On the Mining of the Minimal Set of Time Series Data Shapelets

Soukaina Filali Boubrahimi
Department of Computer Science
Utah State Univeristy
Logan, Utah 84341
Email: soukaina.boubrahimi@usu.edu

Shah Muhammad Hamdi
Department of Computer Science
New Mexico State Univeristy
Las Cruces, NM 88003
Email: shamdi1@nmsu.edu

Ruizhe Ma
Departement of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854
Email: ruizhe_ma@uml.edu

Rafal Angryk
Departement of Computer Science
Georgia State University
Atlanta, GA 30302
Email: rangryk@cs.gsu.edu

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BigData50022.2020.9378454

***Abstract*—Shapelets, also known as motifs, are time series sequences that have the property of discriminating between time series classes. Lately, shapelets studies have gained a lot of momentum due to their interpretable nature. As opposed to traditional time series classifiers, shapelet-based learners provide a visual representation of the pattern that triggers the classification decision. One of the most challenging issues of shapelet-based classifiers is the generation of a large number of shapelet outputs. To the best of our knowledge, this is the first effort that addresses the high numerosity problem of mined shapelets issue by mining the minimal set of discriminative shapelets for time series data. We propose a new shapelet mining learner, *1DCNN*, that has the property of learning shapelets of different lengths using a black-box neural network model. *1DCNN* optimizes the entire classification schema by learning the shapes of the representative patterns. Our proposed model uses network pruning to sparsify the network and keep only the most discriminative shapelets without compromising the classification accuracy. We validated our model using 59 real-world time series datasets from the UCR repository. Our experimental results show the effectiveness and efficiency of our approach in comparison with other competing baselines models. For fairness purposes, we did not compare 1DCNN with ensemble based approaches that encapsulates many learners. Our results show that the performance of our model is superior to all other baselines pertaining to the shapelet-based classifier category, with up to 95% less Floating Points Operations per Second (FLOPs) required by the network.**

***Index Terms*—Shapelet Mining Algorithm, Time Series Classification, One-dimensional convolutional neural network.**

I. Introduction

Time series classification is a pervasive problem in real-world applications from various domains such astronomy, aerospace, and meteorology [1][2][3]. Given the need to accurately classify time series data, a number of well-established baselines have been proposed in the literature to tackle the problem that evolved from simple similarity-based estimators [4] that works on raw time series data with some pre-defined similarity measures (such as Euclidean distance and Dynamic Time Warping [5]), to more complex methods that involve either dimensionality and noise reduction of the input space or a data transformation to another domain (such as frequency domain) [6]. Deep learning models have shown to be promising for computer vision application [7] [8]. Motivated by this fulfillment, we propose a new shapelet mining algorithm that combines the efficiency of black-box deep learning models for time series classification and network pruning for the mining of the minimal interpretable shapelets. Our novelty comes into two-folds: (1) designed a network topology that allows the learning of shapelets of different size in parallel, (2) we generalized two-dimensional convolutional networks traditionally used in image data for the case of one-dimensional time series input data, and to mine the minimal number of shapelets needed for classification. This is first attempt to learn the minimal number of various lengths shapelets for time series data that are traditionally user-set parameters fed to the learning model. In a nutshell, 1DCNN model aims to correctly classify the test instance as well as generate few interpretable pattern for domain-expert to visually examine.

Problem Formalism: *Assume that a feature space represented by a mutually exclusive set of classes* $C = \{C_1, C_2, \ldots, C_M\}$. *A shapelet-based miner, or expert, predicts the class of a data sample* $x \in C$ *as* $e(x, S) = j$ *using a set of shapelets* $S = \{S_1, S_2, \ldots, S_k\}$ *such that* $j \in \{1, \ldots, M\}$ *and* $k \in \{1, \ldots, K\}$. *Shapelets minimization consists of finding a subset* $S^{'} \subseteq S$ *such that the number of covered elements* $\left|\bigcup_{S_i \in S'} S_i\right|$ *is minimized.*

The rest of the paper is organized as follows: in Section II we review the related works; Section III defines the architecture of the proposed $1DCNN$ method, followed by the pruning strategy in Section IV. Section V explains the experimental setup, followed by a discussion of the results of our experiments presented in Section VI. Finally, Section VII concludes the paper and shows directions for future works.

II. Related Work

Time series classifiers can be grouped under 5 main branches: (1) similarity-based techniques, (2) interval-based, (3) dictionary-based, (4) Shapelets Mining and, (5) deep leaning ensemble. Similarity and interval based classifiers are the most simple approaches. The similarity-based methods use an instance based learner (such as k-Nearest Neighbor(kNN) classifier) coupled with a distance measure to compare time series [9]. While interval-based methods create a new feature space by extracting features from random time series intervals statistics [10][11][12], dictionary-based classifiers are inspired by the bag-of-words (BOW) model from natural language processing field [13]. The methods first transform the input time series into representative word distributions and then feed them to an instance based learner[6]. Shapelets mining algorithms have gained interest lately due to their interpretability and efficiency.

Shapelets, also known as motifs, are discriminative time series segments that have the property of modeling unique patterns appearing in one (or a subset of) class(es) of the multi-class classification problem. Traditional shapelets mining approaches, such as Shapelet Decision Tree [14] and Shapelet Transform [15], consist of extracting sub-series from the training set and selecting only the sub-sequences with high discriminatory power. The latter approaches have a time complexity of $O(n^2.T^4)$ (where n is the number of classes and T is the length of the time series) which makes it hard to adopt for large-scale datasets. Fast Shapelet is another variant that was designed to reduce the time complexity to $O(n.T^2)$ by compromising accuracy [16]. Grabocka et al. [17] proposed the second approach of shapelets mining which formalizes an optimization problem that jointly learns the shapelets from the training data and minimizes their incurred error [17].

The first attempt to use a neural network approach for classifying univariate time series data, as appeared in [8], consisted of a relatively simple four-layered Multi-Layer Perceptron (MLP) where each layer of the network is a fully-connected layer of 500 neurons with a ReLU activation function. MLP achieved the worst performance level compared to all state-of-the-art univariate time series classifiers. Fully Convolutional Neural Network (FCN) models were the second attempt to approach the univariate time series classification problem [8]. FCN architecture is composed of three convolution blocks, used as feature extractors, each of which is composed of a two-dimensional kernel followed by batch normalization [18] and ReLU activation function. FCN achieved the best accuracy levels in the deep learning classifiers category; however, the two-dimensional kernels of the network are not interpretable since the input is in a one-dimensional space. Residual network (ResNet), also proposed by [8], is the deepest architecture that extends traditional neural networks by adding a shortcut connection between each consecutive residual block that enables the flow of the gradient directly to the next layer. ResNet model achieves results with same statistical significance of four other baselines (FCN [8], COTE [4], MCNN [19], and BOSS [6]). This finding suggests that with deeper network architecture, ResNet is more prone to overfitting.

