Master's Thesis

Perceptual and Position-aware Shapelet Networks for Time Series Classification

LE Thi Xuan May

Supervisor: Prof. HUYNH Van Nam

School of Knowledge Science
Japan Advanced Institute of Science and Technology
(Master of Knowledge Science)

June, 2023

Abstract

Shapelets are time series segments that effectively distinguish labels of time series. Recently, shapelet-based time series classifiers have garnered attention from the academic community, primarily due to their ability to produce accurate results while also being interpretable. However, these methods still confront challenges in both the shapelet initialization and shapelet learning phases, which can hinder their performance. In this dissertation, we propose two novel methods to effectively address these problems.

In our first work, we propose the Perceptual Position-aware Shapelet Network (PPSN), a novel shapelet-based classifier, to discover and optimize shapelets efficiently. PPSN leverages perceptually important points for extracting a limited number of high-quality shapelet candidates. These candidates are then evaluated using position-aware subsequence distance. In addition, we introduce Fixed Normalization (FN) and Stop-gradient Epochs (SGE) as two novel techniques for learning shapelets. Specifically, FN addresses the detrimental effects of different value ranges of shapelets' transformed values, while SGE mitigates the negative effects of non-optimal weights in the final linear layer. Results from experiments on 112 UCR datasets illustrate that our PPSN surpasses previous non-ensemble methods, and is competitive with HIVE-COTE 2.0, the current most accurate classifier while maintaining the benefits of interpretability and efficient computation time.

In our second work, we point out two other issues faced by existing shapelet-based methods. Firstly, they employ the soft-minimum function which only retrieves the minimum distance between a shapelet and its best-matching subsequences in the time series, neglecting other distances. Additionally, this function is prone to generating overflow values, which can result in the model not functioning properly in various scenarios. Secondly, previous methods use a General Classification Head to train shapelets, which is not optimized to capture the specific patterns in time series data. To address these problems, we propose PPSN++, a stronger shapelet-based time series classifier that extends our previous work PPSN. PPSN++ uses a Distance Learning Layer instead of the soft-minimum function, we also introduce a Binary Auxiliary Head which supports General Classification Head in training the shapelet for each specific class. Results from experiments on 112 UCR datasets show that PPSN++ outperforms PPSN, making it become the state-of-the-art shapelet-based method for this task.

Keywords: time series · classification · shapelet discovery · efficiency.

Acknowledgments

From the bottom of my heart, I would like to express my gratitude to all those who have contributed to the successful completion of this Master's dissertation.

First and foremost, I wish to thank my supervisor, Prof. HUYNH Van Nam, from Japan Advanced Institute of Science and Technology (JAIST), for providing invaluable guidance and support throughout the entire research process. Your insights and expertise have been a constant source of inspiration to me.

I am also deeply grateful to the faculty and staff of School of Knowledge Science, for creating a stimulating and intellectually rigorous academic environment that challenged me to strive for excellence. I would like to acknowledge the members of my examination committee, Prof. HASHIMOTO Takashi, Prof. YUIZONO Takaya, Prof. GOKON Hideomi, and my second supervisor Prof. DAM Hieu Chi, for their constructive feedback and insightful suggestions that helped me refine my research questions and methodology.

I wish to extend my appreciation to my family for their unwavering support and encouragement throughout my academic journey. Your love and encouragement gave me the strength and motivation to overcome the challenges that I faced, and I am grateful for your sacrifices and belief in my abilities.

Finally, I would like to thank all my friends and fellow members of the Huynh lab without whom this research would not have been possible. Your enthusiasm, patience, and willingness to share your experiences inspired me to surpass my own expectations.

Once again, I express my sincere gratitude to everyone who has contributed to this dissertation, and I hope that this research will serve as a meaningful contribution to the academic community.

Thank you,
LE Thi Xuan May
JAIST, May 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Time series data is a crucial component of Industry 4.0, which is the ongoing digital transformation of the manufacturing industry. In Industry 4.0, sensors and other monitoring devices are used to collect substantial volumes of data from machines, production lines, and other equipment. This data is then analyzed to identify patterns and insights that can be used to improve efficiency, quality, and productivity. Time series data is particularly important in this context, as it allows manufacturers to monitor the performance of their equipment over time and detect deviations from normal operating conditions. By using advanced analytics and machine learning techniques, manufacturers can gain a deeper understanding of their operations and make data-driven decisions to optimize their processes. Time series analysis has thus become a critical tool in Industry 4.0, enabling manufacturers to improve their operations, reduce costs, and deliver better products and services to their customers. Theoretically, a time series is a collection of observations of well-defined data objects gathered over time using repeated measurement. It has relevance to several fields not only manufacturing systems [10], but also biology [3], medicine [13], and economic [24]. A classic example of a time series is the number of heartbeats per minute measured by an electrocardiogram (ECG).

Time series classification (TSC) has attracted considerable interest from the industrial and research community due to its crucial role in numerous practical applications. Several algorithms for TSC have been developed in recent decades, with ensemble-based [18, 26], feature-based [7, 6], and shapelet-based techniques [17, 20, 19] considered as the current state-of-the-art. Among these, shapelet-based methods are distinguished by their potential to yield more comprehensible decisions compared to other techniques, as they utilize critical local patterns (known as shapelets) extracted from the primary time series to determine their class. Intuitively, shapelets are

defined as subsequences of time series that accurately classify time series labels. This approach enables greater interpretability by providing an intuitive representation of the underlying structure of the time series data.

However, previous shapelet-based methods [27, 17] pose certain issues in both the phases of initializing shapelets and learning shapelets, including:

- **Problem 1:** For the shapelet initialization phase, they usually use the Full Extractor to extract all potential shapelet candidates and then used the Euclidean Distance method to evaluate the distance between these candidates and the target time series, resulting in an excellent performance, but bearing high level of complexity in computing. To overcome this issue, the Fixed-Length Extractor [11] was proposed, which only extracted candidates of the same length and clustered them using k-Means. It considers k-Means centroids as the initial shapelets, nevertheless, this method assumes a fixed shapelet length, even though the dataset may contain shapelets of various lengths. Recent attempts to speed up the shapelet extractor process and optimize parameters automatically have used piecewise aggregate approximation (PAA) [15, 23, 28], but the utilization of this technique carries the risk of missing certain data characteristics.

- **Problem 2:** For the learning shapelet phase, conventional techniques have relied on direct learning from subsequence distances, which poses challenges in training and convergence owing to some shapelets providing extremely high values and others significantly smaller values. Additionally, the linear layer in the final network typically produces unsatisfactory predictions in the first training epochs due to non-optimal weights. These challenges may reduce the overall performance of the model.

- **Problem 3:** Prior methods based on shapelets employ the soft-minimum function, which solely obtains the minimum distance between a shapelet and its best-matching subsequences in the time series, disregarding other distances. Moreover, this function has a tendency to generate overflow values, thereby making the model dysfunctional in various situations.

- **Problem 4:** Existing shapelet-based classifiers utilize a General Classification Head for training shapelets, which is not designed to effectively identify the particular patterns that are inherent in time series data.

In our first work, we propose a novel shapelet-based method for TSC, namely Perceptually and Position-aware Shapelet Network (PPSN, Chapter

3) to tackle two first problems. Specifically, PPSN efficiently extracts a relatively small number of high-quality shapelets with various lengths by leveraging the perceptually important points, next these candidates will be evaluated by utilizing the position-aware subsequence distance (**Problem 1**). After that, these selected shapelets are used to transform the original time series dataset for the ultimate purpose of classifying time series. In addition, we propose to use two novel techniques, namely Fixed Normalization (FN) and Stop-gradient Epoch (SGP) to overcome the mentioned problems in the shapelet learning phase. In particular, FN aims to tackle the detrimental impact of various value ranges of transformed values, while SGP is used in the initial few epochs of the process to lessen the undesirable effects of the final linear layer's suboptimal weights. (**Problem 2**). The results demonstrate that our PPSN achieves better performance and reduces the computation time.

In our second work, we aim to propose a stronger shapelet-based classifier, which is designed based on the PPSN framework (called PPSN++, Chapter 4). Our PPSN++ involves the utilization of a Distance Learning Layer (DLL) in lieu of the soft-minimum function (**Problem 3**), as well as the incorporation of a Binary Auxiliary Head (BACH) to support the General Classification Head in training shapelets for individual classes (**Problem 4**). Through these enhancements, PPSN++ effectively addresses the two last-mentioned issues and demonstrates superior performance in time series classification. Empirical results on 112 UCR datasets indicate that PPSN++ outperforms existing methods (including our first work, PPSN), and achieves state-of-the-art results in this task.

The rest of this dissertation is organized as follows. Chapter 2 provides background and related work. Chapter 3 presents our novel Perceptual and Position-aware Shapelet Network for TSC (PPSN), followed by Chapter 4, we build a stronger shapelet-based classifier upon PPSN framework (PPSN++). In Chapter 5, we conclude this work and draw out some potential directions for future work.

Chapter 2

Background and Related Works

2.1 State-Of-The-Art Time Series Classifiers

Time series classification (TSC) is the process of analyzing data that changes over time to classify it into different classes. This is achieved by identifying patterns or features within the data. Over the past few decades, several algorithms have been proposed for Time Series Classification (TSC). At present, ensemble-based and feature-based approaches are the most advanced. The state-of-art algorithms (SOTA) for TSC currently include HIVE-COTE [18] and its variations [22], InceptionTime [14], TS-CHIEF [26], ROCKET [6], and its improved version (MiniROCKET [7]).

SOTA ensemble-based TSC methods, including:

- InceptionTime was first introduced by Fawaz et al. [14], which is currently the most effective deep learning model for TSC. This architecture consists of five convolutional neural networks based on Inception. By combining these networks, the variance of the model is reduced, resulting in a significantly more accurate method than other TSC techniques based on deep learning, such as the Fully Convolutional Network (FCN) and Residual Network (ResNet).

- TS-CHIEF is a scalable Time Series Classification (TSC) algorithm proposed by Shifaz et al. [26] that achieves competitive accuracy with HIVE-COTE [18]. TS-CHIEF is based on the Proximity Forest, which is an ensemble of decision trees that employ distance measurements as the splitting criterion at each node. By combining interval and spectrum based splitting criteria, TS-CHIEF enhances Proximity Forest and allows the ensemble in order to capture a more diverse range of

representations.

- HIVE-COTE [18], a meta-ensemble classifier for time series, that includes the most accurate ensemble classifiers from different domains of time series representation. The original HIVE-COTE comprises five ensemble classifiers: Ensemble of Elastic Distances (EE) [16], Shapelet Transform Classifier (STC) [12], Bag of SFA Symbols (BOSS) Ensemble [25], Time Series Forest (TSF) [9], and Random Interval Forest (RIF) [18], each of which is the most accurate classifier in its respective domain. Therefore, HIVE-COTE is much more accurate than each of its component parts, thus it has served as a high standard for classification accuracy.

- Lately, a novel model for classifying time series data has been proposed known as HIVE-COTE 2.0 [22], which was developed by Middlehurst et al. [22]. This model has demonstrated superior average accuracy rankings when compared to various state-of-the-art techniques across both the univariate UCR archive [5] and the multivariate UEA archive [1]. HIVE-COTE 2.0 is constructed as a meta-ensemble of four primary components, including STC [12], Arsenal, Temporal Dictionary Ensemble (TDE) [21], and Diverse Representation Canonical Interval Forest (DrCIF) [22].

SOTA feature-based TSC methods, including:

- ROCKET was first introduced by Dempster et al. [6]. This method involves the transformation of time series data using a multitude of convolutional kernels that are characterized by random attributes such as length, weights, bias, dilation, and padding. The resulting transformed features are subsequently utilized to train a linear classifier (using ridge regression for all, with the exception being the largest datasets, for which logistic regression is also utilized).

- MiniRocket [7] is an enhanced model of the ROCKET that makes a few modifications to its kernels and significantly improves the convolutional process. Both ROCKET and MiniRocket use convolutional kernels to transform the original input series. MiniRocket employs a predetermined set of 84 kernels and creates several dilations and biases for each kernel. As a default, this process generates 10,000 features for the convolution operations.

In general, with the exception of ROCKET and MiniROCKET, most SOTA methods for TSC suffer from considerable computational complexity. While ROCKET and its variant lack interpretative power.

2.2 Shapelet-based Time Series Classifiers

Shapelet-based classifiers [27, 17] is another type of feature-based method. Nevertheless, they can produce more interpretable decisions since they rely on critical local patterns (shapelets) extracted from the primary time series to determine their class.

Two initial classifiers explore all feasible shapelet candidates in a given dataset, subsequently choosing the final shapelets according to their information gain. Following this, a shapelet decision tree was constructed, whereby the optimal shapelet was assigned to each of its nodes [27]. Alternatively, another approach involved transforming the target time series based on their distance to shapelets and then applying a range of standard techniques (such as SVM, Decision Tree,...) to classify the transformed values in lieu of the original time series [17]. Recent research has incorporated both shapelets and learning algorithms, resulting in the direct training of shapelets capable of identifying time series belonging to different classes [11, 19, 20]. Notably, several methods have made notable contributions to this task, including:

- Learning Time Series Shapelets (LTS) was first proposed by Grabocka et al. [11] in 2004, it presents a novel approach to learning shapelets for time series classification tasks. The proposed method involves iteratively selecting discriminative subsequences from a dataset of time series. The most informative shapelets are then selected and used to transform the original time series into a more discriminative representation, which is subsequently used to train a classifier on the transformed data.

- Triple-shapelet Network (TSN) [19] is a novel approach to learning discriminative features from time series data by extracting three shapelets (namely, Category-specific Shapelet, General Shapelet, Sample-specific Shapelets) per class and combining them into a single representation. This effectively captures the distinctive patterns within each class of time series, leading to improved accuracy and robustness to noise and outliers.

