	**0.971817**	0.960200	0.960200	0.965015	0.969582	0.963071
LargeKitchenApp.	0.8960	0.8068	0.928000	0.898667	**0.936000**	0.933333	0.808000	0.757300	0.904000	0.900533	0.898667
Lightning2	**0.8853**	0.7481	0.852459	0.852459	0.754098	0.868852	0.852500	0.852500	0.770492	0.759016	0.786885

continued on next page

continued from previous page

Dataset	Existing SOTA [25]	TS-CHIEF	Vanilla:ResNet Transformer	ResNet-Trans1	ResNet-Trans2	ResNet-Trans3	ResNet 50 SC	ResNet 152 SC	Inception--Time	ROCKET	Ours
Lightning7	**0.8630**	0.7634	0.821918	0.849315	0.383562	0.835616	0.849300	0.794500	0.835616	0.823288	0.835616
Mallat	**0.9800**	0.9750	0.977399	0.975267	0.934328	0.979104	0.932600	0.944100	0.955224	0.955949	0.955224
Meat	**1.0000**	0.8879	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.983300	0.933333	0.948333	**1.000000**
MedicalImages	0.7920	0.7958	0.780263	0.765789	0.759211	0.789474	0.786800	0.780300	0.794737	**0.799474**	0.732895
MiddlePhalanxO.A.G.	**0.8144**	0.5832	0.655844	0.662338	0.623377	0.662338	0.636400	0.623400	0.551948	0.590260	0.623377
MiddlePhalanxO.C.	0.8076	0.8535	0.848797	0.848797	0.848797	0.835052	0.852200	**0.859100**	0.817869	0.838488	0.810997
MiddlePhalanxTW	0.6120	0.5502	0.564935	0.577922	0.551948	**0.623377**	0.623400	0.590900	0.512987	0.560390	0.610390
MoteStrain	**0.9500**	0.9475	0.940895	0.916933	0.937700	0.200000	0.840300	0.886600	0.886581	0.914617	0.881789
NonInvasiveFetalECG.	**0.9610**	0.9113	0.953181	0.953181	0.947583	0.948092	0.940500	0.940500	0.960814	0.952977	0.953181
NonInvasiveFetalECG.	0.9550	0.9450	0.955216	0.954198	0.948601	0.952672	0.952200	0.954700	0.963868	**0.969059**	0.952672
OliveOil	0.9333	0.8879	**0.966667**	0.900000	0.933333	0.900000	0.733300	0.766700	0.833333	0.916667	**0.966667**
OSULeaf	0.9880	0.9914	0.987603	**0.991736**	0.987603	**0.991736**	0.867800	0.851200	0.942149	0.940909	0.950413
PhalangesO.C.	0.8300	0.8450	0.855478	0.848485	0.854312	0.850816	0.857800	**0.859000**	0.849650	0.834266	0.850816
Phoneme	0.3492	0.3691	**0.363924**	0.191983	0.357595	0.348101	0.241000	0.241600	0.341245	0.279852	0.348101
Plane	**1.0000**	**1.0000**	**1.000000**	**1.000000**	0.371492	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
ProximalPhalanxO.A.G.	0.8832	0.8497	0.887805	**0.892683**	0.882927	**0.892683**	0.887800	0.873200	0.848780	0.855610	0.887805
ProximalPhalanxO.C.	0.9180	0.8882	**0.931271**	**0.931271**	0.683849	0.924399	0.924400	0.927800	**0.931271**	0.898969	0.900344
ProximalPhalanxTW	0.8150	0.8186	**0.819512**	0.814634	**0.819512**	**0.819512**	0.804900	0.819500	0.775610	0.816585	**0.819512**
RefrigerationDevices	0.5813	0.5583	0.605333	0.616000	0.592000	**0.618667**	0.552000	0.544000	0.517333	0.537333	0.605333
ScreenType	**0.7070**	0.5081	0.669333	0.645333	0.666667	0.680000	0.464000	0.472000	0.589333	0.485333	0.666667
ShapeletSim	**1.0000**	**1.0000**	**1.000000**	0.911111	0.888889	0.977778	0.555600	0.633300	0.955556	**1.000000**	0.577778
ShapesAll	0.9183	0.9300	0.923333	0.876667	0.921667	**0.933333**	0.885000	0.860000	0.928333	0.906833	0.918333
SmallKitchenApp.	0.8030	0.8221	0.808000	0.810667	**0.829333**	0.813333	0.733300	0.688000	0.760000	0.818400	0.813333
SonyAIBORobotSur.1	0.9850	0.8264	**0.988353**	0.978369	0.708819	0.985025	0.880200	0.960100	0.868552	0.922463	0.775355