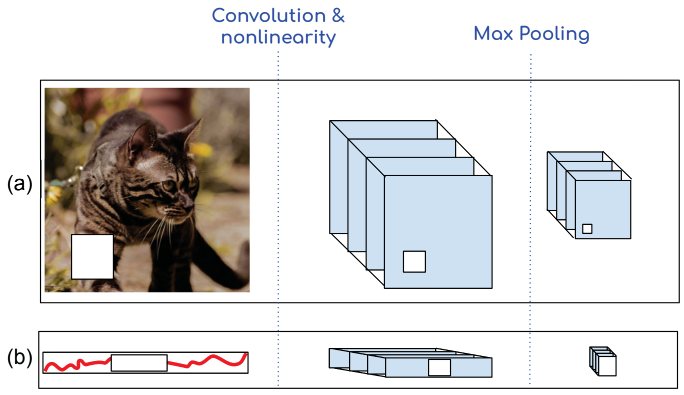


Fig. 1. Analogy of two-dimensional CNN for image segementation and 1DCNN for time series data

Another reason for low performance in deeper architecture is the significant redundancy that has been demonstrated, which is mainly caused by the overwhelming amount of parameters in deep neural networks. An over-parameterized model not only wastes memory and computation, but also leads to a serious overfitting problem. Therefore, reducing the number of parameters has been studied by many researchers in this field. However, there is little work directly addressing the optimization of the number of neurons. Most previous works on improving network architectures fall into two main categories; one concentrates on the high-level architectural design and the other focuses on low-level weight pruning.

Exhaustive search shapelet mining algorithm has the highest time complexity of $O(n^2.T^4)$, followed by similarity-based approaches with $O(nk+nT)$. Learning Shapelets and 1DCNN have both a complexity of $O(n.T^2.l)$, while Fast Shapelet algorithm complexity is $O(n.T^2)$ (where n is the number of instances, l is the number of classes, T is the length of the time series instances). In this paper, we propose a new mining model that pertains to both deep learning and shapelet mining methods. The idea is to approach the classification problem from an optimization perspective by leveraging convolutional neural network models, that have shown to have compelling efficiency for semantic image segmentation, without compromising the model interpretability which is captured through the mined shapelets.

III. Network Architecture

The 1DCNN model is the core of shapelet-based and prediction-based queries. The general network architecture consists of a four-layered convolutional neural network. The original convolution neural networks, initially designed for image processing, takes a two-dimensional image input that is convolved with a three-dimensional (tensor) kernel in the next layer. In analogy with traditional CNN, our input is a one-dimensional time series followed by a two-dimensional/matrix kernel in the convolution layer. Figure 1-a illustrates the analogy of the traditional two-dimensional convolutional network and our proposed 1DCNN model in Figure 1-b. Following the same line of thoughts as [6], we hypothesize that a dataset can have discriminative shapelets of different lengths all of which can contribute with independent information to the classification. As a matter of fact, we considered three shapelet sizes in the same model. Our model architecture is shown in Figure 3 where the first layer represents the input time series data matrix that is passed to the convolutional layer. The former convolves the same matrix three times using filter matrices of different sizes and pass it to the fully connected layer after applying a max-pooling and non-linear activation function. The max-pooling operation preserves only the convolution operation that led to the maximum sum when sliding the kernel over the time series. In other terms, the only similarity between shapelet and the best alignment in the time series is taken into consideration for the classification. Figure 2 illustrates the idea behind one-dimensional convolution and max pooling operations in the context of shapelets. A candidate shapelet is overlaid over the input time series and shifted to the right along the time dimension until all the possible alignments are considered. The result of each convolution operation between the shapelet and the time series segments is recorded in a convolution vector that is then passed to the max-pooling layer which keeps the element with the maximum value in the vector. In Figure 2 9 is the maximum element of the convolutional vector that represents the fourth alignment. The concatenation of the max convolution elements of all the kernels represents the new feature space input for the fully connected layer

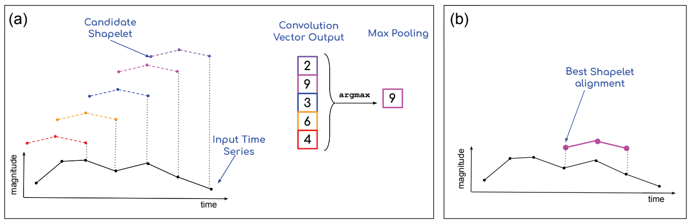


Fig. 2. (a) Illustration of the convolution process of the candidate shapelet (shown in dotted line) over the input time series (shown in solid line) and the max pooling operation and (b) the best candidate shapelet alignment that resulted

Finally the last layer consists of the softmax function that produces probability likelihoods of a time series instance belonging to a particular class. The basic convolution operations are defined in Equation 1.

$$\begin{aligned} y &= W \circledast x + b \\ h &= ReLU(y) = max(0, y) \\ s &= max(h) \end{aligned} \quad (1)$$

where $\circledast$ is the convolution operation. We train our 1DCNN with backpropagation in conjunction with Adaptive Moment Estimation (Adam) [20] optimization algorithm. Algorithm 1 shows the full mini-batch gradient descent algorithm that we used.

