Supplementary Material: Adversarial Dynamic Shapelet Networks

Qianli Ma,[1] **Wanqing Zhuang,**[1] **Sen Li,**[1] **Desen Huang,**[1] **Garrison W. Cottrell**[2]
[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[2]Department of Computer Science and Engineering, University of California, San Diego, CA, USA
qianlima@scut.edu.cn, scutzwq@gmail.com

A. Datasets introduction and hyperparameters

In the experiment section, we report the results of ADSN on 18 UCR datasets and 8 UEA datasets. Here, we show the statistics of these 26 datasets and the hyperparameter settings of ADSN that we found via grid search in Figure 3.

Table 3: Statistics of datasets and the parameters of ADSN.

Dataset	#train/test	length	#class	dropout	k	L
Adiac	390/391	176	37	0	120	0.2,0.3
Beef	30/30	470	5	0	90	0.2,0.3
Chlorine.	467/3840	166	3	0	120	0.2,0.3
Coffee	28/28	286	2	0.25	90	0.3,0.4
Diatom.	16/306	345	4	0	120	0.7,0.8
DP_Little	400/645	250	3	0	30	0.1,0.2
DP_Middle	400/645	250	3	0	90	0.1,0.2
DP_Thumb	400/645	250	3	0.25	90	0.3,0.4
ECGFiveDays	23/861	136	2	0.25	120	0.7,0.8
FaceFour	24/88	350	4	0.5	60	0.3,0.4
Gun_Point	50/150	150	2	0.25	60	0.2,0.3
ItalyPower.	67/1029	24	2	0.25	60	0.5,0.6
Lightning7	70/73	319	7	0.25	90	0.2,0.3
MedicalImages	381/760	99	10	0.25	90	0.3,0.4
MoteStrain	20/1252	84	2	0.25	120	0.1,0.2
MP_Little	400/645	250	3	0.25	60	0.3,0.4
MP_Middle	400/645	250	3	0.25	60	0.2,0.3
Otoliths/Herring	64/64	512	2	0	120	0.1,0.2
PP_Little	400/645	250	3	0.25	60	0.1,0.2
PP_Middle	400/645	250	3	0.25	120	0.1,0.2
PP_Thumb	400/645	250	3	0	60	0.2,0.3
Sony.	20/601	70	2	0.5	30	0.7,0.8
Symbols	25/995	398	6	0.25	30	0.6,0.7
SyntheticC.	300/300	60	6	0.25	30	0.5,0.6
Trace	100/100	275	4	0.25	30	0.5,0.6
TwoLeadECG	23/1139	82	2	0.5	30	0.1,0.2

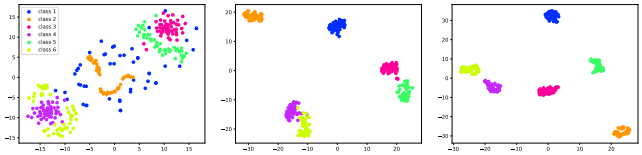
B. Introduction of the comparison methods

ADSN is compared with 6 shapelet-based methods, details of these model are as follows:

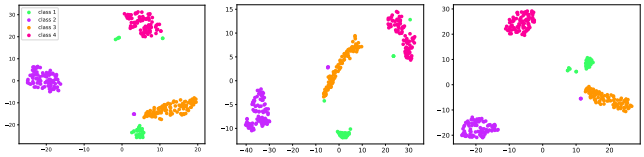
- **Fast shapelet (FSH)**: This method transforms the raw time series into a discrete and low dimensional representation, accelerating the shapelet searching process.
- **Scalable discovery (SD)**: This method prunes the shapelets that are similar, and use a supervised selection process to choose candidate shapelets based on how well they improve classification accuracy.
- **Learning time-series shapelets (LTS)**: This method uses the gradient descent algorithm to learn the shapelets directly.
- **Ultra-fast shapelet (UFS)**: This method uses random shapelets to accelerating the computation.
- **Shapelet transformation with linear SVM classifier (IGSVM)**: This method uses the shapelet transformation to obtain the new representation of time series and uses the linear SVM as classifier.
- **Fused lasso generalized eigenvector (FLAG)**: This method learns the shapelet positions by using a generalized eigenvector method, and uses a fused lasso regularizer to get a sparse and "block" solution.