SonyAIBORobotSur.2	0.9620	0.9248	0.976915	0.974816	**0.984260**	0.976915	0.814300	0.845800	0.946485	0.912592	0.950682
StarLightCurves	0.9730	**0.9824**	0.978873	0.979237	0.978873	0.975838	0.980600	0.980600	0.979359	0.980962	0.979237
Strawberry	0.9760	0.9663	**0.986486**	**0.986486**	**0.986486**	**0.986486**	0.981100	0.983800	0.983784	0.981351	**0.986486**
SwedishLeaf	0.9664	0.9655	**0.977200**	0.972800	0.969600	0.966400	0.968000	0.961600	0.974400	0.964000	0.972800
Symbols	0.9668	0.9766	0.979900	0.970854	0.976884	0.252261	0.971900	0.966800	**0.980905**	0.974271	0.950754
SyntheticControl	**1.0000**	0.9979	**1.000000**	0.996667	**1.000000**	**1.000000**	0.713300	0.670000	0.996667	0.999667	**1.000000**
ToeSegmentation1	0.9737	0.9653	0.969298	0.969298	0.978070	**0.991228**	0.916700	0.921100	0.964912	0.968421	0.964912
ToeSegmentation2	0.9615	0.9553	**0.976923**	0.953846	0.953846	**0.976923**	0.938500	0.884600	0.938462	0.923846	0.946154
Trace	**1.0000**	**1.0000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
TwoLeadECG	**1.0000**	0.9946	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.989500	0.995600	0.995610	0.999122	0.995610
TwoPatterns	**1.0000**	**1.0000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.515700	0.515400	**1.000000**	**1.000000**	**1.000000**
UWaveGestureL.All	**0.9685**	0.9689	0.856784	0.933277	0.939978	0.879118	0.937500	0.943600	0.951982	0.975377	0.939978
UWaveGestureL.X	0.8308	0.8411	0.780849	0.814629	0.810999	0.808766	0.707400	0.700400	0.824958	**0.854746**	0.810999
UWaveGestureL.Y	0.7585	0.7723	0.664992	0.716360	0.671413	0.678950	0.751300	0.713300	0.767169	**0.773981**	0.664992
UWaveGestureL.Z	0.7725	0.7844	0.756002	0.761027	0.760469	0.762144	0.728900	0.705200	0.764098	**0.791904**	0.760469
Wafer	**1.0000**	0.9991	0.998540	0.998215	0.998540	0.999027	0.997700	0.996600	0.998540	0.998232	0.998215
Wine	0.8890	0.8906	0.851852	0.870370	0.870370	**0.907407**	0.666700	0.833300	0.611111	0.812963	0.851852
WordSynonyms	**0.7790**	0.7874	0.661442	0.650470	0.636364	0.678683	0.685000	0.677100	0.733542	0.753448	0.678683
Worms	0.8052	0.8017	0.831169	0.779221	0.818182	0.259740	0.818200	**0.844200**	0.779221	0.740260	0.811812
WormsTwoClass	0.8312	0.8158	0.831169	0.779221	0.818182	0.259740	0.806600	**0.844200**	0.792208	0.797403	0.701299
Yoga	**0.9183**	0.8347	0.906333	0.905667	0.884000	0.866667	0.900000	0.882700	0.901667	0.910367	0.905667
ACSF1	—	—	**0.960000**	0.910000	0.930000	0.170000	0.780000	0.790000	0.920000	0.886000	0.880000
AllGestureWiimoteX	—	—	0.770000	0.760000	0.762857	0.754286	0.494300	0.520000	**0.790000**	**0.790000**	0.742857
AllGestureWiimoteY	—	—	0.814286	0.798571	0.808571	0.800000	0.600000	0.562900	**0.832857**	0.772714	0.808571
AllGestureWiimoteZ	—	—	0.782857	0.752857	0.767143	0.748571	0.651400	0.587100	**0.811429**	0.766143	0.757143
BME	—	—	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.993333	**1.000000**	**1.000000**
Chinatown	—	—	**0.985507**	**0.985507**	**0.985507**	**0.985507**	0.724600	0.756500	0.985423	0.982507	**0.985507**
Crop	—	—	0.743869	0.742738	0.746012	0.740476	0.755900	0.753200	0.772202	0.751345	**0.774702**
DodgerLoopD.	—	—	0.537500	0.550000	0.462500	0.500000	0.487500	0.512500	0.150000	0.572500	**0.762500**