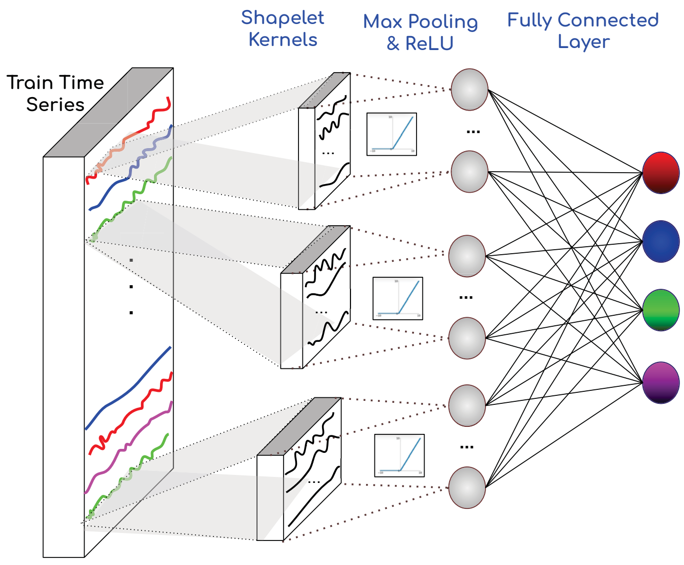


Fig. 3. 1DCNN General Architecure for a four-class time series classification problem

The model architecture has a fixed number of kernels that have three different lengths. The number of kernels in each of the convolution units is set to 300.

As mentioned earlier, we designed our model to be able to capture patterns of different-length following our initial hypothesis that states that a dataset can have more than one optimal pattern size as shown in [17]. The second assumption of variable 1DCNN is that patterns should be proportional to the initial time series size. The idea is that shapelet lengths should be proportional to the time series lengths in the database and should not be set to a fixed length for all prediction problems. In other terms, for shorter time series of size 100, shapelets of size 12 are good candidates to capture discriminate patterns, while the same size could be too small for time series of length 1,000. For the latter dataset, shapelets of length 12 could capture very granular micro-behaviors that could represent noise. The statement is especially true in our case since we do not use any dimensionality reduction pre-processing step on the raw time series. We used the same shapelets lengths window suggested in [17]: $len_i \in \{0.3, 0.5, 0.7\}$ x time series size (where $i \in [1, 3]$). Figure 4-(b) illustrate the variable kernel size (width) with respect to the dataset time series size in Figure 4-(a).

As a result, another indirect size variability that affects the network topology occurs in the concatenation and fully connected layers (third layer in Figure 3) due to the variation in size (length and width) of the former convolution layer (second layer in Figure 3). The number of the input neurons equals the number of max-pooling operations which equates the length of the kernels x 3 (since there are three parallel convolution blocks). In this case, the number of neurons on the fully connected layer is 900 (300 kernels * 3 layers).

Algorithm 1 1DCNN mini-batch Gradient Descent Algorithm

Input: Dataset $\mathscr{D}$, learning rate α, len_1 size of the first kernels, len_2 size of the second kernels, len_3 size of the third kernels,number of epochs ep , mini-batch size n_m

Output: The trained $network$

```
1:  $iteration \leftarrow 0$
2:  while $iteration < ep$ do
3:      ▷ Propagate the input (Forward Pass)
4:      for $j$ = 1,..., $n_m$ do
5:          for $i$ in $[len_1, len_2, len_3]$ do
6:              $I_j = W_{len} \circledast x + \theta$
7:              $H_j = ReLU(I_j)$
8:              $O_j = maxPool(H_j)$
9:              $y_{hat_j} = Softmax(O_j)$
10:             $Loss_{ij} = -\frac{1}{N}\sum_{c=1}^{m} y_{hat_j} log(p_j)$
11:         end for
12:         ▷ Backpropagate the errors (Backward Pass)
13:         for $i$ in $[len_1, len_2, len_3]$ do
14:             $W_i = ADAM(\sum_1^{n_m} Loss_j, W_i, \alpha, \beta_1, \beta_2)$
15:         end for
16:     end for
17:     $iteration \leftarrow iteration + 1$
18: end while
19: return $network$
```

IV. Network Pruning

Network pruning is used in this application to reduce the number of shapelets learned by the model. The resulting pruned model is a more robust network having a better generalization capability. In addition, the pruned model retains the exact number of shapelets necessary to keep the same performance levels of the original unpruned model. The proposed pruning methods are a four-steps procedure, as illustrated in Figure 5, that takes the original trained model (M) as an input. The first step of the procedure is to measure the importance of all the (initial 900) shapelets/kernels of the unpruned model using a weighting heuristic. The second step of the pruning procedure consists of deleting the $r\%$ least important shapelets in the network and then fine-tuning the new network (M$_{pruned1}$). The later step consists of using the weights of (M) as a weight initialization for the new model (M$_{pruned1}$) before training it on the same train data. Finally, the above three steps are repeated until the maximum sparsity level (K%) is reached.

We used the Average Percentage of Zeros (APoZ) as the pruning criterion to measure the shapelets' importance of the intermediate models. APoZ criteria was proposed by Hu et.al in [21] and it stems from the idea that since a large portion of zero activation function outputs exists in a neural networks, a network can be pruned by trimming the neurons that did not highly contribute to the decision.

APoZ is formally defined as the percentage of neurons that did not fire (have zero activation) after applying the ReLU function. In addition to pointing the most discriminative

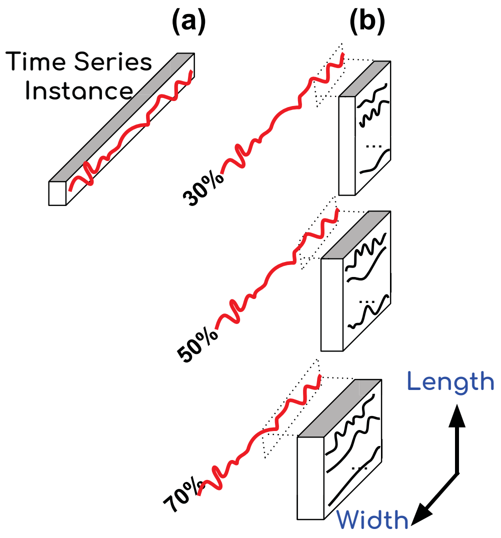


Fig. 4. (a) Example of input time series used to size the convolutional layer kernels in (b) and fixed-sized kernels

shapelets in the network, APoZ is also a general measure of redundancy in the model due to the overwhelming amount of parameters in the network which is generally the cause of an overfit model and inaccurate queries. The canonical form of the APoZ measure is shown in Equations 2 and 3.

$$APoZ_c^i = APoZ(O_c^i) = \frac{\sum k^N \sum j^M f(O_{c,j}^{(i)}(k))}{N * M} \tag{2}$$

$$\begin{cases} f(.) = 0, & \textbf{if} \quad O_{c,j}^{(i)}(k) = 0 \\ f(.) = 1, & \textbf{if} \quad O_{c,j}^{(i)}(k)! = 0 \end{cases} \tag{3}$$

where O_c^i is the output of the c-th channel in the i-th layer, M is the dimension of the feature map, N is the number of data points, and $f(.)$ is an indicator function that evaluates to 1 in case the output of the activation is zero as shown in Equation 3.