ADSN is compared with 11 state-of-the-art methods, which can be divided into 4 categories:

- **Distance-based Methods:** These methods measure the similarity of two given time series through predefined

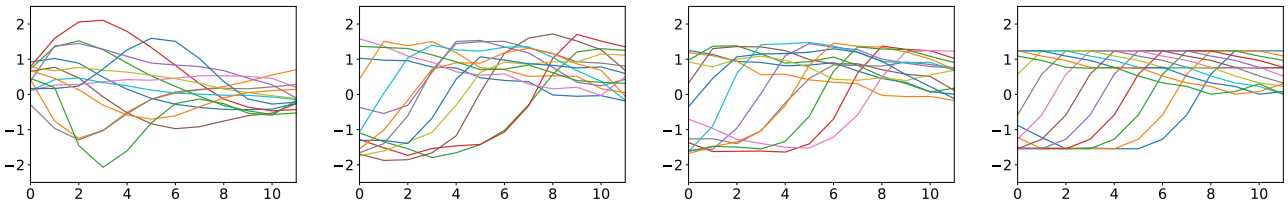


(a) SyntheticC.



(b) Diatom.

Figure 6: The visualizations with t-SNE on the datasets (a)*SyntheticC.* and (b)*Diatom*. The subfgure of each row from left to right are the original data distribution of the test set, the shapelet transformation representation using LTS and the shapelet transformation representation using ADSN, respectively. The colors of the points indicate the true labels.

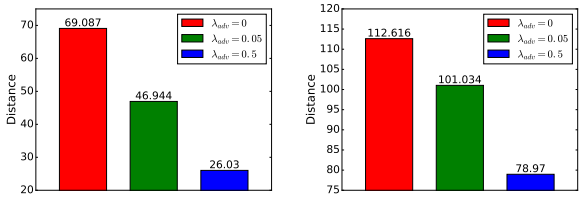


(a) λ_{adv}=0 (b) λ_{div}=0.05 (c) λ_{adv}=0.5 (d) True Slice

Figure 7: The generated shapelets of ADSN with different λ_{adv}. Here we set the number of shapelets equal to the number of slices for clearer observation.

similarity measures, and then the classification can be done by using k-nearest neighbors (kNN). DD_{DTW} is an approach using a weighted combination of the DTW distance between two time series and one between their corresponding first-order difference series. Based on DD_{DTW}, DTD_C further considers the DTW distance between the sequences transformed by the sin, cosine and Hilbert transform.

- **Feature-based Methods:** These methods extract representative features from the raw time series to represent local or global patterns, and then classifies them based on these. Bag of SFA symbols (BOSS) uses windows to form "words" over the series and uses a truncated Discrete Fourier Transform on each window to obtain the features. Time series forest (TSF) divides the time series into different intervals and calculates the mean, standard deviation and slope as the interval features, and then the intervals are randomly selected to train a forest of trees. Time series bag of features (TSBF) selects multiple random length subsequences from random locations, and then partitions these subsequences into shorter intervals to capture local information. Learned pattern similarity (LPS) is also based on intervals, but the main difference is that the subsequences themselves are used as attributes rather than the extracted interval features.
- **Ensemble-based Methods:** These methods combine different classifiers to achieve high performance. Elastic ensemble (EE) is the combination of 1-NN classifiers based on 11 elastic distance measures and uses a voting scheme to combine them. The collection of transformation ensembles (COTE) uses weighted votes over 35 different classifiers, where the weights are proportional to their cross-validation accuracy on the training data.
- **Deep Learning Methods:** These methods applied deep learning models to time series classification tasks. Multilayer Perceptrons (MLP) consists of three fully connected layers with 500 units per layer, and uses a softmax layer to get the final result. Fully Convolutional Networks (FCN) stacks three 1-D convolution blocks with 128, 256, and 128 filters in each block, with kernels of size 3, 5, and 8. After the convolution blocks, the features are fed into a global average pooling layer and a softmax layer to get the final result. FCN model uses ReLU activation function and batch normalization. Residual Network (ResNet) stacks three residual blocks comprised of three convolutional blocks each. The number of filters in the three residual blocks are 64, 128, and 128, respectively. ResNet also uses the global average pooling layer and a softmax layer.