continued on next page

continued from previous page

Dataset	Existing SOTA [25]	TS-CHIEF	Vanilla:ResNet Transformer	ResNet-Trans1	ResNet-Trans2	ResNet-Trans3	ResNet 50 SC	ResNet 152 SC	Inception--Time	ROCKET	Ours
DodgerLoopG.	—	—	0.876812	0.891304	0.550725	0.905797	0.681200	0.681200	0.855072	0.873188	**0.920290**
DodgerLoopW.	—	—	0.963768	0.978261	0.949275	0.963768	0.942000	0.971000	0.971014	0.974638	**0.985507**
EOGHorizontalSignal	—	—	0.610497	0.591160	0.602210	0.610497	0.279000	0.265200	0.593923	**0.638950**	0.602210
EOGVerticalSignal	—	—	0.450276	0.488950	0.146409	0.480663	0.223800	0.256900	0.475138	**0.541436**	0.450276
EthanolLevel	—	—	0.824000	**0.868000**	0.820000	0.820000	0.840000	0.710000	0.814000	0.582800	0.820000
FreezerRegularTrain	—	—	0.999649	0.999298	0.999298	**0.999649**	0.996100	0.997500	0.996491	0.997614	0.994035
FreezerSmallTrain	—	—	0.975088	0.958947	0.906667	0.771579	**0.979300**	0.935400	0.867368	0.949579	0.820702
Fungi	—	—	**1.000000**	**1.000000**	0.994624	0.075269	0.887100	0.919400	**1.000000**	**1.000000**	**1.000000**
GestureMidAirD1	—	—	0.715385	0.723077	0.723077	0.700000	0.530800	0.600000	**0.746154**	0.716923	0.723077
GestureMidAirD2	—	—	**0.746154**	0.692308	0.676923	0.700000	0.623100	0.553800	0.730769	0.660769	0.692308
GestureMidAirD3	—	—	0.353846	0.369231	0.338462	0.338462	0.415400	**0.461500**	0.400000	0.414615	0.407692
GesturePebbleZ1	—	—	**0.936047**	0.831395	**0.936047**	0.906977	0.918600	0.918600	0.924419	0.905814	0.866279
GesturePebbleZ2	—	—	0.873418	0.841772	**0.911392**	0.879747	0.854400	0.835400	0.886076	0.830380	0.797468
GunPointAgeSpan	—	—	0.996835	0.996835	**1.000000**	0.848101	0.987300	0.984200	0.987342	0.996835	0.981013
GunPointMaleV.F.	—	—	**1.000000**	**1.000000**	0.996835	0.996835	0.993700	0.993700	0.993671	0.998418	**1.000000**
GunPointOldV.Y.	—	—	**1.000000**	**1.000000**	**1.000000**	0.990476	0.981000	0.981000	0.965079	0.991111	**1.000000**
HouseTwenty	—	—	0.983193	0.907563	0.983193	0.991597	0.831900	0.840300	0.974790	0.963866	0.974790
InsectEPGRegularT.	—	—	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.971900	0.963900	**1.000000**	**1.000000**	**1.000000**
InsectEPGSmallT.	—	—	0.955823	0.927711	0.971888	0.477912	0.943800	0.879500	0.943775	0.979116	**1.000000**
MelbournePedestrian	—	—	0.912245	0.911837	0.904898	0.901633	0.360400	0.356300	**0.913899**	0.904387	0.911837
MixedShapesRegularT.	—	—	0.975670	0.969897	0.975670	**0.980206**	0.965400	0.950900	0.970309	0.971052	0.969897
MixedShapesSmallT.	—	—	0.910103	0.918763	0.928660	**0.940619**	0.902700	0.863500	0.914639	0.938227	0.910103
PickupGestureW.Z	—	—	0.800000	0.780000	0.780000	0.780000	0.740000	0.800000	0.760000	**0.830000**	0.800000
PigAirwayPre.	—	—	0.336538	0.091540	0.173077	0.153846	0.144200	0.168300	**0.543269**	0.095192	0.336538
PigArtPre.	—	—	**1.000000**	0.168269	0.043269	0.533654	0.351000	0.528800	0.995192	0.953846	0.918270
PigCVP	—	—	0.908654	0.081731	0.211538	0.019231	0.427000	0.528800	0.961538	**0.934135**	0.908654
PLAID	—	—	0.944134	0.921788	0.147114	**0.945996**	0.823100	0.811900	0.944134	0.902607	0.944134
PowerCons	—	—	0.933333	0.944444	0.927778	0.927778	0.938900	0.972200	0.944444	0.940000	**1.000000**
Rock	—	—	0.780000	**0.920000**	0.820000	0.760000	0.780000	0.840000	0.800000	0.900000	0.860000
SemgHandG.Ch2	—	—	0.866667	**0.916667**	0.848333	0.651667	0.786700	0.776700	0.816667	0.926833	**0.916667**
SemgHandM.Ch2	—	—	0.513333	0.504444	0.391111	0.468889	0.524400	0.526700	0.482222	0.645111	**0.593333**
SemgHandS.Ch2	—	—	0.746667	0.740000	0.666667	0.788889	0.664400	0.688900	0.824444	0.881111	**0.873333**
ShakeGestureW.Z	—	—	**0.940000**	**0.940000**	**0.940000**	**0.940000**	0.880000	**0.940000**	0.900000	0.898000	**0.940000**
SmoothSubspace	—	—	**1.000000**	**1.000000**	0.993333	**1.000000**	0.993300	0.986700	0.993333	0.978667	**1.000000**
UMD	—	—	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.826400	0.784700	0.986111	0.992361	**1.000000**