V. Experimental Setup

A. Datasets

The performance of different variants of the original and pruned 1DCNN methodology and other baselines are evaluated on 59 datasets from the UCR Machine Learning repository [22] that are available on UCR [1] website. The datasets originate from different sources and domains such as human motion data, sensor data, simulated, and medial sequences. The first columns of Table I presents the metadata of the datasets used in this study (dataset name, dataset size, time series length, and the number of classes). The fourth column of the table ($\mathscr{K}\%$) shows the optimal sparsity percentage

[1]https://www.cs.ucr.edu/~eamonn/time_series_data/

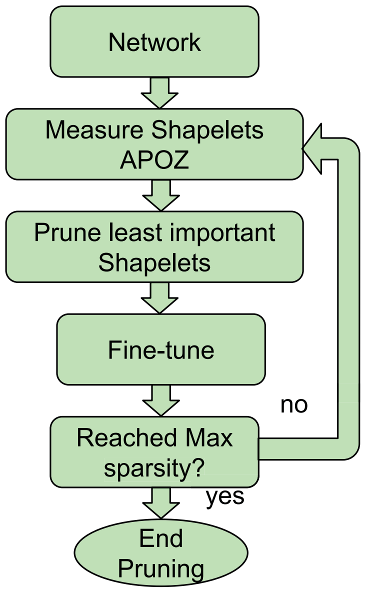


Fig. 5. Model Pruning protocol

level that we experimentally found for each dataset. We used the same default train and test split that was provided for comparability purposes. We performed z-normalization of the train and test sets prior to classification as shown in eq. 4.

$$Z_{norm} = \frac{x - \hat{x}_{train}}{\sigma_{train}} \tag{4}$$

where $\hat{x}_{train}$ and σ_{train} are the mean and standard deviation of the training set respectively.

B. Parameters Setting

In this section, we will specify the user-defined input parameters of the algorithm. The learning rate of the algorithm has been set to a default value of $\alpha = 0.001$ and the number of epochs used to train 1DCNN variants is $ep = 3000$ to avoid premature training leading to underfitting.

We used a mini-batch size of n_m=64, and the kernel lengths are set to $\{30\%,50\%,70\%\}$ of the time series length in the database. The percentage pruning at the end of each training cycle was set to $r = 5\%$ the total percentage of pruning was set to 95%. In other terms, for every time series database, we tested 20 models including the original model to find the optimal network.

In addition, in order to speed up the learning process, we used batch normalization [18] of the kernels before applying ReLU as the activation function for non-linearity of our model and a categorical cross-entropy cost function as defined in Eq. 5 (labels are provided in one-hot representation). We used the softmax function to map the non-normalized output of the model to a probability distribution over the number of classes the time series database contains. We trained our network with Adam optimizer with default parameters (β_1=0.9, β_2=0.999 and ϵ=1e-8). We also imposed a learning rate reduction once the learning plateaus and there is no improvement for

TABLE I
UCR Datasets Metadata and optimal K% sparsity

ID	Dataset Name	$\mathscr{L}$	$\mathscr{K}$ %	DS Size	TS Length
1	Adiac	37	35	781	176
2	ArrowHead	3	40	211	251
3	Beef	5	75	60	470
4	BeetleFly	2	80	40	512
5	BirdChicken	2	30	40	512
6	Car	4	75	120	577
7	ChlorineConcentration	3	70	4307	166
8	Coffee	2	95	56	286
9	Computers	2	55	500	720
10	Cricket_X	12	60	780	300
11	Cricket_Y	12	55	780	300
12	Cricket_Z	12	40	780	300
13	DiatomSizeReduction	4	30	322	345
14	DPOutlineAgeGroup	3	95	539	80
15	DPOutlineCorrect	2	50	876	80
16	ECG200	2	70	200	96
17	ECGFiveDays	2	80	884	136
18	FaceAll	14	85	2250	131
19	FaceFour	4	75	112	350
20	FacesUCR	14	65	2250	131
21	50words	50	65	905	270
22	FISH	7	75	350	463
23	Gun_Point	2	60	200	150
24	Haptics	5	25	463	1092
25	Herring	2	60	128	512
26	InlineSkate	7	90	650	1882
27	InsectWingbeatSound	11	65	2200	256
28	Lighting2	2	65	121	637
29	Lighting7	7	45	143	319
30	MedicalImages	10	80	1141	99
31	MPOutlineAgeGroup	3	90	554	80
32	MoteStrain	2	40	1272	84
33	NIFatalECG_Thorax1	42	65	3765	750
34	NIFatalECG_Thorax2	42	85	3765	750
35	OliveOil	4	45	60	570
36	OSULeaf	6	85	442	427
37	Plane	7	80	210	144
38	PPOutlineCorrect	2	85	891	80
39	PPTW	6	85	605	80
40	ScreenType	3	90	750	720
41	ShapesAll	60	50	1200	512
42	SonyAIBORSurface	2	75	621	70
43	SonyAIBORSurfaceII	2	55	980	65
44	SwedishLeaf	15	80	1125	128
45	Symbols	6	85	1020	398
46	synthetic_control	6	75	600	60
47	ToeSegmentation1	2	45	268	277
48	ToeSegmentation2	2	40	166	343
49	Trace	4	95	200	275
50	TwoLeadECG	2	95	1162	82
51	Two_Patterns	4	70	5000	128
52	WGLibraryAll	8	70	4478	945
53	WGLibrary_X	8	65	4478	315
54	WGLibrary_Y	8	40	4478	315
55	WGLibrary_Z	8	80	4478	315
56	wafer	2	95	7164	152
57	WordsSynonyms	25	50	905	270
58	Worms	5	35	258	900
59	yoga	2	70	3300	426