- Adversarial Dynamic Shapelet Network (ADSN), proposed by Ma et al. [20] in 2020, the authors introduce the concept of dynamic shapelets. ADSN first uses the convolutional neural network (CNN) to extract the shapelet (Shapelet Generator). This method also builds the Discriminator to determine whether the shapelet is generated from Shapelet Generator or actually comes from time series (by the traditional way).

From that, they can ensure the shapelet generated from Shapelet CNN Generator is similar to the real subsequence of time series in the dataset.

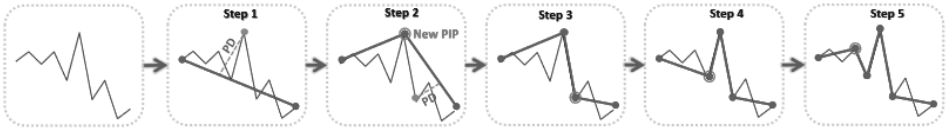


Figure 2.1: The process for obtaining the first six Perceptually Important Points (PIPs). In that, PD is calculated by Eq. 2.1.

Algorithm 1. PIPExtractor

Input:

- T: time series
- k: number of PIPs

Output:

- $\mathcal{P}^T$: set of PIPs in T

1: $\mathcal{P}^T = [1, n]$
2: **for** $i \in [1, k-2]$:
3: Find $pos \in [1, n]$ and $pos \notin \mathcal{P}^T$ with max $PD(T[pos], \mathcal{P}^T)$
4: $\mathcal{P}^T$.add(pos)
5: $\mathcal{P}^T$.sort()

2.3 Perceptually Important Points (PIPs)

The Perceptually Important Points (PIPs) method was initially introduced by Chung et al. [4] to extract salient points from a pricing series and has since been widely utilized in time series data mining for various duties, for example, expressing data in a certain way and decreasing the number of dimensions. Assume that $T = [t_1, ..., t_n]$ is a time series and k is the number of important points to extract. Initially, the first and the last index of T are added to a list of $\mathcal{P}^T$, which is represented as $\mathcal{P}^T = [1, n]$. Subsequently, a recursive process is employed to identify the index in T with the highest Perpendicular Distance from the line created from two previously added elements in $\mathcal{P}^T$. In that, Perpendicular Distance between one position, pos, and $\mathcal{P}^T$ is computed

by Eq. 2.1.

$$PD(pos, \mathcal{P}^T) = \frac{a * P_{pos} - T_{pos} + c}{\sqrt{a^2 + 1}} \tag{2.1}$$
$$a = \frac{T_e - T_s}{P_e - P_s}; \quad c = T_e - a * P_e$$

Given g where $1 \leq g \leq k$ and $\mathcal{P}_g^T < pos < \mathcal{P}_{g+1}^T$, assign $s = \mathcal{P}_g^T$ and $e = \mathcal{P}_{g+1}^T$, and P is list of position with z normalization, $P = z\_norm([1, .., n])$.

The complexity of detecting important points is $O(kn)$, however, detecting PIPs for each time series can be pre-computed before calculating the distance. An illustration of extracting the first six PIPs from the target time series is shown in Fig. 2.1. Algorithm 1 presents the pseudocode for extracting PIPs.

2.4 Preliminaries

This section aims to present all the essential definitions and notations.

Definition 1: *Time Series Dataset.* A collection of m time series is denoted as a time series dataset $\mathcal{M}$, where $\mathcal{M} = [T^1, \ldots, T^m]$. Each time series T^i has an associated label $y^i \in Y$. Furthermore, L represents the number of distinct classes in $\mathcal{M}$, from that $Y = [Y_i, ..., Y_L]$.

Definition 2: *Time Series.* A time series denoted by $T = [t_1, \ldots, t_n]$ of length n is a sequence of n real numbers obtained at regular intervals over a specified duration of time.

Definition 3: *Subsequence.* Given a time series $T = [t_1, ..., t_n]$, a subsequence $T_{i,i+l-1} = [t_i, ..., t_{i+l-1}]$ is a sequence of contiguous data points from the original series T, in which i presents the beginning point and l is the length of $T_{i,i+l-1}$ with $l \leq n$.

Definition 4: *Time Series Distance Measrure.* Time series distance measure is an essential function for calculating the similarity between two time series. In this thesis, we mention two existing popular distance measurements for time series and these distance measures are related to this work, including:

- Euclidean Distance (ED). Given two time series with the same length of n, $Q = [q_1, ..., q_n]$ and $C = [c_1, ..., c_n]$. The Euclidean Distance between Q and C is calculated by Eq. 2.2.

$$ED(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \tag{2.2}$$

- Complexity Invariant Distance (CID). That measure was introduced by Batista et al. [2]. The motivation of CID is based on the observation that complex time series are often more similar to simple time series than to other complex time series. CID of Q and C utilizing the complexity-invariant estimate, CI, is calculated as follow:

$$CID(Q,C) = ED(Q,C) * \frac{max(CI(Q), CI(C))}{min(CI(Q), CI(C))} \tag{2.3}$$

$$CI(Q) = \sqrt{\sum_{i=1}^{n-1}(q_{i+1} - q_i)^2}$$

Note that CI of all time series in the dataset can be pre-computed before calculating the distance, hence the complexity of CID can be the same as that of ED, $O(n)$.

Definition 5: *Subsequence Distance (SubDist).* Given time series T and a subsequence S with n and l are length of T and S respectively. Note that $l \leq n$, the subsequence distance, $SubDist$, of T and S is calculated as:

$$\text{SubDist}(T,S) = \min(\mathcal{DV}(T,S)) \tag{2.4}$$

where $\mathcal{DV}(T,S)$ is the distance vector which contains distances between S and all subsequence $T_{j,j+l-1} \in T, \forall j \in [1, n-l+1]$.

$$\mathcal{DV}(T,S) = [D_1, \ldots, D_{n-l+1}] \tag{2.5}$$

In the classic shapelet-based method, they leverage ED (Eq. 2.2) to calculate D_j. In this work, we leverage the Complexity-Invariant Distance for the calculation of SubDist. Therefore, its D_j is calculated as follows:

$$D_j = CID(T_{j,j+l-1}, S) \tag{2.6}$$

where, $CID(T_{j,j+l-1}, S)$ is calculated by Eq. 2.3.

Definition 6: *Soft-minimum Function.* However, this classic minimum function poses an important issue. Specifically, the minimum function of Eq.2.4 cannot be directly differentiated. This means that this minimum function is not a continuous function, which means that it does not have a derivative in the traditional sense since it has jump discontinuities. Therefore, it is proposed to exploit the soft-minimum function in order to pick out the minimum distance value, $SubDist$ can be calculated by the soft-minimum function as follows:

$$\text{SubDist}(T,S) = \text{SoftMin}(\mathcal{DV}(T,S)) = \frac{\sum_{j=1}^{n-l+1} D_j e^{\alpha D_j}}{\sum_{j=1}^{n-l+1} e^{\alpha D_j}} \tag{2.7}$$

when $\alpha \to \infty$, the soft-minimum approaches the true minimum.

Definition 7: *Information Gain (InfoGain).* Given a time series dataset $\mathcal{M}$ containing the labels A and B, with $p(A)$ and $p(B)$ denoting the respective percentages of instances belonging to each class, a split method, referred to as θ, is employed to partition $\mathcal{M}$ into two distinct subsets, namely, $\mathcal{M}_1$ and $\mathcal{M}_2$. As a result of this splitting, the following InfoGain is calculated:

$$\text{InfoGain}(\theta) = E(\mathcal{M}) - \left(\frac{|\mathcal{M}_1|}{|\mathcal{M}|} E(\mathcal{M}_1) + \frac{|\mathcal{M}_2|}{|\mathcal{M}|} E(\mathcal{M}_2) \right) \tag{2.8}$$

The $|\mathcal{M}|$ represents the count of instances present in the dataset $\mathcal{M}$, while $E(\mathcal{M})$ denotes the entropy of $\mathcal{M}$, and is computed according to the following equation:

$$E(\mathcal{M}) = -p(A)log(p(A)) - p(B)log(p(B)) \tag{2.9}$$

Definition 8: *Optimal Split Point (OSP).* Given a dataset $\mathcal{M}$ and a shapelet candidate S, the initial step entails computing the SubDist between S and every instance belonging to $\mathcal{M}$, which is followed by arranging the set of distances in a sorted manner. Subsequently, given a set of thresholds $\theta = [\theta_1, .., \theta_m]$, each θ_t can divide $\mathcal{M}$ into $\mathcal{M}_1$ and $\mathcal{M}_2$ with $SubDist(S, T^i) \leq \theta_t$ if $T^i \in \mathcal{M}_1$ and $SubDist(S, T^i) > \theta_t$ if $T^i \in \mathcal{M}_2$. The OSP(S) is the threshold that provides the highest information gain:

$$\text{InfoGain}(S, \text{OSP}(S)) \geq \text{InfoGain}(S, \theta_t), \quad \forall \theta_t \in \theta \tag{2.10}$$

Definition 9: *Shapelet.* Given a shapelet candidates set $\mathcal{SC}$, $S \in \mathcal{S}$ is regarded as a shapelet if it provides the highest InfoGain compared to other $S^* \in \mathcal{SC} \backslash \mathcal{S}$.

$$\text{InfoGain}(S, \text{OSP}(S)) \geq \text{InfoGain}(S^*, \text{OSP}(S^*)), \quad \forall S^* \in \mathcal{SC} \backslash \mathcal{S} \tag{2.11}$$

Note that, we use the binary information gain for evaluating shapelets to distinguish a class Y_i from other classes $Y \backslash \{Y_i\}$.

Chapter 3

Proposed Method 1 - A Perceptual Position-aware Shapelets Network for Time Series Classification (PPSN)

3.1 Research Motivation

Ye et al. [27] introduced a novel concept of shapelets for TSC in 2009, which refers to subsequences of time series data capable of effectively differentiating between various classes. As local patterns are generally more informative than global structures for this task, shapelets have proven to be a remarkable success in TSC. Additionally, shapelet-based approaches have the added advantage of producing easily interpretable results.

Typically, shapelet-based classifiers are comprised of two primary stages: (i) a shapelet initialization phase that picks the final shapelets using quality assessments (such as Kruskal-Wallis statistic, F-statistic, or information gain) after identifying shapelet candidates from the training time series; (ii) a shapelet learning phase that fine-tunes the shapelets using a neural network model through a gradient descent algorithm. However, current shapelet-based approaches suffer from limitations in both of these stages.

In the case of the shapelet initialization phase, the initial shapelet-based classifiers [27, 17] use Full Extractor to find all potential shapelet candidates. After that, they utilized the Euclidean Distance method to quantify the distance between these candidates and the target time series. Although this approach produces high performance, suffering a high level of computational complexity. To tackle this issue, [11] proposed to use the Fixed-Length Ex-

tractor, which only extracts candidates of the same length. They then clustered these candidates using k-Means and utilized the k-Means centroids as the initial shapelets. Nevertheless, this method assumes that the length of shapelets is fixed, even though the dataset may contain shapelets of various lengths. To overcome the issue of computational complexity, several recent attempts have been made to speed up the shapelet extractor process and optimize complex parameters (for instance, the quantity and length of shapelets) automatically by utilizing piecewise aggregate approximation (PAA). However, this approach may result in the loss of data characteristics [15, 23, 28]. Additionally, relying on Euclidean Distance to compute the subsequence distance between shapelet candidates and the primary time series requires considerable time and unintentionally disregards the location of the shapelets (**Problem 1**).

For the learning shapelet phase, conventional techniques have utilized direct learning from subsequence distances, which refer to transformed values of shapelets and target time series. Nevertheless, this approach poses challenges in training and convergence, as certain shapelets may provide extremely high values, while others may yield significantly smaller values. Additionally, the linear layer in the final network typically produces unsatisfactory predictions in the first training epochs due to suboptimal weights. These challenges may reduce the overall performance of the model (**Problem 2**).

3.2 Research Contribution

In our first work, we propose a novel method for time series classification, namely Perceptual Position-aware Shapelets Network (PPSN) to solve all mentioned problems. For the shapelet initialization phase, we first build a Perceptual Shapelet Extractor (Section 3.3.1) that uses three consecutive important points to automatically extract a few prominent shapelet candidates. Next, we propose Position-aware Subsequence Distance (Section 3.3.2) in order to evaluate shapelets, which effectively leverage position information to calculate the distance of the shapelet candidate and its corresponding time series instead of comparing it to the whole original time series, as a result, improving performance and speeding up computational time. Ultimately, high-quality shapelets of different lengths are successfully kept (**Problem 1**). To tackle the detrimental effects of each shapelet's various value ranges, our model applied Fixed Normalization (Section 3.3.3) to the transformed values of each shapelet throughout the learning phase. In addition, our PPSN also employs stop-gradient connections during the first few epochs (Stop-gradient Epochs, Section 3.3.3) to minimize the undesirable effects of

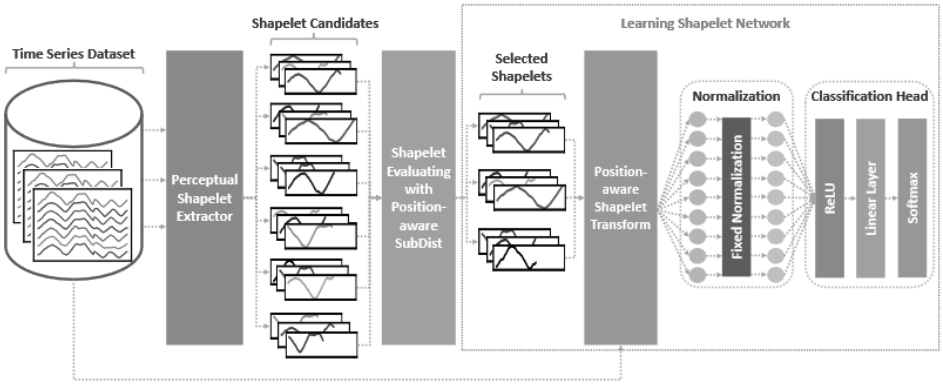


Figure 3.1: The overall architecture of the Perceptual Position-aware Shapelet Network (PPSN).

suboptimal weights in the final linear layer's (**Problem 2**).