(a) ItalyPower. (b) SyntheticC.

Figure 8: The distance between the shapelets of ADSN and subsequences with different λ_{adv}.

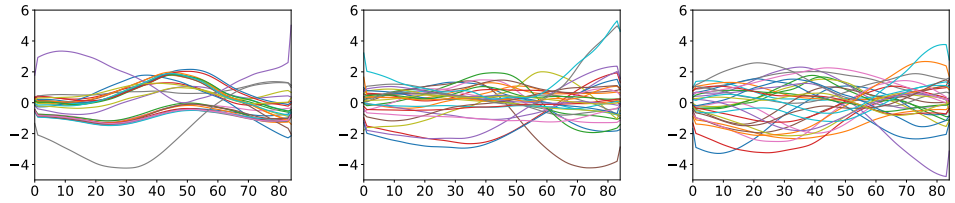
C. Effect of Dynamic Shapelet Transformation

To explore the effectiveness of dynamic shapelet transformation, we employ t-SNE to map the original time series and the shapelet transformation representation into a 2-D space, and then visualize the 2-D coordinates. As shown in Figure 6(a), the shapelet transformation of LTS mixes some samples belonging to different categories. In contrast, after the dynamic shapelet transformation of ADSN, the samples belonging to the same category are clustered together compactly while the distances between different classes are relatively large. In Figure 6(b), after the dynamic shapelet transformation of ADSN, the samples of class 1 are clustered together more compactly compared to LTS. This is evidence that sample-specific shapelets that are dynamically generated by ADSN improve the modeling flexibility and performance.

D. Effect of Adversarial training

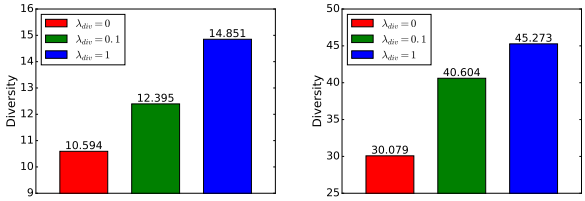
To explore the effects of adversarial training, we show the generated shapelets of ADSN under different values of regularization coefficient λ_{adv}.

As shown in Figure 7, without adversarial training, ADSN will generate shapelets that are dissimilar to the real subsequence. In contrast, ADSN can generate the shapelets that are similar to subsequences thanks to the adversarial strategy. The shape of shapelets is closer to real subsequences when increasing the value of λ_{adv}.



(a) λ_{div}=0 (b) λ_{div}=0.1 (c) λ_{div}=1

Figure 9: The generated shapelets of ADSN with different λ_{div} on *Coffee* datasets.



(a) Coffee (b) Diatom.

Figure 10: The diversity of shapelets generated by ADSN with different λ_{div}.

We quantitatively analyze the impact of λ_{adv} by calculating the distance between the generated shapelets and subsequences in two steps. Firstly, for each testing sample, we calculate the distance between this sample and all the generated shapelets by Equation 4 and sum these distances. Secondly, we average the results calculated by the first step for all testing samples. Therefore, the less the distance to real subsequences, the more similarity to them. As shown in Figure 8, the distance between shapelets and subsequences decrease when increasing λ_{adv}.

E. Effect of Diversity regularization

To explore the effects of diversity regularization, we show the generated shapelets of ADSN under different values of the regularization coefficient λ_{div}. As shown in Figure 9, the generated shapelets converge to similar shapes when the regularization coefficient is 0. When we increase the value of λ_{div}, the generated shapelets become more and more dissimilar from each other and diverge to different shapes. This shows that the diversity regularization can increase the diversity of generated shapelets and thus alleviate the mode collapse problem.

We quantitatively analyze the impact of λ_{div} by calculating the diversity of shapelets generated by ADSN with different λ_{div}. The diversity is calculated by

$$score_{div} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \sum_{j=1}^{k} \sum_{j'=1}^{k} d(\mathbf{s}_{i,j}, \mathbf{s}_{i,j'}) \qquad (1)$$

where n_{test} is the number of test samples, $d(\mathbf{s}_{i,j}, \mathbf{s}_{i,j'})$ is the Euclidean distance between two shapelets. As shown in Figure 10, the diversity score increase when increasing the value of λ_{div}.