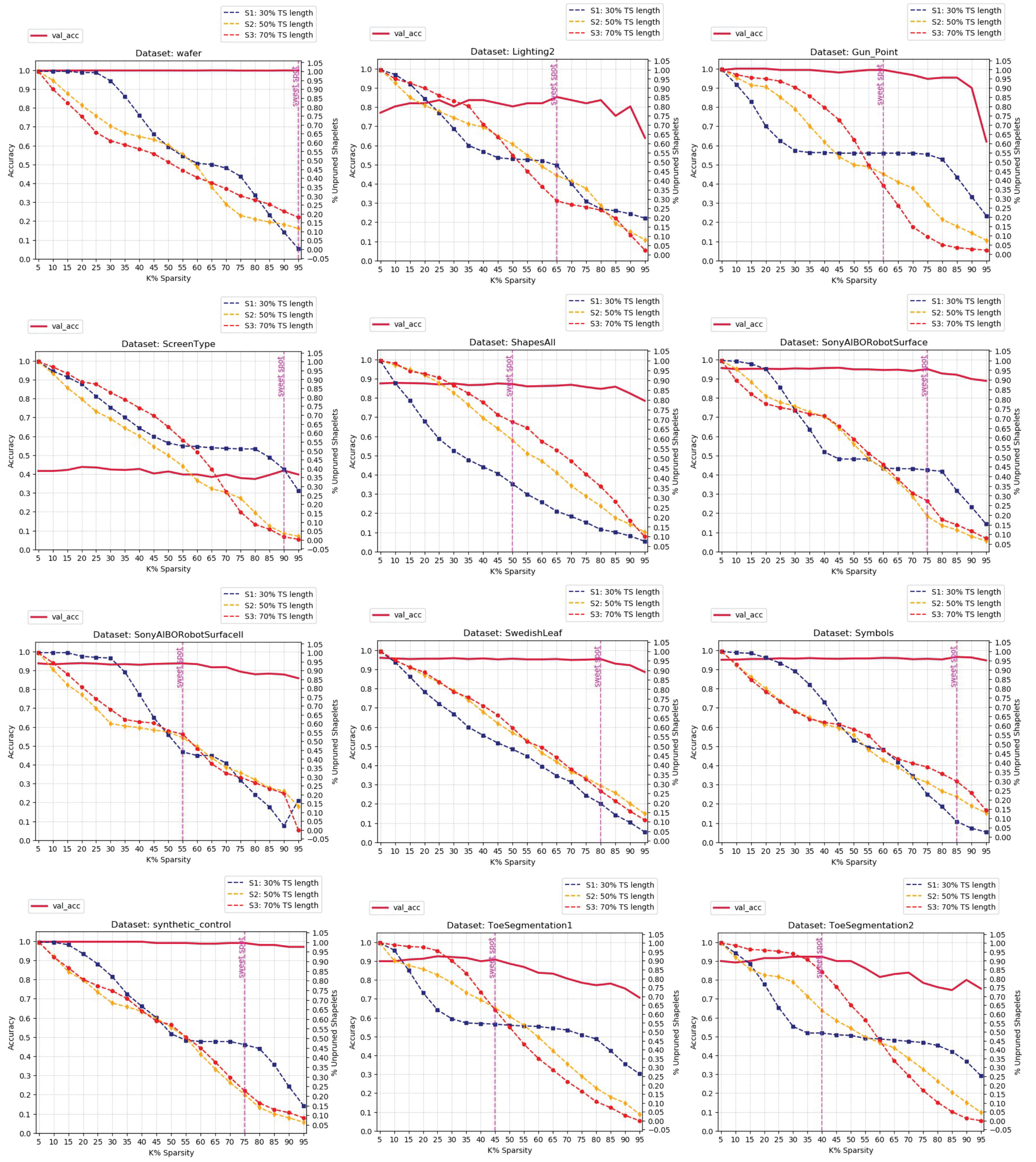


Fig. 6. Validation Accuracy with respect to $K\%$ Sparsity and sparsity curves for the three convolutional blocks corresponding to the three shapelets lengths for : wafer, Lighting2, Gun_Point, ScreenType, ShapesAll, SonyAIBORobotSurface, SonyAIBORobotSurfaceII, SwedishLeaf, Symbols, synthetic_control, ToeSegmentation1, and ToeSegmentation2.

5 consecutive epochs. Since random initialization is doing poorly leading to networks converging more slowly and towards ultimately poorer local minima, we tried to warm the initialization of the network weights and convolutional block kernels, we used Xavier initializer [23]. More details about the model are shown in Table II.

$$\mathscr{L}(y,\hat{y}) = -y_j \sum_{j=1}^{N} log(\hat{y_j})[y_i \in C_i] \tag{5}$$

We used Tensorflow framework [24] coupled with Keras library [25] to implement our proposed network.

VI. Experimental Results

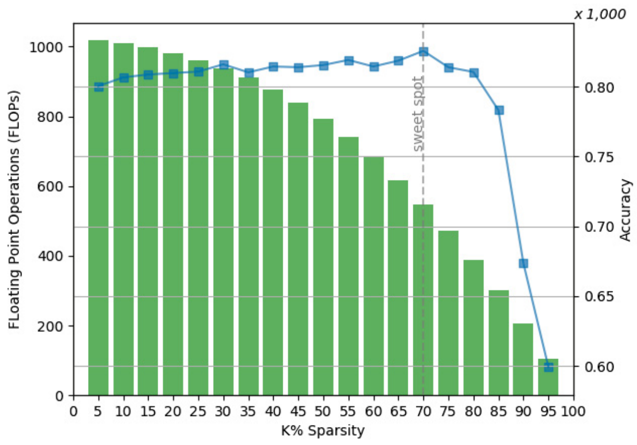
The contribution of the proposed model is its potential use for (1) prediction-based and (2) shapelet-based querying. In this section, we will discuss two types of experiments that we conducted. The first category aims to quantitatively show the efficiency and robustness of the algorithm for prediction-based querying purposes. The second category of experiments aims to qualitatively show the superiority of the model in mining shapelets for the sake of shapelet-based querying. As discussed in section 4, mining accurate shapelets that are truly modeling one subset class of the time series database is achieved through iterative pruning and fine-tuning of the network.

TABLE II
Original and pruned 1DCNN networks parameter. Our model uses the following values: i=30, j=50, k=70. The weight initializations are Glorot Normal and the activation function is ReLu.