The main contributions of our first work can be summarized as follows: (i) We propose PPSN - a novel shapelet-based approach, that incorporates an effective shapelet extractor and use positional information to compute subsequence distance. (ii) We propose two novel techniques, namely fixed normalization and stop-gradient epochs techniques to enhance the accuracy of the model. (iii) We carry out in-depth experiments on 112 UCR datasets, and the results indicate that PPSN surpasses non-ensemble methods and achieves SOTA performance. Furthermore, PPSN retains the interpretability of the model and has a low computational time.

3.3 Methodology

In this part, we provide a comprehensive explanation of PPSN. For the shapelet initialization phase, we propose two novel methods, namely Perceptual Shapelet Extractor (Section 3.3.1), and Position-aware SubDist (Section 3.3.2). For the shapelet learning phase, we propose two novel techniques, namely Fixed Normalization (Section 3.3.3) and Stop-Gradient Epochs (Section 3.3.3). Fig. 3.1 outlines the general PPSN's architecture.

3.3.1 Perceptual Shapelet Extractor

The most crucial part of shapelet-based classifiers is extracting shapelet candidates. As a result, the performance of the model can be improved by the

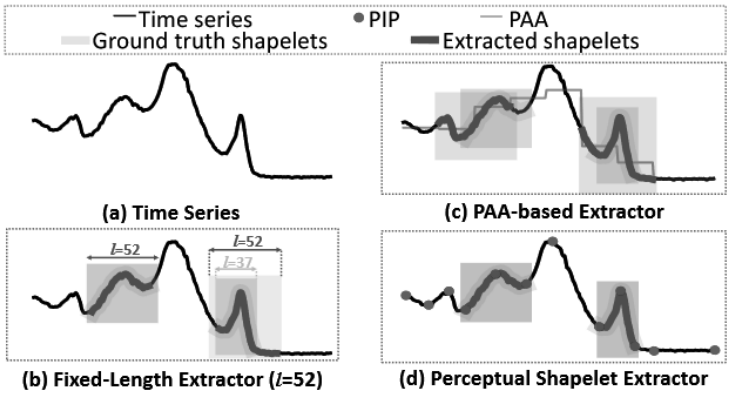


Figure 3.2: Shapelets from the Beef dataset that were chosen by several evaluated extractors. Ground truths are shapelets that have the most information gain .

Number of consecutive PIPs	2	**3 (Default)**	4	5	Full Extractor
Average InfoGain	0.501	**0.631**	0.601	0.581	0.652

Table 3.1: Average Information Gains of PSE on the first 10 UCR datasets using various consecutive PIP numbers.

high-quality shapelets [15, 28]. The existing extractors, however, have their own issues. For instance, the Full Extractor employed in [27, 17] extracts from the dataset all candidates that, after evaluation, can deliver the highest quality, but its complexity is the main issue. To tackle this problem, Fixed-Length Extractor [11] picks out candidates with the same length l in order to get around the issue. Finding the ideal fixed length is challenging, however, the approach needs the shapelet's length as a parameter. Moreover, shapelets in time series can vary in terms of lengths; hence, limiting shapelet length can diminish accuracy. For example, the Fixed-Length Extractor that produces the highest information (with $l = 52$) is unable to extract the shapelets that perfectly cover the second ground truth of length 37. It should be noted that the ground truths are the shapelets with the highest infogain extracted by the Full Extractor. To minimize the complexity, [23, 15, 28] suggested using the PAA-based extractor. The loss of specific data qualities could, however, hurt the methods. As can be seen in Fig. 3.2(c) the extracted shapelet has larger dimensions on both ends than the ground truths as it only uses segments with less information.

In this section, we propose the Perceptual Shapelet Extractor (PSE) approach, which utilizes Perceptually Important Points (PIPs) to effectively

identify high-quality shapelet candidates of varying lengths. We carried out an experiment, as detailed in Table 3.1, to demonstrate that three consecutive PIPs result in the highest infogain score, which is nearly equivalent to the score achieved by the Full Extractor (**0.631** compared to **0.652**). Algorithm 2 presents the pseudocode for PSE. In particular, for each newly extracted Perceptually Important Point (PIP), p, three potential new candidates are assessed and added to the pool of candidates, provided that they exist (lines $9 \rightarrow 15$). Fig. 3.2(d) provides a visualization of how PSE is capable of extracting shapelets that are strikingly similar to the ground truths.

Algorithm 2. Perceptual Shapelet Extractor

Input:

- $\mathcal{M} = [T^1, ..., T^m]$: dataset
- k: number of PIPs
- n: time series length

Output:

- $\mathcal{SC}$: a set of shapelet candidates
- $\mathcal{SC}\_start\_pos$: a set of starting position for each candidates
- $\mathcal{SC}\_end\_pos$: a set of ending position for each candidates

```
01: SC = SC_start_pos = SC_end_pos = []
02: for i ∈ [1, m]:
03:     P = [1, n] # a list of PIPs
04:     for j ∈ [1, k − 2]
05:         Find p ∈ [1, n] and p ∉ P with max PD(T^i[pos], P)
06:         P.add(p).sort()
07:         for z ∈ [0, 2]:
08:             # if candidates are valid add them into SC
09:             if p − z ≥ 1 and p − z + 2 ≤ |P| then
10:                 start_pos = P[p − z]
11:                 end_pos = P[p − z + 2]
12:                 SC.add(T^i[start_pos : end_pos])
13:                 SC_start_pos.add(start_pos)
14:                 SC_end_pos.add(end_pos)
```

3.3.2 Position-aware Sub-Distance for Shapelet Evaluation

Position-aware Sub-Distance (PSD). The distance between a shapelet and its best matching location in a time series instance is known as the sequence distance (SubDist). The SubDist of shapelet candidates is typically calculated using ED, which disregards the positional information of the

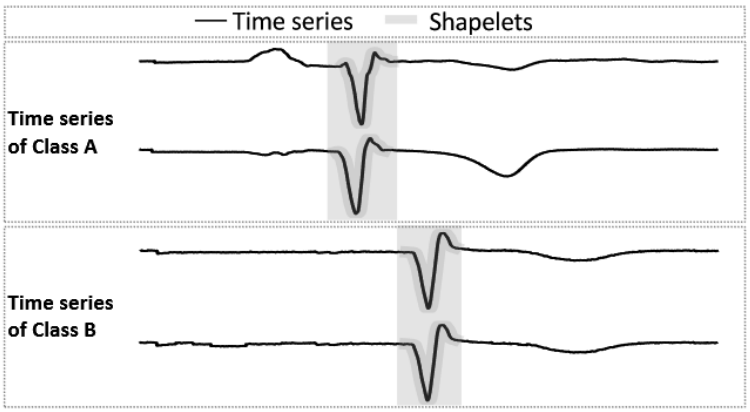


Figure 3.3: The same shapelet appears differently in two classes of the CinCECGTorso dataset.

shapelet, to the full target time series. As a result, when the position of the shapelet serves as the primary distinction between time series of various classes, it considerably increases their computing cost and makes them vulnerable. As can be seen in Fig. 3.3, two of the first time series all come to class A, while the two last time series are in class B. Clearly, they all have the same subsequence, but the occurrence positions are different. To solve this problem, we employ a Position-aware SubDist (PSD) approach, which exclusively computes the SubDist between the shapelet and the subsequence in the target time series, while taking into account the original position of the shapelet, but widens on the left and right sides with a window size w. Provided a time series T that has a length of n, a shapelet S^i that spans from the starting point s^i to the final point e^i, and a window size w, as a result, PSD of T and S^i is determined using Eq. 3.1. It is important to note that instead of using Euclidean Distance (ED), we utilize the Complexity-Invariant Distance (CID) for SubDist calculation, as described in Eq. 2.5.

$$PSD(T, S^i) = \text{SubDist}(\mathcal{PDV}(T, S^i)) \tag{3.1}$$

$$\mathcal{PDV}(T, S^i) = \mathcal{DV}(T[s\_pos^i : e\_pos^i], S^i) \tag{3.2}$$

$$s\_pos^i = \begin{cases} s^i - w + 1, & s^i - w + 1 \geq 1 \\ 1, & \text{otherwise} \end{cases}$$

$$e\_pos^i = \begin{cases} e^i + w, & e^i + w \leq n \\ n, & \text{otherwise} \end{cases}$$

where $SubDist$ is presented at Eq. 2.4 and $\mathcal{DV}$ at Eq. 2.5. It is significant to note that since it only expands the ranges of calculating distances within a window size w, $\mathcal{DV}(T[s\_pos^i : e\_pos^i], S^i)$ only has a length of $2w + 1$.

Shapelet Evaluation with PSD. Given a set of shapelet candidates denoted as $\mathcal{SC} = [S^1, ..., S^c]$, we determine the OSP for each shapelet candidate S^i by calculating the PSD between S^i and all instances of D. Next, we select the g shapelet candidates with the highest infogain score as the chosen shapelets $S = [S^1, ..., S^g]$. Assuming that S^* is an arbitrary element in $\mathcal{SC} \backslash S$.

$$IG(S^i, OSP(S^i)) \geq IG(S^*, OSP(S^*)) \tag{3.3}$$

3.3.3 Learning Shapelet Network

The training time series instance might not have the same discriminant abilities as the ground truth shapelets. In this module, we aim to optimize it using a learning shapelet network leveraging the collection of chosen shapelets as the learnable parameters. The model consists of four parts, namely Position-aware Shapelet Transform, Fixed Normalization, Classification Head, and Stop-Gradient Epochs.

Position-aware Shapelet Transform. Our method transforms the time series using the PSD (Eq. 3.1) rather than the original SubDist using the potentiality of PSD described in Section 3.3.2. Given the input time series T and the set of shapelets $\mathcal{S} = [S^1, ..., S^g]$. The transformed vector, $Z = [Z^1, ..., Z^g]$ of T is calculated as below:

$$Z^i = PSD(T, \mathcal{S}^i), \quad \forall i \in [1, .., g] \tag{3.4}$$

Fixed Normalization. The distinct OSP of each shapelet enables it to classify its respective class and other classes, resulting in varying ranges of SubDist values. This poses a challenge in achieving convergence during model training, as certain shapelets may generate excessively high SubDist values while others yield considerably lower values. To overcome this challenge, we propose a fixed normalization approach that is applied to the transformed values of each shapelet. Specifically, provided the vector of transformed Position-ware SubDist over a mini-batch: $\mathcal{B} = [Z_i^1, ..., Z_i^b]$ of Shapelet S^i, in that b is the quantity of time series instances in the batch. The normalization vector $V^i = [V_1^i, ..., V_b^i]$ of Z_i is computed as follows:

$$V_j^i = 1 - \frac{Z_j^i}{\sigma}, \quad \forall j \in [1, .., b] \tag{3.5}$$

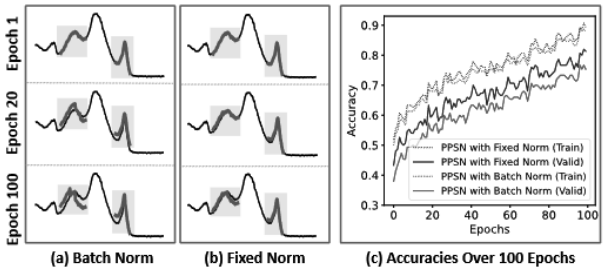


Figure 3.4: (a, b) The changing of shapelets under Batch Norm and our Fixed Norm at the 1st, 20th, and 100th epochs. (c) The average accuracies obtained from five different runs of compared methods in the Beef dataset[5]

.

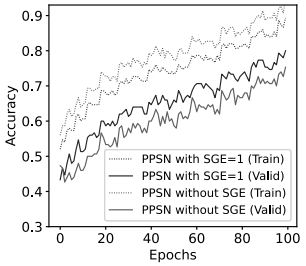


Figure 3.5: Average accuracies obtained from five different runs in Beef dataset [5] of compared methods.

in which $\sigma = max(\sigma, max(Z^i))$ is the learned parameter. Fixed normalization differs from batch normalization in that σ is updated only once, during the first epoch. The results presented in Fig. 3.4 demonstrate that by halting the update of σ, the shapelets of PPSN are less susceptible to excessive changes while achieving higher accuracies than those attained by batch normalization. It illustrates the effectiveness of our proposed fixed normalization method.

General Classification Head. Following the normalization of the transform value, we utilize a basic neural network comprising an activation function, ReLU, a single Linear Layer to optimize selected shapelets. The Softmax function is then applied to determine the predicted label. Utilizing the normalization vector $V = [V^1, ..., V^g]$, the predicted label $\hat{y} = [\hat{y}_1, ..., \hat{y}_L]$ can be determined as shown below:

$$\hat{y}_i = \text{GCH}(V) = \frac{e^{h_i}}{\sum_{j=1}^{L} e^{h_j}}, \quad \forall i \in [1, .., L] \tag{3.6}$$

$$h_i = W_{i,0} + \sum_{j=1}^{g} W_{i,j} ReLU(V^j), \quad \forall i \in [1, .., L]$$

where $W_{i,j}$ and $W_{i,0}$ refer to the weights and bias of the Linear Layer, respectively. The activation function used is ReLU, which is defined as $ReLU(V^j) = max(0, V^j)$. In this model, we employ the cross-entropy loss function:

$$Loss_{\text{gen}} = -\sum_{i=1}^{L} y_i * log(\hat{y}_i) \tag{3.7}$$

Stop-Gradient Epochs. For classifying time series, our Perceptual Shapelet Extractor offers shapelets of the best quality. Unfortunately, due to its non-optimal weights, the linear layer in the final network typically produces extremely unsatisfactory predictions during the first training epochs. The results presented in Fig. 3.5 demonstrate a significant decline in validation accuracies during the first few epochs of PPSN without Stop Gradient Epochs (SGE) when compared to PPSN with SGE. This leads to a larger gap in the model's performance. It highlights the adverse effects of suboptimal weights and the benefits of implementing stop-gradient epochs to enhance the accuracy of the model.