F. Pseudo codes of ADSN

The training process of ADSN is shown in Algorithm 1.

Algorithm 1 Adversarial Dynamic Shapelet Networks

Require:
The number of epoch: maxEpoch, the number of batch in an epoch: maxBatch, the batch size n_b, the number of shapelets k, the length of shapelets L, the regularization parameters, λ_{div} and λ_{adv}, the learning rate η, the number of times the discriminator is optimized: maxIter. We fix maxIter to 3 in our experiments.

Ensure:
The parameters of Discriminator D: θ_D and the parameters of ADSN: θ_{ADSN}

for $epoch = 1, \ldots, maxEpoch$ **do**
  for $batch = 1, \ldots, maxBatch$ **do**
    Sample minibatch of n_b examples $\{(t_i, y_i)\}_{i=1}^{n_b}$ from dataset $\mathbb{D}$
    for $i = 1, \ldots, n_b$ **do**
      $\mathbf{O}_i = \mathbf{t}_{i,1:L} \oplus \mathbf{t}_{i,2:L+1} \oplus \cdots \oplus \mathbf{t}_{i,m-L+1:m}$
      for $j = 1, \ldots, k$ **do**
        $\mathbf{s}_{i,j} = \mathbf{W}_j * \mathbf{O}_i + b_j$
        for $j' = 1, \ldots, k$ **do**
          $\mathbf{G}_i(\mathbf{s}_{i,j}, \mathbf{s}_{i,j'}) = \exp(-\frac{d(\mathbf{s}_{i,j}, \mathbf{s}_{i,j'})}{\sigma^2})$
        end for
        $h_{i,j} = \min_{p=1,\cdots,P} \sqrt{\sum_{l=1}^{L}(t_{i,p+l-1} - s_{i,j,l})^2}$
      end for
      $\hat{\mathbf{y}}_i = \mathbf{W}_{out}\mathbf{h}_i$
    end for
    for $iter = 1, \ldots, maxIter$ **do**
      $L_D = -\sum_i \sum_p \log(D(\mathbf{t}_{i,p:p+L-1})) - \sum_i \sum_j \log(1 - D(\mathbf{s}_{i,j}))$
      $\theta_D = \theta_D - \eta \frac{\partial L_D}{\partial \theta_D}$
    end for
    $L_{cls} = -\frac{1}{n_b} \sum_{i=1}^{n} \sum_{r=1}^{c} 1\{y_{i,r} = 1\} \log \frac{\exp(\hat{y}_{i,r})}{\sum_{l=1}^{c} \exp(\hat{y}_{i,l})}$
    $L_{div} = \|\mathbf{G}_1 \oplus \mathbf{G}_2 \oplus \cdots \oplus \mathbf{G}_{n_b}\|_F^2$
    $L_{adv} = -\frac{1}{n_b \times k} \sum_{i=1}^{n_b} \sum_{j=1}^{k} \log(D(\mathbf{s}_{i,j}))$
    $L_{ADSN} = L_{cls} + \lambda_{div} L_{div} + \lambda_{adv} L_{adv}$
    $\theta_{ADSN} = \theta_{ADSN} - \eta \frac{\partial L_{ADSN}}{\partial \theta_{ADSN}}$
  end for
end for

G. Full results on 85 UCR datasets

The full result on 85 UCR datasets are shown in Table 4. ADSN ranks first among all the methods by achieving the

Table 4: Accuracies of ADSN and 11 state-of-the-art methods on 85 UCR datasets.