Network	**1DCNN**	**1DCNN_$K\%pruned$**
Conv Layers	0.3 * TS length, pool	0.3 * TS length, pool
	0.5 * TS length, pool	0.5 * TS length, pool
	0.7 * TS length, pool	0.7 * TS length, pool
FC Layers	TS length, 300	TS length, i% * 300
	TS length, 300	TS length, j% * 300
	TS length, 300	TS length, k% * 300
Iterations	3,000	3.000
Batch	64	64
Optimizer	Adam	Adam

One of the prominent measures of success of the pruning is the declining average percentage of zero activation from one pruning iteration to the next iteration. APoZ is a relative measure indirectly proportional to the number of remaining unpruned shapelets in the model, as shown in Equations 2 and 3. Since it is a normalized measure, it is fair to compare the values of APoZ from two different models. Figure 8 shows the distribution of the APoZ values of models from the two extremes, the original unpruned model (in orange) and a model pruned with the maximal percentage of pruned shapelets 95% (in purple). The first observation that can be made is that the mean of the distribution of the pruned model is more than three standard deviations away from the mean of the original unpruned model. This suggests that there is a statistically significant difference between the two means. The shift of the pruned distribution to the bottom values is indicative of the significant drop in the network redundancy. Also, the diminishing width (standard deviation) of the distribution was diminished from the original unpruned to the pruned APoZ distributions which are indicative that the weak neurons are vanishing, a proof of the benefit gained from weight initialization as discussed in section IV. The second observation that can be made is that the original unpruned model follows a bimodal distribution having the first mode peak at 0.63 and another mode at 0.45. This suggests that the at the end of the training of the original model there are two families of shapelets, the ones with high contribution to the network decision and the ones with low support. in other terms, there still exists room of improvement of the original network based on the wide and binomial nature of the distribution of its APoZ values. The 95%-sparsely pruned network follows a balanced Gaussian distribution with a clear mode of 0.41. Simply put, more than half of the shapelets are significantly contributing to the model by having non-zero activations.

It is important to understand that an optimally pruned model has a good balance of simplicity (fewer parameters) and good accuracy levels, a crucial pre-requisite for precise querying. Figure 6 shows the learning curves of a subset of twelve datasets. In this context, the model complexity represents the sparsity level (K) of the 1DCNN algorithm. Ideally, a good K parameter is the one where the model sparsity is maximized without compromising accuracy. The vertical line in Figure 6 shows a good cut-off point for the learning curves. Upon the completion of all experiments, we noticed that there are three categories of learning curves. The first category of learning curves is characterized by a model that is equally accurate regardless of its sparsity level. The wafer dataset is a good example of this pattern. Maintaining the same accuracy level of the original unpruned model with reduced complexity is the desired outcome since the model complexity is drastically reduced, making the model more robust and with better ability to generalize and produce more accurate shapelets. The second family of learning curves shows a pattern of accuracy improvement as the network is sparsified. The lightining2 dataset shows a maximum improvement of 6% from the original network accuracy at sparsity level 65%. This category of datasets is benefiting from the network sparsity by reaching unprecedented accuracy levels. This behavior suggests that unimportant shapelets were not simply ignored (by zeroing out their activations) but rather hindering the maximal learning potential of the model. Finally, the last learning curve behavior, which is the most intuitive one, is accuracy levels having a downward trend with increasing sparsity (model simplicity). Gun_Point dataset shows an example of this behavior where the accuracy starts dropping starting from the $\mathscr{K}$=60% mark where the model starts deteriorating and indicating it is too simple to accurately solve the query. Figure 6 also shows the decreasing percentage of shapelets from each of the three-length categories as the model is being

a. 

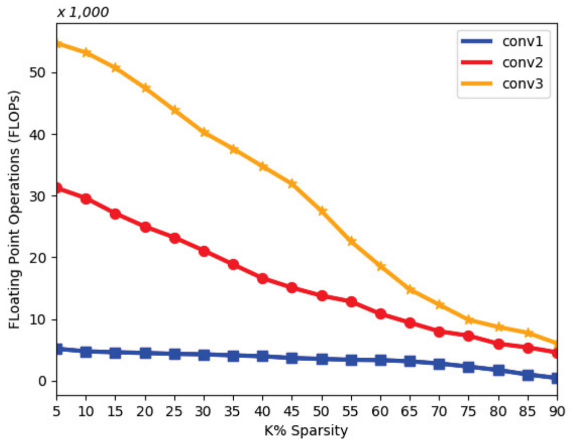
b. 

Fig. 7. (a) Floating Points Operations per Second (FLOPs) with respect to network sparsity and (b) Floating Point Operations with the three shapelets lengths categories with respect to the network sparsity

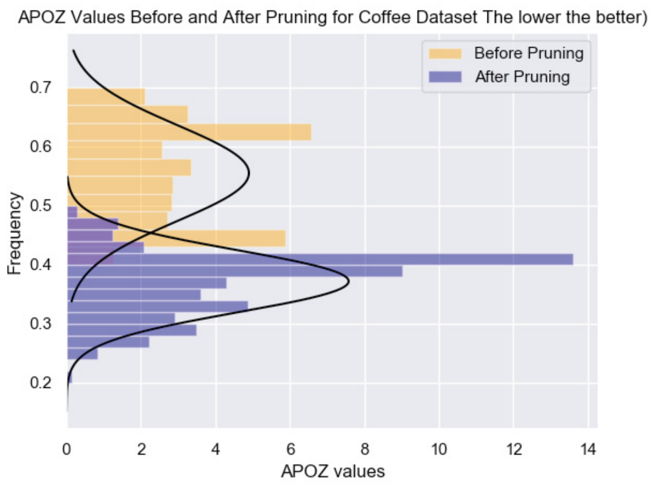


Fig. 8. APOZ values before and after pruning for the Coffee Dataset

pruned. It is clear that there is no shapelet category length that is superior over another category. All three lengths of shapelets are grasping important discriminative patterns, although, the shortest shapelet lengths (in blue) show more resistance to the pruning in the range of $\mathscr{K}$=[40,60] for the three datasets. This could be explained that the shortest shapelets are more prone to model highly granular noise that is falsely deemed as important. The fourth column of Table I ($\mathscr{K}$) shows the optimal sparsity percentage level corresponding to the vertical sweet spot line in Figure 6. The table does not show any particular correlations between the number of classes, dataset size, time series length with the optimal sparsity level. This means that network pruning is a data-driven process and there is no size fits all $\mathscr{K}$ parameter.