3.4 Experimental Result

In our first work, we conduct experiments on 112 datasets from the UCR Time Series Archive [5], utilizing the original train/test split as per the approach described in [22]. It should be noted that the datasets considered in our experiments do not include those with unequal length or missing values, and exhibit variations in their categories, the quantity of classes, the number of instances, and lengths of time series.

Following the recommendation in [8], we compare various classifiers across various datasets and present the results on a critical difference diagram that uses average ranks rather than error rates. To determine whether the pairwise classification accuracy difference between methods is statistically significant, a two-sided Wilcoxon signed-rank test is conducted, with a black horizontal line connecting the methods that exhibit no significant difference ($\alpha = 5\%$). Additionally, the Holm correction is employed as a post-hoc test to the Friedman test [8] for all comparisons.

We have built a website [1] to reproduce our experiments which contains the results on 112 UCR datasets along with the source code.

3.4.1 Hyperparameter Setting

The experiment parameters for PPSN are set as follows: Stop-Gradient Epochs and PIPs are kept constant at 1 and 0.3 of the time series length, respectively. The number of shapelets, g, is explored within the range of $\{0.1, 0.2, 0.5, 1, 2, 5, 10\}$ of the time series length. The window size, w, is determined using a simple heuristic approach, where the information gain for shapelet candidates is computed for all $\{5, 10, 20, 30, 50, 100, 200\}$ for each

[1] `https://github.com/xuanmay2701/ppsn`

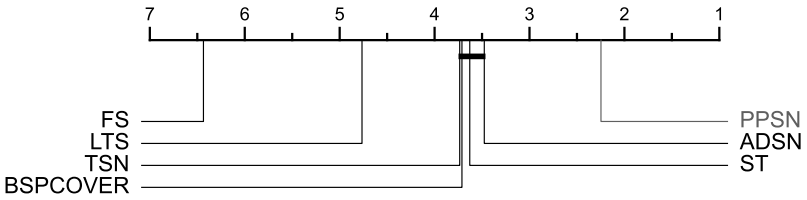


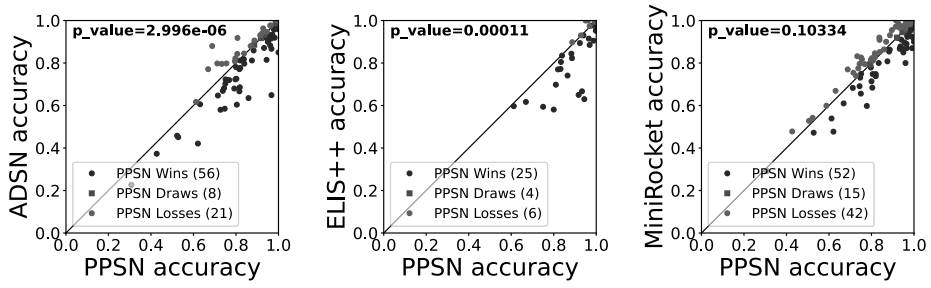
Figure 3.6: Critical different diagram presents the average ranks of PPSN and seven shapelet-based classifiers on 85 UCR Dataset. The groups with no significant difference (p-value > 0.05) are represented by solid lines.

value of g. The window size with the highest average information gain of the top g selected shapelets is then selected. Notably, PPSN requires only one parameter, g, to be tuned.

Using PyTorch, we conducted experiments and utilized the AdamW optimizer with a learning rate of 0.01 and momentum of 0.9. We then applied a Smoothing Label of 0.1 for all datasets and determined the batch size based on the dataset size. In particular, we selected the batch size from $\{16, 32, 64, 128, 256\}$ when the number of training instances exceeded $\{0, 100, 200, 400, 800\}$, respectively. For instance, if there were 500 training instances, we assigned the batch size as 128.

3.4.2 Compared with Shapelet-based Methods

This section presents the experimental comparison between our proposed PPSN and six existing shapelet-based classifiers that represent the state-of-the-art in the field. These include Learning Time Series Shapelet (LTS) [11], Shapelet Transform (ST) [17], Fast Shapelet (FS) [23], BSPCOVER [15], Triple-Shapelet Network (TSN) [19], and Adversarial Dynamic Shapelet Network (ADSN) [20]. Following the methodology outlined in previous studies [20, 19, 15], we present our results on 85 of the UCR datasets. Our comparison does not include ELIS++ [28] since their reported results are limited to only 35 out of the 85 datasets. The critical difference diagram presented in Fig. 3.6 enables us to compare the performance of our PPSN model with the baseline classifiers. The results show that our proposed method outperforms all other shapelet-based classifiers, achieving the highest rank. Additionally, we present a pair-wise comparison of our PPSN with ADSN in Fig. 3.7(a) and with ELIS++ in Fig. 3.7(b). The results demonstrate that PPSN performs better (including equal performance) than ADSN and ELIS++ on a majority of datasets, specifically on 64 out of 85 datasets for ADSN and 29 out of 35 datasets for ELIS++.



(a) PPSN vs ADSN (b) PPSN vs ELIS++ (c) PPSN vs MiniRocket

Figure 3.7: Scatter charts indicate the accuracy of our proposed method (PPSN), ADSN, ELIS++, and MiniRocket. Each data point depicted the accuracy achieved on a particular dataset. In order to ensure a fair comparison, we only evaluate the result of datasets that had been previously reported. Specifically, the comparison was conducted on a total of 85, 35, and 109 datasets for ADSN, ELIS++, and MiniRocket, correspondingly.

3.4.3 Compared with Current State-Of-The-Art Methods

We evaluate the performance of PPSN against seven state-of-the-art (SOTA) methods, which are considered to be the most precise techniques for time series classification at present. These methods include: (i) 4 ensemble-based methods HIVE-COTE (HC1) [18], HIVE-COTE 2.0 (HC2) [22], TS-CHIEF [26], InceptionTime [14]; (ii) 2 feature-based methods Rocket [6], MiniRocket [7]; and (iii) interval-based algorithms DrCIF [22]. We select these methods for comparison based on their high level of accuracy.

In this section, we perform experiments on a total of 112 datasets, but we report the results of only 109 datasets to comply with the protocol described in [7]. Fig. 3.8 presents the average ranking of our proposed model (PPSN) and other SOTA methods. On average, PPSN achieves higher accuracy than MiniRocket and InceptionTime but is relatively less accurate than the most accurate ensemble classifiers available, particularly TSCHIEF, HIVE-COTE, and HIVE-COTE 2.0, although these differences are not statistically significant. Nevertheless, it should be noted that InceptionTime, HIVE-COTE, TS-CHIEF, and HIVE-COTE 2.0 are all ensemble methods that combine numerous models, including multiple shapelet-based classifiers. Additionally, our proposed model is significantly faster than these ensemble methods in terms of computation time. We also present a scatter chart for pairwise comparison of PPSN and the top-performing non-ensemble method,

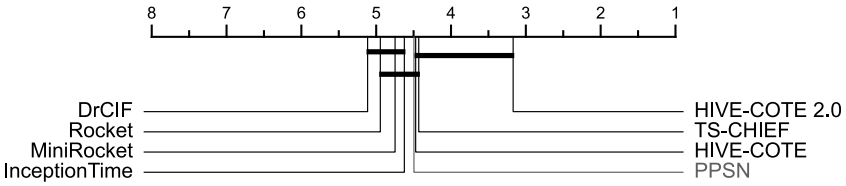


Figure 3.8: A critical different diagram indicates the average ranks of compared methods on 109 UCR datasets. As we can see, PPSN significantly outperforms all methods except HIVE-COTE 2.0. Note that, HIVE-COTE 2.0, TS-CHIEF, HIVE-COTE, and InceptonTime are ensemble methods that all have extremely high computational costs.

Methods	DrCIF	Rockett	MiniRocke	InceptionTime	HC1	TS-CHIEF	HC2
Training time	45.412	2.853	0.254	86.581	340.212	1016.874	427.183

Our Methods	PPSN (64 thread)		PPSN (32 threads)		PPSN (1 threads)	
	Initializing Shapelet	Shapelet Learning	Initializing Shapelet	Shapelet Learning	Initializing Shapelet	Shapelet Learning
Training time	2.472	0.624	4.233	0.624	13.732	0.624

Table 3.2: The run time (in hours) necessary to train 109 UCR datasets was calculated by executing the shapelet initialization phase on a single thread utilizing an AMD EPYC 7H12 2.6GHz CPU and running the shapelet learning phase threads on an NVIDIA A40 GPU.

MiniRocket (refer to Fig. 3.7(c)). The chart illustrates that PPSN is superior or equally accurate to MiniRocket on the majority of datasets (68 out of 109 datasets), with a p-value of 0.10334. All experimental results of PPSN are presented in Table 4.2.

3.4.4 Computation Time Comparison

Table 3.2 demonstrates that PPSN trains all 109 UCR datasets in 14.35 hours, which is much less time than all other SOTA techniques except Rocket and MiniRocket. Particularly, PPSN is 34 times quicker than HC2 and two orders of magnitude faster than TS-CHIEF. Furthermore, the processing time is substantially sped up by running PPSN in several threads. For instance, if we train PPSN using 32 or 64 threads, the training time is reduced significantly to only 4.23 and 2.47 hours, respectively. It is noted that the shapelet initialization phase in PPSN is responsible for the majority of the running time, it takes 13.73 hours, which is much longer than only 0.62 hours for the shapelet learning phase when using a single thread. This implies that PPSN

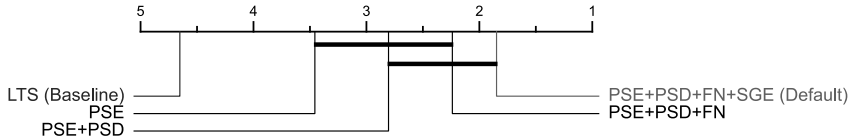


Figure 3.9: The average ranks of the LTS baseline and four ablation versions of PPSN.

has a very fast testing time, requiring only around 10 minutes to run on all 109 UCR datasets. These outcomes indicate the superiority of our proposed method in terms of computing efficiency.

3.4.5 Ablation Study and Sensitivity Study

In order to assess the impact of proposed components and the important parameter selection for PPSN, we perform a variety of experiments on the first 30 UCR datasets.

Component Evaluation. Initially, we access the influence of four proposed components of our PPSN model, namely, the Perceptual Shaplet Extractor (PSE) in Section 3.3.1, Position-aware SubDist (PSD) in Section 3.3.2, Fixed Normalization (FN), and Stop-Gradient Epochs (SGE), also at Section 3.3.3. To determine the impact of each component on the final accuracy, we incorporated them one by one. Fig. 3.9 demonstrates that all four components positively affected the final model's results by increasing its accuracy.

Number of Perceptually Important Points. We perform experiments to run our PPSN with various PIP values, and we calculate the average information gain (Eq. 2.11) of a few chosen shapelets. The parameter used is associated with the length of the time series. This implies that, for a time series of length n, the value of k is determined as $k = n * npips$. As can be seen in 3.3, the average information gain of PPSN with $npips = 0.3$ (the default parameter) is almost equivalent to that of $npips = 0.4$ and $npips = 0.5$, while significantly fewer candidates are extracted. Moreover, the information gain of PPSN with $npips = 0.3$ is remarkably similar to that of Full Extractor, at 0.633 and 0.652, respectively, even though PPSN extracts only 280 candidates, while Full Extractor extracts 102,380 candidates.

Number of Stop-Gradient Epochs. The impact of varying the number of Stop-Gradient Epochs (SGE) on the performance of our PPSN model is

Number of PIPs (npips)	0.1	0.2	**0.3 (Default)**	0.4	0.5	Full Extractor
Avg. Information Gain	0.592	0.611	**0.631**	0.633	0.635	0.652
Avg. No. Extracted Candidates	100.1	195.2	**280.2**	370.5	475.5	102380.9

Table 3.3: The table compares the averaging information gain and number of extracted shapelet candidates of our PPSN with various number of PIPs.

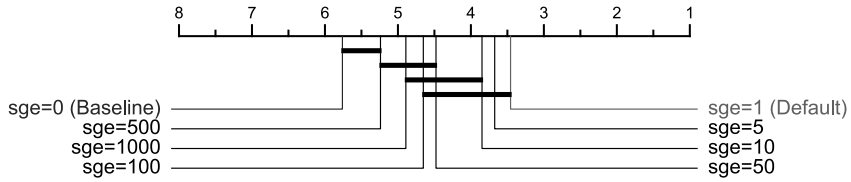


Figure 3.10: The average ranks for PPSN with various numbers of Stop-Gradient Epochs.

illustrated in Fig. 3.10. The results indicate that all versions of PPSN with different numbers of SGE (varying from 1 to 1000) outperform the baseline. However, there is little advantage in increasing the number of SGE. Eventually, the best performance was achieved by PPSN with SGE set to 1.

Normalization. The comparison of Fixed Normalization, Batch Normalization, and Baseline (without Norm) is presented in Fig. 3.11. The results show that while applying normalization improves the performance of PPSN, Fixed Normalization provides a considerably better outcome compared to its counterparts.