Dataset	DD_{DTW}	DTD_C	BOSS	TSF	TSBF	LPS	EE	COTE	MLP	FCN	ResNet	ADSN
Adiac	0.701	0.701	0.765	0.731	0.770	0.770	0.665	0.790	0.752	**0.857**	0.826	0.798
ArrowHead	0.789	0.720	0.834	0.726	0.754	0.783	0.811	0.811	0.823	0.880	0.817	**0.886**
Beef	0.667	0.667	0.800	0.767	0.567	0.600	0.633	0.867	0.833	0.750	0.767	**0.933**
BeetleFly	0.650	0.650	0.900	0.750	0.800	0.800	0.750	0.800	0.850	**0.950**	0.800	0.900
BirdChicken	0.850	0.800	0.950	0.800	0.900	**1.000**	0.800	0.900	0.800	0.950	0.900	0.850
Car	0.800	0.783	0.833	0.767	0.783	0.850	0.833	0.900	0.833	0.917	**0.933**	0.917
CBF	0.997	0.980	0.998	0.994	0.988	0.999	0.998	0.996	0.860	**1.000**	0.994	0.999
Chlorine	0.708	0.713	0.661	0.720	0.692	0.608	0.656	0.727	0.872	0.843	0.828	**0.880**
CinCECGtorso	0.725	0.852	0.887	0.983	0.712	0.736	0.942	**0.995**	0.842	0.813	0.771	0.976
Coffee	**1.000**	**1.000**	**1.000**	0.964	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Computers	0.716	0.716	0.756	0.720	0.756	0.680	0.708	0.740	0.540	**0.848**	0.824	0.604
CricketX	0.754	0.754	0.736	0.664	0.705	0.697	0.813	0.808	0.569	0.815	**0.821**	0.723
CricketY	0.777	0.774	0.754	0.672	0.736	0.767	0.805	**0.826**	0.595	0.792	0.805	0.723
CricketZ	0.774	0.774	0.746	0.672	0.715	0.754	0.782	**0.815**	0.592	0.813	0.813	0.726
Diatom	0.967	0.915	0.931	0.931	0.899	0.905	0.944	0.928	0.964	0.930	0.931	**0.987**
DistPhxAgeGp	0.705	0.662	0.748	0.748	0.712	0.669	0.691	0.748	0.827	0.835	0.798	**0.843**
DistPhxCorr	0.732	0.725	0.728	0.772	0.783	0.721	0.728	0.761	0.810	0.812	0.820	**0.822**
DistPhxTW	0.612	0.576	0.676	0.669	0.676	0.568	0.647	0.698	0.747	0.790	0.740	**0.798**
Earthquakes	0.705	0.705	0.748	0.748	0.748	0.640	0.741	0.748	0.792	0.801	0.786	**0.812**
ECG200	0.830	0.840	0.870	0.870	0.840	0.860	0.880	0.880	**0.920**	0.900	0.870	**0.920**
ECG5000	0.924	0.924	0.941	0.939	0.940	0.917	0.939	0.946	0.935	0.941	0.931	**0.947**
ECGFiveDays	0.769	0.822	**1.000**	0.956	0.877	0.879	0.820	0.999	0.970	0.985	0.955	**1.000**
ElectricDevices	0.592	0.594	**0.799**	0.693	0.703	0.681	0.663	0.713	0.580	0.723	0.728	0.585
FaceAll	0.902	0.899	0.782	0.751	0.744	0.767	0.849	0.918	0.885	**0.929**	0.834	0.812
FaceFour	0.830	0.818	**1.000**	0.932	**1.000**	0.943	0.909	0.898	0.830	0.932	0.932	0.977
FacesUCR	0.904	0.908	0.957	0.883	0.867	0.926	0.945	0.942	0.815	0.948	**0.958**	0.932
FiftyWords	0.754	0.754	0.705	0.741	0.758	0.818	**0.820**	0.798	0.712	0.679	0.727	0.686
Fish	0.943	0.926	**0.989**	0.794	0.834	0.943	0.966	0.983	0.874	0.971	0.989	0.931
FordA	0.723	0.765	0.930	0.815	0.850	0.873	0.738	**0.957**	0.769	0.906	0.928	0.933
FordB	0.667	0.653	0.711	0.688	0.599	0.711	0.662	0.804	0.629	0.883	0.900	**0.918**
GunPoint	0.980	0.987	**1.000**	0.973	0.987	0.993	0.993	**1.000**	0.933	1.000	0.993	0.987
Ham	0.476	0.552	0.667	0.743	0.762	0.562	0.571	0.648	0.714	0.762	**0.781**	**0.781**
HandOutlines	0.868	0.865	0.903	**0.919**	0.854	0.881	0.889	**0.919**	0.807	0.776	0.861	0.871