Another common way of quantifying the complexity of the neural network is to measure the floating-point operations (addition, subtraction, multiplication, or division) per second (FLoPs) required by the model. A prominent benefit of network pruning is the reduction of the number of convolutional feature maps and as therefore the total estimated floating-point operations (FLoPs). Figure 7-a shows the FLoPs for each model with different sparsity levels and their corresponding accuracy. The optimal $\mathscr{K}$ in this dataset (ChlorineConcentration) has been identified as 70%. Having a 70% pruned model saves almost half ($\approx$ 42%) of the total operations required by the original unpruned model. It is also worth mentioning that the FLoPs are highly dependent on the shapelet lengths that exist in the network. In the case where longer lengths shapelets are pruned on a higher rate than other smaller lengths shapelets, the FLoPs are reduced with a higher rate as well. Figure 7-b illustrates the FLoPs operations required by each convolutional layer as the network is being sparsified. To further illustrate the idea, we can clearly see from Figure 7-b that the first convolutional block has a smaller negative slope compared to the second and third convolutional layers. The linear scale of the figure realistically illustrates the rate of change for each shapelet's length.

We compared our proposed 1DCNN models, pruned with the optimal sparsity levels found experimentally, to all the three baseline models pertaining to the shapelets classifiers category: Fast Shapelets[16], Learning Shapelets [17], and Shapelet Transforms [15]. The blue area of figure 9 illustrates the area where our model is outperforming the baseline counterpart. Every point in the figure represents one of the datasets defined in Table I. The more points are in the diagonal, the less significant is the difference in performance between the two baselines. Considering the total number of datasets, our model is more accurate than all the three baselines. It is particularly outperforming FS and LS baselines with a (54 wins/5 losses) and (35 wins/24 losses) respectively. Furthermore, we can see that the points in the blue area are more scattered and far from the diagonal which suggests that there is an important magnitude difference. It is important to note that in addition to the fact that 1DCNN is more accurate than all the other

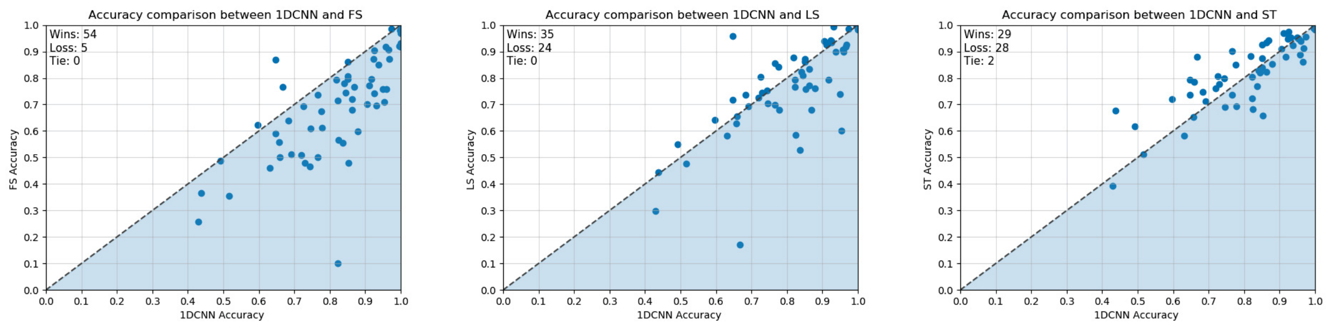


Fig. 9. Accuracy results obtained from pruned 1DCNN classifier with optimal $\mathscr{K}$ parameter, in comparison with: FS, LS and ST classifiers.

learners in the shapelets, it produces significantly less amount of shapelets than its counterparts, which is the goal of the model.

The second byproduct of our model in the mined shapelets that are the basis of shapelet-based querying. So far, we have assessed the performance of the 1DCNN model based solely quantitatively through FLoP, accuracy, and APOZ measures. To ensure that the mined shapelets are equally precise, we conducted another experiment to quantify the degree of closeness of the mined shapelet pattern to the original time series instances in the database. To do so we used Dynamic Time Warping to measure the shape differences.

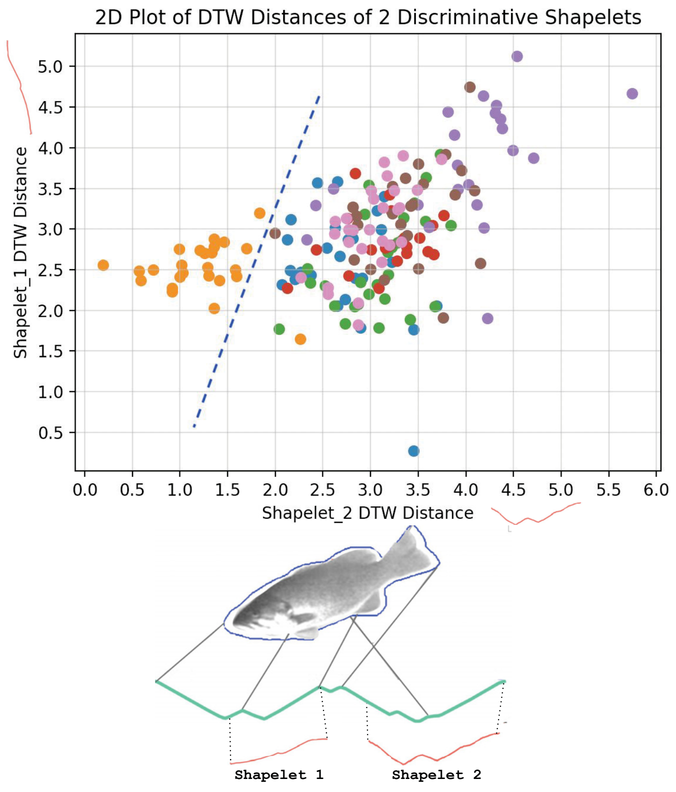


Fig. 10. Dynamic Time Warping Distances of the two most discriminative shapelets for each class with respect to all instances.