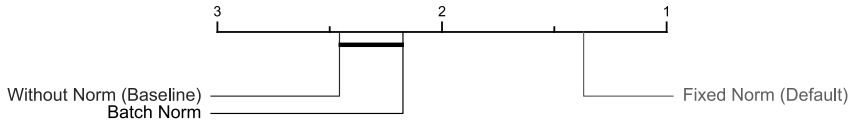


Figure 3.11: The average ranks for two novel techniques in the learning shapelet phase, namely Fixed Normalization and Batch Normalization.

3.4.6 Experiments on Interpretability

The interpretability of shapelets is a valuable capability that enhances data comprehension. Fig. 3.12 demonstrates that shapelets can discriminate between two classes of the ECGFiveDays dataset [5], which is used in the study of the heart through electrocardiography (ECG). The time series instances

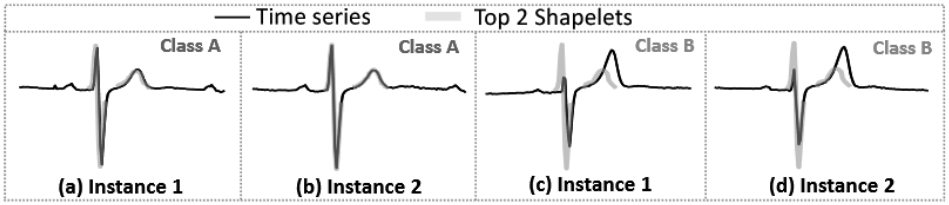


Figure 3.12: Shapelets were chosen by PPSN for the ECGFiveDays dataset.

in Fig. 3.12 (a) and (b) belong to Class A, while those in Fig. 3.12 (c) and (d) belong to Class B. The selected shapelets by PPSN (orange lines) effectively highlight the major differences between the segments of the two classes. In particular, the first shapelet depicts a QRS complex, while the second shapelet depicts a T wave of ECG. It is evident that the T wave shows a larger peak than the QRS complex in Class B, which is known as a hyperacute T wave in the medical field, occurring in certain illnesses like ischemia or hyperkalemia.

Chapter 4

Proposed Method 2 - A Stronger Shapelet-based Time Series Classifier using Distance Learning and Binary Auxiliary Head (PPSN++)

4.1 Research Motivation

In our first work, we propose a novel method called Perceptually Position-aware Shaplet Network for TSC (PPSN, Chapter 3). It has demonstrated achieving remarkable accuracy in the task. However, PPSN as well as shapelet-based classifiers still have some shortcomings.

Firstly, they utilized a soft-minimum function to pick out the lowest distance features. This function calculates the minimum distance between a shapelet and its best-matching subsequences within the time series, only considering the distance of the best-matching subsequence and disregarding others. Consequently, crucial information could be missed during the training process, and significant patterns within the data could be overlooked. Additionally, the function has been shown to occasionally generate a value exceeding the system's maximum limit, resulting in model failure in some cases (**Problem 3**).

Secondly, the general classification head (GCH) is undoubtedly a valuable component for various time series classification tasks. However, its effectiveness may be limited when used in the specific context of shapelet-based methods. The major issue is that GCH may not be optimized to suit the

particular characteristics of the input time series data. The use of shapelets can efficiently differentiate between distinct classes and mitigate the problem of data imbalance. To achieve this, PPSN proposes the extraction of shapelets using binary information gain and an equal number of shapelets for each class. Nevertheless, PPSN only employs the general classification head to train these shapelets, leading to shapelets that learn more global information and less specific information about their respective classes. This directly impacts the model's performance, as capturing the specific patterns of the data is crucial for the success of shapelet-based models (**Problem 4**).

4.2 Research Contribution

In our second work, we propose a stronger shapelet-based classifier that is built upon the PPSN framework (called PPSN++) to address the problems mentioned in Section 4.1.

On the one hand, we propose to use a Distance Learning Layer (DLL) instead of relying solely on the minimum distance selection. The DLL effectively utilizes a significant pattern of the distance vector to compute the shapelet-transformed feature between the shapelet and the target time series. By utilizing PSD (as described in Chapter 3, Section 3.3.2), the number of elements in the distance vector is reduced to only $2w + 1$, which is much less compared to the conventional SubDist. Furthermore, to prevent overflow values, the DLL is designed to utilize fixed normalization, which ensures that the distance values remain within the range of 0 to 1 (**Problem 3**).

On the other hand, we propose to use both the General Classification Head (GCH) and the Binary Auxiliary Classification Head (BACH) simultaneously in order to improve the accuracy of time series classification. BACH is designed to enable more specific learning of shapelet candidates from their respective classes. Combining BACH with GCH can also enhance the effectiveness of the distance learning layer by ensuring that the output of the DLL always tends towards the value of 1 for all target time series of its class (**Problem 4**).

The main contributions of our second work can be summarized as follows: (i) We propose PPSN++ which is a stronger shapelet-based method using a Distance Learning Layer to capitalize on significant distances (ii) We introduce Binary Auxiliary Classification Head to capture specific patterns of their respective classes (iii) We conduct extensive experiments on 112 UCR datasets, and the results indicate that PPSN++ outperforms PPSN, making it become the state-of-the-art shapelet-based classifier for time series classification.

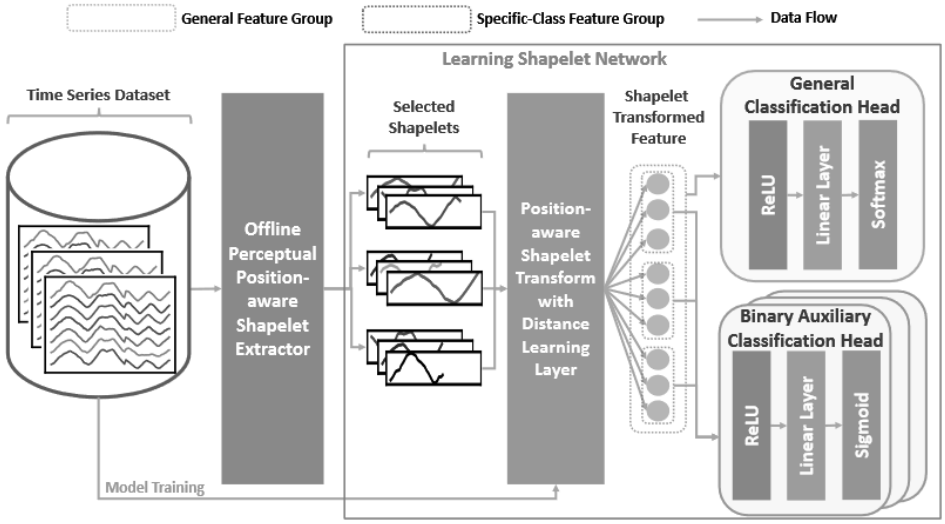


Figure 4.1: The chart illustrates the general architecture of PPSN++. First, it extracts high-quality shapelets using the Offline Perceptual Position-aware Shapelet Extractor. Second, PPSN++ transforms time series data using the Position-aware Shapelet Transform with a Distance Learning Layer and then feeds the transformed values into both a General Classification Head and a Binary Auxiliary Head. This model is an extended version of our previous work, PPSN, with two main improvements, including a Distance Learning Layer to capitalize on significant distances and a Binary Auxiliary Head to capture specific patterns of their respective classes.

4.3 Methodology

In this section, we present an extended Shapelet-based Time Series Classifier, referred to as PPSN++, which builds upon the PPSN framework with two primary enhancements: Distance Learning Layer (Section 4.3.1) and a Binary Auxiliary Head (Section 4.3.2). Firstly, PPSN++ extracts high-quality shapelets by employing the Offline Perceptual Position-aware Shapelet Extractor (Section 3.3.1). Subsequently, the selected shapelets are incorporated into the Position-aware Shapelet Transform with a Distance Learning Layer to transform the original time series data. Ultimately, the transformed values are concurrently fed into both the General Classification Head and the Binary Auxiliary Head, with the overarching goal of classifying time series data. The comprehensive architecture of PPSN++ is depicted in Fig. 4.1.

4.3.1 Distance Learning Layer

Numerous prior approaches have employed a soft-minimum function to determine the distance between shapelets and the target time series. This function extracts the minimal distance between a shapelet and its best-matching subsequences within the time series (the lowest of SubDist). As a result, it only considers the best-matching subsequence and disregards other distances, which could lead to important information being overlooked and significant patterns within the data being missed during the training process. Furthermore, the function is known to occasionally generate a value surpassing the maximum value the system can handle, leading to model failure in specific instances.

To tackle this issue, we propose to use a Distance Learning Layer (DLL) rather than exclusively selecting the minimum distance. Our DLL efficiently capitalizes on a significant pattern of distances vector to compute the shapelet transformed feature between the shapelet and target time series. Thanks to exploiting PSD (Section 3.3.2), the number of elements in the distance vector is only $2w + 1$, which is significantly less than the conventional SubDist. Additionally, the distance learning layer (DLL) is designed to prevent the production of overflow values by using fixed normalization, which keeps the distance within the range of 0 and 1.

Given a time series T and a shapelet $S^i \in \mathcal{S}$, the distance vector of T and S^i using our Position-aware Sub-Distance (PSD) is represented by:

$$D^i = [D^i_0, ...D^i_{2w}] = \mathcal{PDV}(T, S^i) \tag{4.1}$$

In that, $\mathcal{PDV}$ is computed using Eq. 3.2. It is important to note that PSD solely computes the SubDist between shapelet S^i and the corresponding subsequence in the target time series T. Additionally, we extend the width on both sides by w, resulting in the distance vector containing only $2w + 1$ elements.

The DLL initially normalizes the distance feature vector D^i to obtain $\bar{D}^i = [\bar{D}^i_0, ..., \bar{D}^i_{2w}]$, where $\forall \bar{D}^i_j \in [0, 1]$, based on fixed normalization (Eq. 3.5) using $\mathcal{B}^i_{D^i}$, which encompasses all distances within the batch, and $\sigma = \max(\sigma, \max(\mathcal{B}^i_{D^i}))$:

$$\bar{D}^i = ReLU(\text{FixNorm}(\mathcal{B}^i_{D^i}, i)) = ReLU(1 - \frac{D^i}{\sigma}) \tag{4.2}$$

We utilize the ReLU function to prevent the FixNorm from acquiring negative values, which can occur since σ is only updated during the first epoch, while the distance value may exceed σ in subsequent training epochs.

Subsequently, the normalized distance vector is transformed to $Z^i = [Z_0^i, ..., Z_{2w}^i]$ as follows:

$$Z_j^i = \text{DLL}(\bar{D}_j^i) = \bar{D}_j^i * M(\bar{D}_j^i) * B(j) \tag{4.3}$$

We propose two additional terms to enhance the learning of the distance vector $\bar{D}^i$. The first one, $M(\bar{D}_j^i)$, is inspired by the soft-minimum function, with the aim of reducing the impact of low distances. It is defined as follows:

$$M(\bar{D}_j^i) = e^{-\alpha(1-\bar{D}_j^i)} \tag{4.4}$$

Given that the values of $\bar{D}^i$ lie within the range [0, 1], it is ensured that $M(\bar{D}_j^i)$ will not cause an overflow, provided that the value of α remains within reasonable bounds. This function thus effectively diminishes the influence of low distances without generating overflow errors.

The second term is designed to emphasize the importance of the position information of the shapelet by reducing the impact of features located at the border positions of $\bar{D}^i$. Notably, the vector $\bar{D}^i$ is obtained by computing the distance between the shapelet S^i and the corresponding subsequence in the target time series T. Additionally, we widen the window on both sides by w, resulting in a distance feature vector containing only $2w + 1$ elements. Hence, the outermost position in $\bar{D}^i$ corresponds to the highest shift of the subsequences. We then develop a learnable function that can decrease the outer values in $\bar{D}^i$ by a quantity controlled by the learnable $\beta \in [0, 1]$ parameter, as expressed by the following equation:

$$B(j) = -\frac{(1-\beta)}{(1+\beta)w^2}(j-w)^2 + 1 \tag{4.5}$$

Examples of $B(j)$ are illustrated in Fig. 4.2, where it can be observed that:

- $B(j)$ takes the shape of an upside-down parabola function, with the highest value $B(j) = 1$ at $j = w$. This makes the middle values maintain a constant value of 1 and diminish toward the two boundaries.
- $B(j) \in [0, 1], \forall j \in [0, 2w]$. This makes sure that Z_j^i values decrease or remain within the range of 0 to 1.
- A higher value of β results in a smaller slope of the parabolic curve. Thus, the function B can reduce the outer values in $\bar{D}^i$ by an amount determined by the learnable β parameter.