Haptics	0.399	0.399	0.461	0.445	0.490	0.432	0.393	0.523	0.461	**0.551**	0.506	0.458
Herring	0.547	0.547	0.547	0.609	0.641	0.578	0.578	0.625	0.687	**0.703**	0.594	**0.703**
InlineSkate	**0.562**	0.509	0.516	0.376	0.385	0.500	0.460	0.495	0.351	0.411	0.365	0.373
InsWngSnd	0.355	0.473	0.523	0.633	0.625	0.551	0.595	**0.653**	0.631	0.402	0.531	0.606
ItalyPower	0.950	0.951	0.909	0.960	0.883	0.923	0.962	0.961	0.966	0.970	0.960	**0.972**
LrgKitApp	0.795	0.795	0.765	0.571	0.528	0.717	0.811	0.845	0.480	**0.896**	0.893	0.635
Lightning2	0.869	0.869	0.836	0.803	0.738	0.820	**0.885**	0.869	0.721	0.803	0.754	0.777
Lightning7	0.671	0.658	0.685	0.753	0.726	0.740	0.767	0.808	0.644	**0.863**	0.836	0.808
Mallat	0.949	0.927	0.938	0.919	0.960	0.908	0.940	0.954	0.936	**0.980**	0.979	0.919
Meat	0.933	0.933	0.900	0.933	0.933	0.883	0.933	0.917	0.933	0.967	**1.000**	0.967
MedicalImages	0.737	0.745	0.718	0.755	0.705	0.746	0.742	0.758	0.729	**0.792**	0.772	0.720
MidPhxAgeGp	0.539	0.500	0.545	0.578	0.578	0.487	0.558	0.636	0.735	0.768	0.760	**0.771**
MidPhxCorr	0.732	0.742	0.780	**0.828**	0.814	0.773	0.784	0.804	0.760	0.795	0.793	0.797
MidPhxTW	0.487	0.500	0.545	0.565	0.597	0.526	0.513	0.571	0.609	0.612	0.607	**0.617**
MoteStrain	0.833	0.768	0.879	0.869	0.903	0.922	0.883	0.937	0.869	**0.950**	0.895	0.906
NonInvThor1	0.806	0.841	0.838	0.876	0.842	0.812	0.846	0.931	0.942	**0.961**	0.948	0.902
NonInvThor2	0.893	0.890	0.901	0.910	0.862	0.841	0.913	0.946	0.943	**0.955**	0.951	0.917
OliveOil	0.833	0.867	0.867	0.867	0.833	0.867	0.867	**0.900**	0.400	0.833	0.867	0.867
OSULeaf	0.880	0.884	0.955	0.583	0.760	0.740	0.806	0.967	0.570	**0.988**	0.979	0.649
PhalCorr	0.739	0.761	0.772	0.803	**0.830**	0.756	0.773	0.770	**0.830**	0.826	0.825	0.796
Phoneme	0.269	0.268	0.265	0.212	0.276	0.237	0.305	**0.349**	0.098	0.345	0.324	0.227
Plane	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.981	**1.000**	**1.000**	**1.000**
ProxPhxAgeGp	0.800	0.795	0.834	0.849	0.849	0.795	0.805	0.854	0.824	0.849	0.849	**0.859**
ProxPhxCorr	0.794	0.794	0.849	0.828	0.873	0.842	0.808	0.869	0.887	0.900	**0.918**	0.880
ProxPhxTW	0.766	0.771	0.800	0.815	0.810	0.732	0.766	0.780	0.797	0.810	0.807	**0.818**
RefDev	0.445	0.445	0.499	**0.589**	0.472	0.459	0.437	0.547	0.371	0.533	0.528	0.421
ScreenType	0.429	0.437	0.464	0.456	0.509	0.416	0.445	0.547	0.408	0.667	**0.707**	0.451
ShapeletSim	0.611	0.600	**1.000**	0.478	0.961	0.867	0.817	0.961	0.483	0.867	**1.000**	0.883
ShapesAll	0.850	0.838	0.908	0.792	0.185	0.873	0.867	0.892	0.775	0.898	**0.912**	0.818
SmlKitApp	0.640	0.648	0.725	**0.811**	0.672	0.712	0.696	0.776	0.389	0.803	0.797	0.667
SonyAIBOSurf1	0.742	0.710	0.632	0.787	0.795	0.774	0.704	0.845	0.727	0.968	**0.985**	0.915
SonyAIBOSurf2	0.892	0.892	0.859	0.810	0.778	0.872	0.878	0.952	0.839	**0.962**	0.962	0.940
StarlightCurves	0.962	0.962	0.978	0.969	0.977	0.963	0.926	**0.980**	0.957	0.967	0.975	0.957
Strawberry	0.954	0.957	0.976	0.965	0.954	0.962	0.946	0.951	0.967	0.969	0.958	**0.977**