Dynamic time warping (DTW) is a standard elastic measure for assessing the dissimilarity between two time series [26]. Due to its effectiveness in finding an optimal match between two sequences, DTW has been used in many different domains such as shape interpolation [2] [27] and time series matching for incomplete medical data [28]. DTW works by warping the time series in the time domain in such a way that the final warping cost is minimal. The canonical form of DTW is shown in Equation 6. M and N represent the lengths of the input time series x and y. Initially the D matrix is initialized to $D_{0,0} = 0$ and $D_{i,j}$ to $D_{i,j} = \inf$. The cost function f in Eq. 6 is usually chosen to be the square of the differences between the time series x_i and y_j.

$$D_{i,j} = f(x_i, y_j) + min\{D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}\} \\ s.t: i \in (1, M), j \in (1, N) \tag{6}$$

In this context, we used DTW to compute the distance between the mined shapelets and all the instances in the time series database. This step is useful to show the mined shapelet relative closeness to a subset of classes and distantness to another subset of classes. For the sake of simplicity, we showed an example of output from the FISH dataset that has 7 classes. We used the pruned model at optimal 75% sparsity level and the APOZ heuristic to sort the mined shapelets in the model and selected the two most discriminative shapelets for 2 distinct classes. Figure 10 shows a plot of distances between the two most significant shapelets. The dashed line shows the separation between the orange class instances and the rest of the database instances. Figure 11 shows the two most discriminative shapelets overlaid on top of their optimal positions on instances belonging to two different classes of the Gun_Point dataset. Visually, the red shapelet corresponds to a pattern of the first class of the dataset. Similarly, the purple shapelet corresponds a pattern of the second class of the problem. The visual observations are in line with the DTW distances shown in the figure (the distance is smaller to the class the shapelet represents).

VII. Conclusion

In this work, we proposed a new shapelet mining learner, 1DCNN, that leverage the use of convolutional neural network from the image processing field to learn interpretable

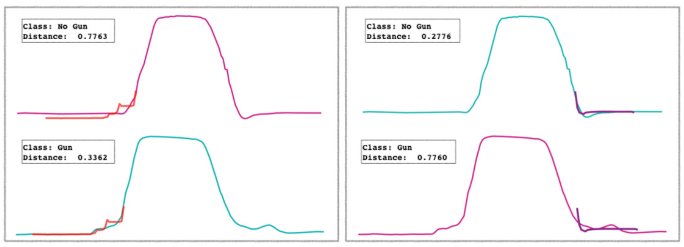


Fig. 11. Most Discriminative Shapelets for Gun_Point dataset binary classes

shapelets of different lengths. This is the first attempt to mine the minimal set of shapelets needed for an optimal model accuracy. Our model optimizes the entire classification schema by learning the shapes of the representative patterns and then use a pruning mechanism that iteratively trims unimportant shapelets from the model, leaving only a fraction of shapelets without compromising prediction accuracy. We have shown experimentally that our method is superior to all other baseline models pertaining to the shapelet category group. In addition, our pruned models are more compact and robust delivering highly accurate predictions with up to 95% less Floating Points Operations per Second (FLOPs) required by the network. Now that we omited the need of number of shapelets as a user-defined parameter, a natural future work direction that we would like to address is the automatic parameter setting for shapelet lengths.

ACKNOWLEDGMENT

This project has been supported in part by funding from the funding from Division of Advanced Cyberinfrastructure within the Directorate for Computer and Information Science and Engineering, the Division of Astronomical Sciences within the Directorate for Mathematical and Physical Sciences, and the Division of Atmospheric and Geospace Sciences within the Directorate for Geosciences, under NSF award 1931555. It was also supported in part by funding from the Heliophysics Living With a Star Science Program, under NASA award NNX15AF39G.

REFERENCES

[1] S. F. Boubrahimi, B. Aydin, P. Martens, and R. Angryk, “On the prediction of >100 MeV solar energetic particle events using GOES satellite data,” in *Big Data*. IEEE, 2017.

[2] S. F. Boubrahimi, B. Aydin, M. A. Schuh, D. Kempton, R. A. Angryk, and R. Ma, “Spatiotemporal interpolation methods for solar event trajectories,” *APJs*, vol. 236, no. 1, p. 23, 2018.

[3] S. F. Boubrahimi, B. Aydin, D. Kempton, S. S. Mahajan, and R. Angryk, “Filling the gaps in solar big data: Interpolation of solar filament event instances,” in *BDCloud 2016*. IEEE, 2016, pp. 97–104.

[4] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, “Time-series classification with cote: the collective of transformation-based ensembles,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2522–2535, 2015.

[5] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series.” in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[6] P. Schäfer, “The boss is concerned with time series classification in the presence of noise,” *Data Mining and Knowledge Discovery*, vol. 29, no. 6, pp. 1505–1530, 2015.

[7] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[8] Z. Wang, W. Yan, and T. Oates, “Time series classification from scratch with deep neural networks: A strong baseline,” in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 1578–1585.

[9] S. F. Boubrahimi, R. Ma, B. Aydin, S. M. Hamdi, and R. Angryk, “Scalable knn search approximation for time series data,” in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 970–975.

[10] M. G. Baydogan, G. Runger, and E. Tuv, “A bag-of-features framework to classify time series,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2796–2802, 2013.

[11] M. G. Baydogan and G. Runger, “Time series representation and similarity based on local autopatterns,” *Data Mining and Knowledge Discovery*, vol. 30, no. 2, pp. 476–509, 2016.

[12] J. Lines, S. Taylor, and A. Bagnall, “Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 5, p. 52, 2018.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[14] L. Ye and E. Keogh, “Time series shapelets: a new primitive for data mining,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 947–956.

[15] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, “A shapelet transform for time series classification,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 289–297.

[16] T. Rakthanmanon and E. Keogh, “Fast shapelets: A scalable algorithm for discovering time series shapelets,” in *proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 668–676.

[17] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, “Learning time-series shapelets,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 392–401.

[18] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

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

[20] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[21] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, “Network trimming: A data-driven neuron pruning approach towards efficient deep architectures,” *arXiv preprint arXiv:1607.03250*, 2016.

[22] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, “The ucr time series archive,” *arXiv preprint arXiv:1810.07758*, 2018.

[23] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[25] A. Gulli and S. Pal, *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[26] E. Keogh and C. A. Ratanamahatana, “Exact indexing of dynamic time warping,” *KIS*, vol. 7, no. 3, pp. 358–386, 2005.

[27] S. F. Boubrahimi, B. Aydin, D. Kempton, and R. Angryk, “Spatio-temporal interpolation methods for solar events metadata,” in *IEEE Big Data 2016*. IEEE, 2016, pp. 3149–3157.

[28] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli, “Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation,” *Artificial intelligence in medicine*, vol. 45, no. 1, pp. 11–34, 2009.