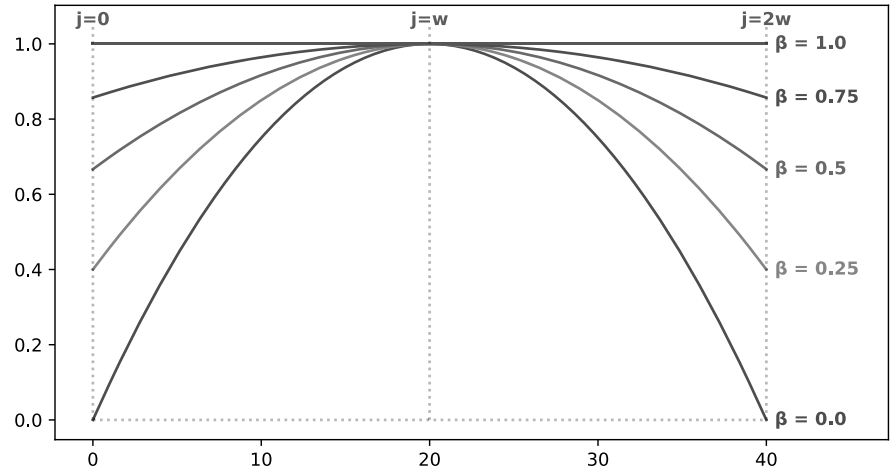


Figure 4.2: The graph illustrates the different values of function $B(j), \forall j \in [0, 2w]$ with different values of β. Specifically, the function $B(j)$ exhibits the shape of an inverted parabolic curve, reaching its maximum value of 1 at $j = w$, resulting in middle values that remain constant at 1 and diminish towards the two boundaries. Furthermore, the function is restricted to the interval of $[0, 1]$ for all values of j in the range of $[0, 2w]$. This guarantees that the value of the function will either decrease or remain within the limits of 0 and 1, precluding any negative values. Additionally, the slope of the parabolic curve is determined by the learnable parameter β, which when increased, results in a smaller slope and reduces the outer values in $\bar{D}^i$.

Finally, we utilize linear regression to learn the shapelet transformed feature, which is represented by the equation:

$$V = \frac{1}{2w+1} \sum_{i=0}^{2w} W^i * Z^i + b^i \tag{4.6}$$

where Z^i is the i^{th} element of the shapelet transformed feature vector Z, W^i is the corresponding weight, and b^i is the bias term. The aim of this process is to derive a linear combination of distance vector and efficiency to determine the weight for each position in the vector $\bar{D}^i$.

The interpretability of DLL: The objective of DLL is that can produce a distance feature with avoiding overflow values, while still retaining interpretability. From that, β is used solely to control the slope of $B(j)$, which can better indicate the similarity of shapelet and subsequences in T. After that, final linear regression is used to only determine the weight for each po-

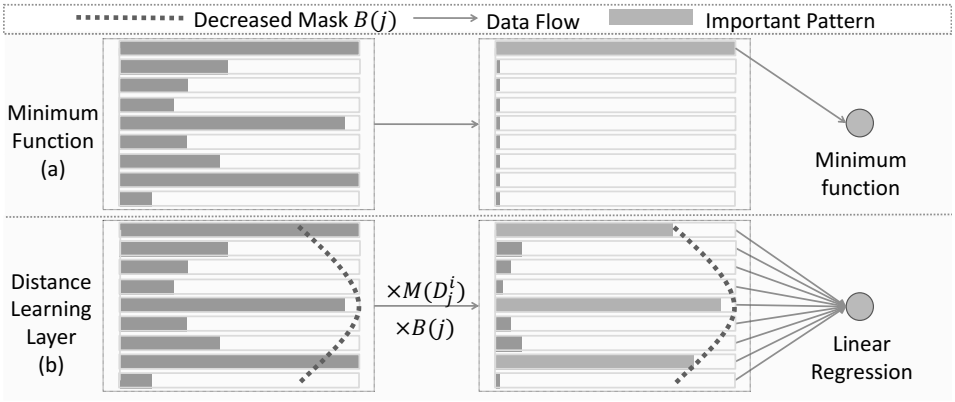


Figure 4.3: The graph illustrates how the DLL and classic minimum functions work. We solely compare the classic minimum function in order to better illustrate the differences between them. Additionally, although the soft-minimum function employs distinct computational techniques, it nonetheless generates the same results. It is of significance to mention that for ease of comprehension, we utilize the normalization process represented by Eq. 4.2 to scale the value range between 0 and 1. Herein, a value of 1 indicates the smallest distance, whereas a value of 0 represents the highest distance.

sition in the vector D and the inherent property of linear regression is that it involves the averaging sum of the vector D. Therefore, it is noteworthy that neither of these techniques compromises the interpretability of the model. Moreover, the output of the DLL may be interpreted as an approximate distance between the shapelet S and the time series T.

A comparison of DLL and classic minimum function is illustrated at Fig. 4.3. Our comparison is focused only on the traditional minimum function to highlight the contrasts between them more effectively. Moreover, even though the soft-minimum function uses different computational methods, it produces identical outcomes compared with classic minimum functions. By presenting the examples in Fig. 4.4, we demonstrate that the previous functions, which concentrate solely on the most significant value (i.e., the smallest distance), yield equivalent values for all three instances in both class A and class B. However, DLL can produce distinct values for each example by utilizing all the pattern information in the distance vector.

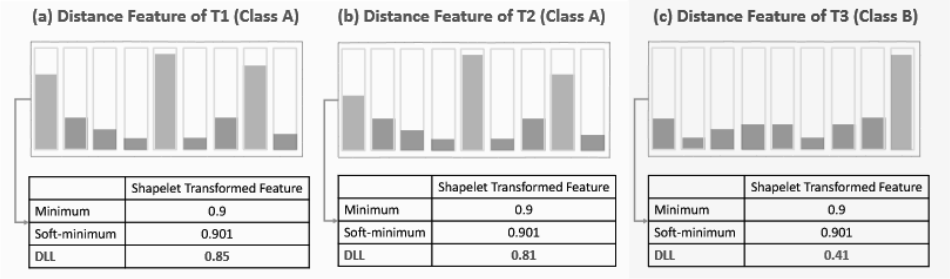


Figure 4.4: The graph illustrates how the DLL can discriminate better the classic minimum/soft-minimum functions. Since previous functions only focus on the highest value (smallest distance), they generate equal values for all examples in both classes. On the other hand, DLL can generate the different values between them by using all pattern information in Distance Vector.

4.3.2 Binary Auxiliary Classification Head

Undoubtedly, the general classification head (GCH) of PPSN can be a useful tool for multiple time series classification tasks. However, its effectiveness may be limited when applied to the specific context of using shapelets. The major problem is that GCH may not be optimized for the specific characteristics of the input time series data. According to theory, shapelets are effective at distinguishing specific classes that can address the minority class problem. Along with solving the problem of data imbalance, PPSN proposes extracting shapelets using binary information gain and equal amounts of shapelets for each class. However, PPSN only employs the general classification head to train these shapelets, resulting in shapelets that learn more global information and less specific information about their respective classes. This directly affects the performance of the model since capturing the specific patterns of the data is the key to the success of shapelet-based models.

To address these limitations, we propose the concurrent utilization of the GCH in conjunction with our proposed approach, the Binary Auxiliary Classification Head (BACH), to classify time series more accurately. BACH aims to enable the shapelet candidates to be learned more specifically from their respective classes. Additionally, using BACH together with GCH can help the distance learning layer learn more effectively, as it ensures that all output of the DLL is always towards the value of 1 for all target time series of its class. The architecture of BACH is shown in Fig. 4.5 (right).

Given the shapelet transformed features $V = [V^1, ..., V^g]$, V is divided

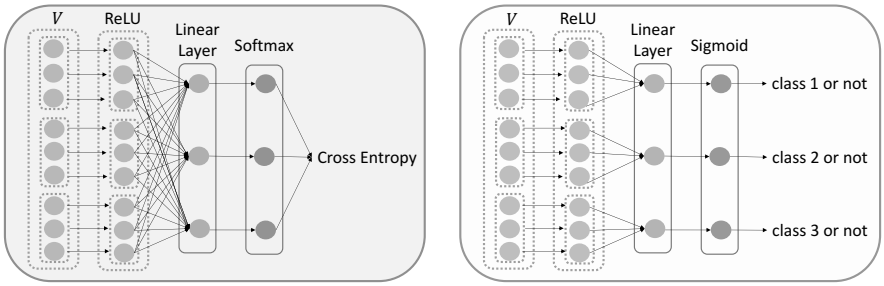


Figure 4.5: The General Classification Head (GCH, left) may not be optimized for the specific characteristics of input time series data, leading to issues in distinguishing specific classes and addressing data imbalance. PPSN attempts to solve this by using shapelets extracted using binary information gain and equal amounts for each class. However, training these shapelets using only GCH results in shapelets that learn more global information and less specific information about their respective classes, negatively impacting the performance of the model. While PPSN++ proposes to use concurrently GCH and Binary Auxiliary Classification Head (BACH, right) intending to capture specific-class patterns in the data.

into L group represented for L classes of datasets.

$$\begin{aligned} V &= [V_1, ..., V_L], \\ V_l &= [V_l^1, ..., V_l^{g/L}], \quad \forall l \in 1, ..., L \end{aligned} \tag{4.7}$$

We also define L linear layer to classify each element $V_l \in V$. The binary predicted $\hat{y}_l$ label for each class is calculated as follows:

$$\begin{aligned} \hat{y}_l &= \text{BACH}(V_l) = \frac{e^{h_l}}{e^{h_l} + 1}, \quad \forall l \in 1, ..., L \\ h_l &= W_l^0 + \sum_{j=1}^{g/L} W_l^j \text{ReLU}(V_l^j), \quad \forall l \in 1, ..., L \end{aligned} \tag{4.8}$$

Here, given $Y = [Y_1, ..., Y_L]$ is the set of different classes in $\mathcal{M}$, the binary entropy function is employed for this module:

$$Loss_{\text{aux}} = \sum_{l=1}^{L} (\hat{y}_l log(Y_l) + (1 - \hat{y}_l) log(1 - Y_l)) \tag{4.9}$$

The GCH is directly employed for V using Eq. 3.6 as $\hat{y}_i = \text{GCH}(V)$, and the loss function is obtained based on Eq. 3.7. Finally, the general loss

function is calculated as follows:

$$Loss = Loss_{\text{gen}} + \gamma Loss_{\text{aux}} \tag{4.10}$$

In that, γ is the tuning parameter.

4.4 Experimental Result

4.4.1 Hyperparameter Setting

For experiments for PPSN++, since our proposed method is not related to the tuning of the number of shapelets g and window size w. Therefore, we keep the tuned parameters of PPSN which is searched over $\{0.1, 0.2, 0.5, 1, 2, 5, 10\}$ for g and $\{5, 10, 20, 30, 50, 100, 200\}$ for w. The detail of this tuning can be found in Chapter 3, Section 3.4.1. In the case of α (Eq. 4.4) and γ of the final loss function (Eq. 4.10), we conducted the ablation study at Section 4.4.4 and fixed $\alpha = 5$ and $\gamma = 1$ respectively.

We employed PyTorch to carry out our experiments and utilized the AdamW optimizer with a learning rate of 0.01 and a momentum of 0.9. Additionally, we applied a Smoothing Label of 0.1 for all datasets and chose the batch size based on the dataset size. Specifically, we follow the setting of PPSN (Chapter 3, Section 3.4.1)and select the batch size from the set 16, 32, 64, 128, 256 when the number of training instances exceeded 0, 100, 200, 400, 800, respectively. For instance, if the dataset contained 500 training instances, we set the batch size to 128.

We have built a website [1] to reproduce our experiments which contains the results on 112 UCR datasets.

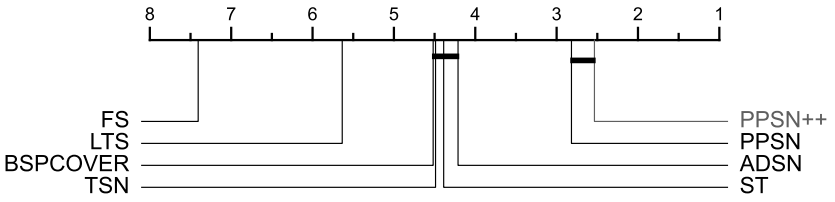
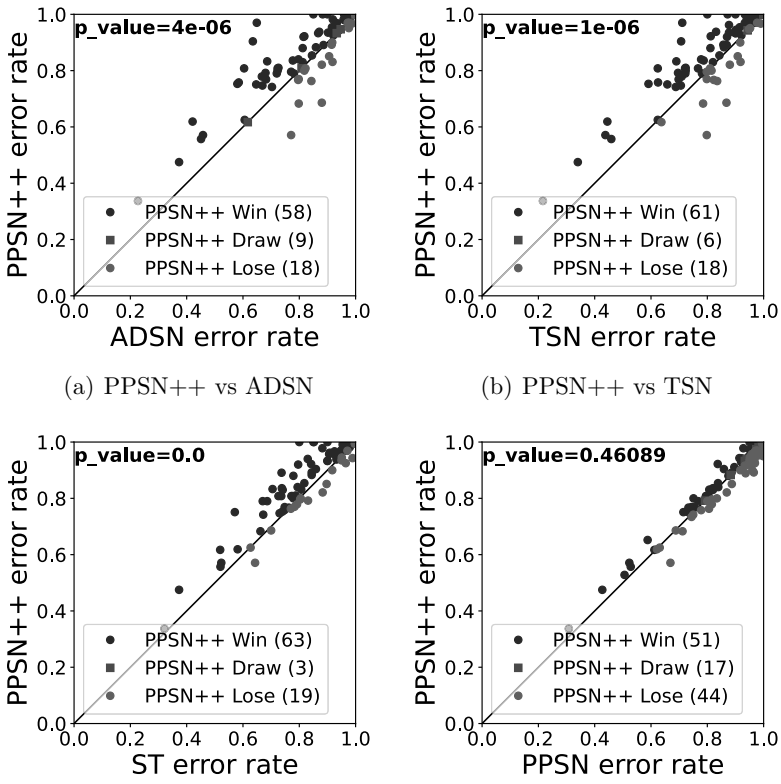


Figure 4.6: Critical different diagram presents the average ranks of PPSN++ and seven shapelet-based classifiers on 85 UCR Dataset. The groups with no significant difference (p-value > 0.05) are represented by solid lines.