SwedishLeaf	0.901	0.896	0.922	0.914	0.915	0.920	0.915	0.955	0.893	**0.966**	0.958	0.916
Symbols	0.953	0.963	**0.967**	0.915	0.946	0.963	0.960	0.964	0.853	0.962	0.872	0.963
SyntheticControl	0.993	0.997	0.967	0.987	0.993	0.980	0.990	**1.000**	0.950	0.990	**1.000**	**1.000**
ToeSegmentation1	0.807	0.807	0.939	0.741	0.781	0.877	0.829	**0.974**	0.601	0.969	0.965	0.921
ToeSegmentation2	0.746	0.715	**0.962**	0.815	0.800	0.869	0.892	0.915	0.746	0.915	0.862	0.930
Trace	**1.000**	0.990	**1.000**	0.990	0.980	0.980	0.990	**1.000**	0.820	**1.000**	**1.000**	**1.000**
TwoLeadECG	0.978	0.985	0.981	0.759	0.866	0.948	0.971	0.993	0.853	**1.000**	**1.000**	0.986
TwoPatterns	**1.000**	**1.000**	0.993	0.991	0.976	0.982	**1.000**	**1.000**	0.886	0.897	**1.000**	0.959
UWaveGestX	0.779	0.775	0.762	0.804	**0.831**	0.829	0.805	0.822	0.768	0.754	0.787	0.771
UWaveGestY	0.716	0.698	0.685	0.727	0.736	**0.761**	0.726	0.759	0.703	0.725	0.668	0.669
UWaveGestZ	0.696	0.679	0.695	0.743	**0.772**	0.768	0.724	0.750	0.705	0.729	0.755	0.682
UWaveGestAll	0.935	0.938	0.939	0.957	0.926	0.966	0.968	0.964	0.954	0.826	0.868	**0.976**
Wafer	0.980	0.993	0.995	0.996	0.995	0.997	0.997	**1.000**	0.996	0.997	0.997	0.997
Wine	0.574	0.611	0.741	0.630	0.611	0.630	0.574	0.648	0.796	**0.889**	0.796	0.815
WordSynonyms	0.730	0.730	0.638	0.647	0.688	0.755	**0.779**	0.757	0.594	0.580	0.632	0.647
Worms	0.584	0.649	0.558	0.610	0.688	**0.701**	0.662	0.623	0.343	0.669	0.619	0.580
WormsTwoClass	0.649	0.623	**0.831**	0.623	0.753	0.753	0.688	0.805	0.597	0.729	0.735	0.680
Yoga	0.856	0.856	**0.918**	0.859	0.819	0.869	0.879	0.877	0.855	0.845	0.858	0.845
best	5	3	13	4	6	5	6	17	3	22	16	**25**

best results on 25 datasets. FCN ranks second with the best results on 22 datasets.

H. Runtime comparison

We first compare the run time of the full ADSN model and its ablation models. We run the experiments on ECGFiveDays dataset 5 times and compare the average run time. ADSN takes 0.177s per epoch, while ADSN without adv takes 0.099s per epoch and ADSN without div takes 0.162s. Then we compare ADSN and LTS model (the best competitor). The comparison to LTS is difficult since the implementation of LTS is on a CPU instead of a GPU, and LTS needs to initialize the shapelets by k-means. Hence, it is much slower than ADSN (e.g., LTS takes 3.818s per epoch).

I. Robust analysis

To explore whether ADSN is a more robust model, we conduct the experiments on ECGFiveDays dataset. Specifically, we add the Gaussian noise to the test samples and compare the accuracy of ADSN and LTS. When adding Gaussian noise to the data point of 5% of the testing data, the accuracy of LTS and ADSN drops to 0.938 and 0.979, respectively. When adding Gaussian noise to data point of 10% of the testing data, the accuracy of LTS drops to 0.870, while the accuracy of ADSN slightly drops to 0.970. It proves that ADSN is more robust than LTS.