[1] `https://github.com/xuanmay2701/ppsnplus`

4.4.2 Compared with Shapelet-based Methods

In Fig. 4.6, we show an experimental result for comparing our second work PPSN++ against seven state-of-the-art shapelet-based approaches, consisting of LTS [11], ST [17], FS [23], BSPCOVER [15], TSN [19], ADSN [20], and our first work PPSN (Chapter 3). We follow the protocol in [20, 15] and only illustrate the result on 85 UCR datasets. Similar to the experiments of PPSN in Chapter 3, Section 3.4.2, we also do not include ELIS++ [28] in our comparison as their results are only available for 35 out of the 85 datasets. Fig. 4.6 displays the critical difference diagram used to compare our PPSN++ model with the baseline classifiers.



(a) PPSN++ vs ADSN

(b) PPSN++ vs TSN

(c) PPSN++ vs ST

(d) PPSN++ vs PPSN

Figure 4.7: (a,b,c) Scatter charts show the accuracy of our second work (PPSN++), ADSN, TSN, and ST in a total of 85 datasets. (d) The chart compares PPSN++ and PPSN on 112 datasets. Each data point on the charts represented the accuracy achieved on a specific dataset.

Our experimental results demonstrate that our second work PPSN++ outperforms all other shapelet-based classifiers, including our first work PPSN and achieves the highest rank. Additionally, we present pair-wise comparisons between PPSN++ and ADSN in Fig. 4.7 (a), between PPSN++ and TSN in Fig. 4.7 (b), and between PPSN++ and ST in Fig. 4.7 (c). The results indicate that our PPSN++ performs better or equally well compared to ADSN, TSN, and ST in the majority of datasets (67/85, 67/85, and 66/85 datasets, correspondingly). The improvements are primarily because of the efficiency of using the Distance Learning Layer and the combination of utilizing the General Classification Head and Binary Auxiliary Classification Head.

Compared with PPSN: In this experiment, we make a comparison of our two proposed methods PPSN (Chapter 3) and PPSN++ (Chapter 4) in terms of accuracy. Fig. 4.7 (d) shows the pairwise accuracy of PPSN versus PPSN++ for the same 112 datasets. Specifically, PPSN++ is more accurate than PPSN on 51 datasets, and less accurate on 44 datasets. The large difference in accuracy between PPSN++ and PPSN on two datasets. The first one is *ShapesAll*, the accuracy of PPSN++ is **92.2** compared to PPSN **83.7**. The second is *EOGHorizontalSignal* dataset, in that, PPSN++ is more accurate than PPSN with **65.2** vs **58.8**, respectively. These results demonstrate the benefits of our proposed methods and claim the contributions of this work.

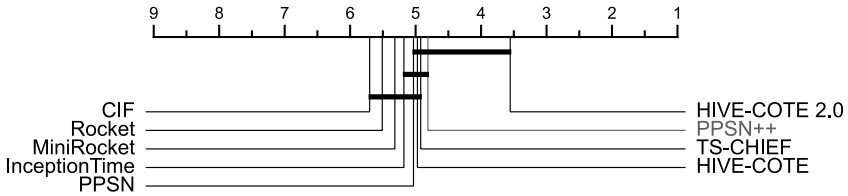
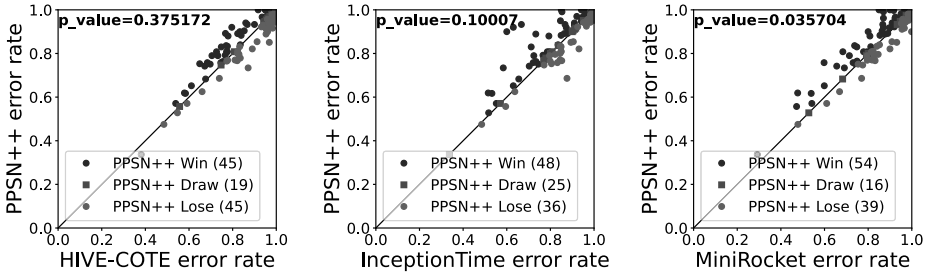


Figure 4.8: A critical different diagram indicates the average ranks of compared methods on 109 UCR datasets. As we can see, PPSN++ significantly outperforms all methods except HIVE-COTE 2.0. Note that, HIVE-COTE 2.0, TS-CHIEF, HIVE-COTE, and InceptonTime are ensemble methods that all have extremely high computational costs.

4.4.3 Compared with Current State-Of-The-Art Methods

In Fig. 4.8, our second work PPSN++ is compared with eight SOTA approaches: (i) two feature-based approaches: MiniRocket [7], Rocket [6]; (ii)



(a) PPSN++ vs HIVE-COTE (b) PPSN++ vs InceptionTime (c) PPSN++ vs MiniRocket

Figure 4.9: Scatter charts indicate the accuracy of our second work (PPSN++), HIVE-COTE, InceptionTime, and MiniRocket on 109 datasets. Each data point on the charts presented the accuracy achieved on a specific dataset.

an interval-based approach: DrCIF [22]; (iii) a shapelet-based classifier (our first work): PPSN (Chapter 3); (iv) four ensemble-based approaches TS-CHIEF [26], HIVE-COTE 2.0 [22], HIVE-COTE [18], and InceptionTime [14].

To follow the protocol at [7], we experimented on all 112 UCR datasets but only show the results of 109 datasets. It is demonstrated that PPSN++ is more accurate than PPSN (our first work), MiniRocket, InceptionTime, TSCHIEF, and HIVE-COTE and its results are comparatively less than the most accurate ensemble classifier, HIVE-COTE 2.0. The improvements, however, are not statistically significant. The experiment results show that our second work PPSN++ is a state-of-the-art non-ensemble method. Noted that HIVE-COTE 2.0 is an ensemble approach that involves the combination of approximately 40 different models (including multiple shapelet-based classifiers). As a result, our PPSN++ is significantly faster in terms of computing time than the ensemble techniques and the high accuracy of PPSN++ may also increase the accuracy of HIVE-COTE 2.0. All experimental results of PPSN++ are presented in Table 4.2.

4.4.4 Ablation Study and Sensitivity Study

To assess the impact of proposed components and the important parameter selection for PPSN++, we run a number of experiments on the first 30 UCR datasets.

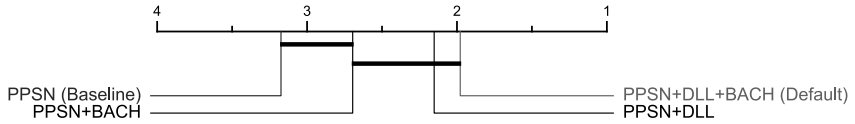


Figure 4.10: Average ranks of PPSN baseline and two ablation versions of PPSN++.

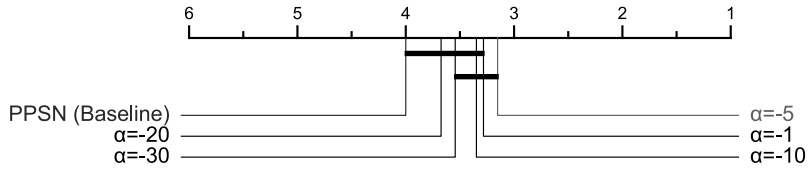


Figure 4.11: Average ranks for PPSN++ with different number of Alpha in DLL component.

Component Evaluation. We first assess the impact of two proposed components of our PPSN++, namely Distance Learning Layer (DLL, Section 4.3.1) and Binary Auxiliary Classification Head (BACH, Section 4.3.2) These components were added incrementally to measure their effect on the final accuracy of the model. As can be seen in Fig. 4.10, the proposed model's results can be improved positively by both of its components.

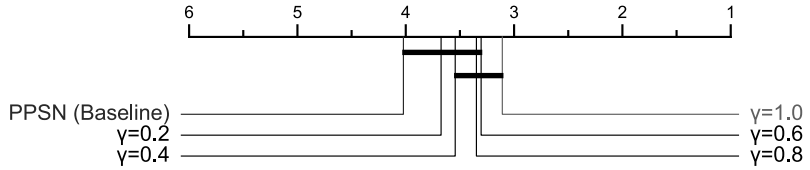


Figure 4.12: Average ranks for PPSN++ with different number of Gamma in BACH component.

Number of Alpha in DLL. Fig. 4.11 demonstrates the impact of varying the number of Alpha values in the Distance Learning Layer (DLL) of our PPSN++ model, with values ranging from -20 to -5. In comparison to the baseline model, all iterations of PPSN++ featuring varying numbers of Alpha exhibited superior performance. The PPSN++ model with Gamma equal to -5 achieved the highest level of performance.

Number of Gamma in Final Loss Function. Fig. 4.12 indicates the

comparison of our proposed Binary Auxiliary Classification Head with various values of Gamma. All variations of PPSN++ with different numbers of Gamma performed better than the baseline model. However, decreasing the number of Gamma did not provide any significant improvement. The best performance was achieved by the PPSN++ model with Gamma set to 1.

Dataset	HIVE-COTE 2.0	TS-CHIEF	HIVE-COTE	InceptionTime	MiniRocket	Rocket	CIF	PPSN	PPSN++
Adiac	0.813	0.798	0.811	**0.844**	0.818	0.77	0.821	0.755	0.77
ArrowHead	0.863	0.806	0.823	0.863	0.863	0.794	0.806	0.886	**0.891**
Beef	0.867	0.767	0.9	0.667	0.867	0.833	0.767	**0.933**	**0.933**
BeetleFly	0.95	**1**	0.95	0.85	0.85	0.9	0.9	0.95	**1**
BirdChicken	0.9	0.95	0.95	0.95	0.9	0.9	0.9	1	**1**
Car	0.917	0.867	0.833	0.9	**0.999**	0.85	0.883	0.917	0.9
CBF	**1**	0.998	0.999	0.999	0.917	1	0.999	0.998	0.997
ChlorineConcentration	0.771	0.66	0.739	**0.876**	0.77	0.807	0.739	0.688	0.686
CinCECGTorso	**1**	0.981	0.995	0.853	0.87	0.837	**1**	0.991	0.991
Coffee	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Computers	0.764	0.7	0.776	**0.808**	0.684	0.752	0.736	0.804	**0.808**
CricketX	0.826	0.821	0.808	**0.849**	0.815	0.826	0.746	0.797	0.808
CricketY	0.851	0.8	0.823	0.844	0.826	**0.856**	0.805	0.751	0.8
CricketZ	**0.864**	0.831	0.823	0.849	0.823	0.859	0.81	0.802	0.81
DiatomSizeReduction	0.958	0.964	0.935	0.948	0.925	0.967	0.863	**0.987**	0.967
DistalPhalanxOutlineCorrect	0.775	0.757	0.772	0.786	0.779	0.761	0.786	0.797	**0.808**
DistalPhalanxOutlineAgeGroup	0.748	0.748	0.755	0.734	0.748	0.748	0.748	**0.806**	0.763
DistalPhalanxTW	0.705	0.676	0.676	0.655	0.683	**0.719**	0.698	0.712	0.683
Earthquakes	0.748	0.748	0.748	0.712	0.748	0.748	0.748	0.82	**0.83**
ECG200	0.89	0.84	0.86	0.91	0.91	0.91	0.89	**0.94**	**0.94**
ECG5000	**0.947**	0.946	**0.947**	0.942	0.945	**0.947**	0.943	0.945	**0.947**
ECGFiveDays	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
ElectricDevices	0.758	**0.76**	0.746	0.721	0.731	0.726	0.736	0.747	0.758
FaceAll	0.868	0.842	0.779	0.793	0.808	**0.947**	0.764	0.875	0.88
FaceFour	**1**	**1**	0.989	0.955	0.989	0.977	**1**	**1**	**1**
FacesUCR	0.963	0.967	0.959	**0.972**	0.958	0.961	0.907	0.958	0.957
FiftyWords	0.833	**0.848**	0.776	0.846	0.84	0.833	0.782	0.818	0.833
Fish	0.989	**0.994**	**0.994**	0.983	0.971	0.971	0.943	0.914	0.943
FordA	0.954	0.95	0.951	0.961	0.95	0.942	0.969	**0.971**	0.969
FordB	0.831	0.823	0.832	**0.849**	0.812	0.806	0.816	0.807	0.831

Table 4.1: The experiments of our PPSN and PPSN++ compared to 7 state-of-the-art methods.

Dataset	HIVE-COTE 2.0	TS-CHIEF	HIVE-COTE	InceptionTime	MiniRocket	Rocket	CIF	PPSN	PPSN++
Adiac	0.813	0.798	0.811	**0.844**	0.818	0.77	0.821	0.755	0.77
ArrowHead	0.863	0.806	0.823	0.863	0.863	0.794	0.806	0.886	**0.891**
Beef	0.867	0.767	0.9	0.667	0.867	0.833	0.767	**0.933**	**0.933**
BeetleFly	0.95	**1**	0.95	0.85	0.85	0.9	0.9	0.95	**1**
BirdChicken	0.9	0.95	0.95	0.95	0.9	0.9	0.9	1	**1**
Car	0.917	0.867	0.833	0.9	**0.999**	0.85	0.883	0.917	0.9
CBF	**1**	0.998	0.999	0.999	0.917	1	0.999	0.998	0.997
ChlorineConcentration	0.771	0.66	0.739	**0.876**	0.77	0.807	0.739	0.688	0.686
CinCECGTorso	**1**	0.981	0.995	0.853	0.87	0.837	**1**	0.991	0.991
Coffee	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Computers	0.764	0.7	0.776	**0.808**	0.684	0.752	0.736	0.804	**0.808**
CricketX	0.826	0.821	0.808	**0.849**	0.815	0.826	0.746	0.797	0.808
CricketY	0.851	0.8	0.823	0.844	0.826	**0.856**	0.805	0.751	0.8
CricketZ	**0.864**	0.831	0.823	0.849	0.823	0.859	0.81	0.802	0.81
DiatomSizeReduction	0.958	0.964	0.935	0.948	0.925	0.967	0.863	**0.987**	0.967
DistalPhalanxOutlineCorrect	0.775	0.757	0.772	0.786	0.779	0.761	0.786	0.797	**0.808**
DistalPhalanxOutlineAgeGroup	0.748	0.748	0.755	0.734	0.748	0.748	0.748	**0.806**	0.763
DistalPhalanxTW	0.705	0.676	0.676	0.655	0.683	**0.719**	0.698	0.712	0.683
Earthquakes	0.748	0.748	0.748	0.712	0.748	0.748	0.748	0.82	**0.83**
ECG200	0.89	0.84	0.86	0.91	0.91	0.91	0.89	**0.94**	**0.94**
ECG5000	**0.947**	0.946	**0.947**	0.942	0.945	**0.947**	0.943	0.945	**0.947**
ECGFiveDays	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
ElectricDevices	0.758	**0.76**	0.746	0.721	0.731	0.726	0.736	0.747	0.758
FaceAll	0.868	0.842	0.779	0.793	0.808	**0.947**	0.764	0.875	0.88
FaceFour	**1**	**1**	0.989	0.955	0.989	0.977	**1**	**1**	**1**
FacesUCR	0.963	0.967	0.959	**0.972**	0.958	0.961	0.907	0.958	0.957
FiftyWords	0.833	**0.848**	0.776	0.846	0.84	0.833	0.782	0.818	0.833
Fish	0.989	**0.994**	**0.994**	0.983	0.971	0.971	0.943	0.914	0.943
FordA	0.954	0.95	0.951	0.961	0.95	0.942	0.969	**0.971**	0.969
FordB	0.831	0.823	0.832	**0.849**	0.812	0.806	0.816	0.807	0.831

Table 4.2: The experiments of our PPSN and PPSN++ compared to 7 state-of-the-art methods.

Dataset	HIVE-COTE 2.0	TS-CHIEF	HIVE-COTE	InceptionTime	MiniRocket	Rocket	CIF	PPSN	PPSN++
GunPoint	**1**	**1**	**1**	**1**	0.993	0.993	0.993	0.993	**1**
Ham	0.695	0.705	0.705	0.724	0.714	0.714	0.733	**0.8**	0.79
HandOutlines	0.938	0.938	0.932	**0.959**	nan	0.943	0.919	0.932	0.938
Haptics	0.545	0.529	0.539	**0.571**	0.542	0.519	0.519	0.523	**0.571**
Herring	0.672	0.641	0.672	0.703	0.656	0.672	0.609	**0.75**	0.742
InlineSkate	**0.553**	0.527	0.485	0.485	0.478	0.475	0.549	0.427	0.475
InsectWingbeatSound	0.663	0.644	0.66	0.637	**0.669**	0.661	0.688	0.631	0.625
ItalyPowerDemand	0.972	0.965	0.96	0.966	0.969	0.969	0.973	**0.977**	0.97
LargeKitchenAppliances	**0.907**	0.768	0.853	0.904	0.864	0.904	0.824	0.859	0.904
Lightning2	0.787	**0.836**	0.77	**0.836**	0.738	0.754	0.77	0.82	**0.836**
Lightning7	**0.822**	0.767	0.726	0.795	0.795	**0.822**	0.753	0.795	0.808
Mallat	0.978	0.971	0.973	0.958	0.951	0.957	0.919	**0.982**	**0.982**
Meat	0.933	0.883	0.917	0.95	0.967	0.95	0.933	**1**	**1**
MedicalImages	**0.808**	0.796	0.732	0.796	0.795	0.793	0.789	0.768	0.79
MiddlePhalanxOutlineCorrect	**0.852**	0.825	0.835	0.835	**0.852**	0.838	0.832	0.839	0.84
MiddlePhalanxOutlineAgeGroup	0.597	0.571	0.591	0.552	0.61	0.591	0.61	**0.669**	0.571
MiddlePhalanxTW	0.558	0.565	0.584	0.532	0.539	0.565	0.584	0.612	**0.617**
MoteStrain	**0.97**	0.927	0.968	0.894	0.931	0.915	0.944	0.887	0.851
NonInvasiveFatalECGThorax1	0.947	0.917	0.924	**0.958**	nan	0.955	0.935	0.95	0.935
NonInvasiveFatalECGThorax2	**0.967**	0.948	0.945	0.959	nan	0.969	0.941	0.951	0.945
OliveOil	0.933	0.9	0.867	0.867	0.933	0.933	0.933	**0.967**	0.933
OSULeaf	0.963	**0.996**	0.983	0.921	0.959	0.942	0.86	0.967	0.97
PhalangesOutlinesCorrect	0.831	0.818	0.821	0.852	0.832	0.834	**0.837**	0.733	0.767
Phoneme	0.355	0.361	**0.381**	0.337	0.292	0.283	0.406	0.308	0.337
Plane	0.933	0.976	0.962	0.942	**1**	0.091	0.284	**1**	**1**
ProximalPhalanxOutlineCorrect	0.9	0.883	0.89	**0.931**	0.9	0.9	0.897	0.869	0.821
ProximalPhalanxOutlineAgeGroup	0.854	0.859	0.839	0.849	0.854	0.859	0.844	**0.883**	**0.883**
ProximalPhalanxTW	0.829	0.824	0.824	0.8	0.8	0.82	0.795	**0.834**	0.8
RefrigerationDevices	0.539	0.571	0.579	0.515	0.477	0.536	0.611	**0.621**	0.619
ScreenType	**0.579**	0.499	0.557	0.595	0.472	0.477	0.541	0.528	0.557

Table 4.3: The experiments of our PPSN and PPSN++ compared to 7 state-of-the-art methods.

Dataset	HIVE-COTE 2.0	TS-CHIEF	HIVE-COTE	InceptionTime	MiniRocket	Rocket	CIF	PPSN	PPSN++
ShapeletSim	**1**	**1**	**1**	0.994	**1**	**1**	0.994	0.961	**1**
ShapesAll	0.922	0.925	**0.932**	0.925	0.92	0.907	0.855	0.837	0.922
SmallKitchenAppliances	**0.837**	0.824	0.835	0.779	0.827	0.813	0.819	0.816	0.779
SonyAIBORobotSurface1	0.908	0.832	0.767	0.879	0.89	0.923	0.892	**0.965**	0.892
SonyAIBORobotSurface2	0.93	0.896	0.942	**0.952**	0.918	0.915	0.902	0.923	0.942
StarlightCurves	0.982	**0.983**	0.981	0.978	**0.983**	0.981	0.981	0.979	0.981
Strawberry	0.976	0.97	0.97	**0.984**	**0.984**	0.981	0.965	0.962	0.97
SwedishLeaf	0.965	0.965	0.954	**0.97**	0.963	0.963	0.962	0.958	0.954
Symbols	0.976	0.978	0.98	**0.985**	0.984	0.973	0.964	0.928	0.978
SyntheticControl	**1**	0.997	0.997	0.997	0.98	0.997	0.997	0.993	**1**
ToeSegmentation1	0.965	0.969	**0.974**	0.965	0.961	0.969	0.925	0.969	0.925
ToeSegmentation2	0.938	0.962	0.962	0.946	0.923	0.915	0.877	**0.969**	0.962
Trace	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
TwoLeadECG	**0.999**	0.994	0.998	0.996	0.998	**0.999**	0.997	0.995	0.994
TwoPatterns	**1**	**1**	0.999	**1**	0.996	**1**	**1**	0.999	**1**
UWaveGestureLibraryX	0.855	0.843	0.829	0.822	0.846	**0.858**	0.837	0.816	0.796
UWaveGestureLibraryY	**0.775**	0.771	0.747	0.771	**0.775**	0.773	0.765	0.738	0.747
UWaveGestureLibraryZ	**0.797**	0.786	0.777	0.769	0.796	0.794	0.781	0.744	0.769
UWaveGestureLibraryAll	0.974	0.97	0.968	0.951	0.971	**0.975**	0.973	0.962	0.951
Wafer	**1**	0.999	**1**	0.999	0.999	0.999	0.999	**1**	0.999
Wine	0.87	0.87	0.796	0.63	0.87	0.815	0.796	**0.944**	0.92
WordSynonyms	0.752	**0.795**	0.697	0.745	0.754	0.751	0.693	0.715	0.751
Worms	0.753	0.818	0.649	0.805	0.74	0.714	**0.766**	0.727	0.753
WormsTwoClass	0.792	**0.831**	0.766	0.766	0.792	0.74	0.779	0.792	0.792
Yoga	**0.934**	0.854	0.912	0.905	0.909	0.917	0.877	0.838	0.854
ACSF1	0.93	0.84	**0.94**	0.91	0.92	0.9	0.88	0.92	0.93
BME	**1**	0.993	0.953	**1**	**1**	**1**	**1**	**1**	**1**
Chinatown	0.983	0.968	0.98	0.985	0.983	0.98	**0.988**	0.983	0.985
Crop	0.765	0.764	0.773	**0.799**	0.749	0.751	0.782	0.751	0.765
EOGHorizontalSignal	**0.657**	0.655	0.611	0.63	0.599	0.652	0.619	0.588	0.652

Table 4.4: The experiments of our PPSN and PPSN++ compared to 7 state-of-the-art methods.

Dataset	HIVE-COTE 2.0	TS-CHIEF	HIVE-COTE	InceptionTime	MiniRocket	Rocket	CIF	PPSN	PPSN++
EOGVerticalSignal	0.569	**0.597**	0.547	0.517	0.528	0.533	0.586	0.506	0.528
EthanolLevel	0.696	0.528	0.684	**0.832**	0.598	0.57	0.598	0.778	0.758
FreezerRegularTrain	**1**	0.998	0.998	0.997	**1**	0.996	**1**	**1**	0.996
FreezerSmallTrain	0.999	0.998	0.986	0.846	0.965	0.95	**1**	0.998	0.95
GunPointAgeSpan	0.997	**1**	0.997	0.991	0.994	0.997	0.994	0.972	0.991
GunPointMaleVersusFemale	**1**	**1**	**1**	**1**	**1**	0.997	**1**	0.997	**1**
GunPointOldVersusYoung	**1**	**1**	**1**	**1**	**1**	0.994	1	0.974	0.974
HouseTwenty	0.966	0.975	**0.983**	0.916	0.966	0.966	0.916	0.933	0.916
InsectEPGRegularTrain	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
InsectEPGSmallTrain	**1**	**1**	**1**	**1**	0.992	0.984	**1**	**1**	**1**
MixedShapesRegularTrain	**0.976**	0.971	0.969	0.97	0.974	0.967	0.956	0.949	0.974
MixedShapesSmallTrain	**0.959**	0.949	0.954	0.911	0.955	0.938	0.921	0.896	0.911
PigAirwayPressure	0.933	**0.976**	0.962	0.942	0.87	0.091	0.284	0.966	0.962
PigArtPressure	0.995	0.981	0.99	**1**	0.986	0.952	0.904	0.98	0.952
PigCVP	**0.966**	0.962	0.962	**0.966**	0.952	0.938	0.649	0.952	0.938
PowerCons	0.983	0.989	**1**	0.994	0.989	0.933	0.994	**1**	0.994
Rock	0.92	0.9	**0.96**	0.6	0.8	0.9	0.88	**0.96**	0.9
SemgHandGenderCh2	0.957	0.923	**0.97**	0.873	0.897	0.918	0.942	0.955	0.918
SemgHandMovementCh2	0.856	**0.878**	0.871	0.584	0.691	0.638	0.856	0.744	0.734
SemgHandSubjectCh2	0.902	0.924	**0.953**	0.733	0.878	0.889	0.922	0.936	0.889
SmoothSubspace	0.987	**1**	0.98	0.98	0.947	0.98	0.98	0.953	0.947
UMD	0.993	0.986	0.979	0.986	0.993	0.993	0.979	**1**	0.979

Table 4.5: The experiments of our PPSN and PPSN++ compared to 7 state-of-the-art methods.

Chapter 5

Conclusion

In this dissertation, we have proposed two novel shapelet-based methods for time series classification.

The first one is a Perceptually Position-aware Shapelet Network (called PPSN) which consists of two phases. In the first phase, we propose an effective method for extracting shapelet candidates that involve utilizing perceptually important points and evaluating them through position-aware subsequence distance. In the second phase, we propose novel two techniques, namely fixed normalization and stop-gradient epochs, to overcome the negative effect of different subsequence distance ranges and the suboptimal weights of the final linear layer. Our experiments show that PPSN outperforms existing non-ensemble methods and is competitive with HIVE-COTE 2.0, the current most accurate classifier while preserving the advantages of interpretability and computational efficiency.

The second one is a stronger shapelet-based classifier which is built upon the PPSN framework (called PPSN++) with two main improvements. Firstly, we propose to use Distance Learning Layer to leverage important distances which are inadvertently ignored by previous methods, and secondly, a Binary Auxiliary Head has been added along with General Classification Head to capture distinctive patterns of individual classes. We demonstrate that our PPSN++ outperforms PPSN in terms of classification accuracy, making it become the state-of-the-art result in TSC tasks by conducting various experiments.

In future works, we tend to explore the potential of PPSN++ in addressing other time series issues, for example, the classification of multivariate time series data.

Bibliography

[1] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

[2] Gustavo EAPA Batista, Xiaoyue Wang, and Eamonn J Keogh. A complexity-invariant distance measure for time series. In *Proceedings of the 2011 SIAM international conference on data mining*, pages 699–710. SIAM, 2011.

[3] Andrew Butler, Paul Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature biotechnology*, 36(5):411–420, 2018.

[4] Fu Lai Korris Chung, Tak-Chung Fu, Wing Pong Robert Luk, and Vincent To Yee Ng. Flexible time series pattern matching based on perceptually important points. In *Workshop on Learning from Temporal and Spatial Data in International Joint Conference on Artificial Intelligence*, 2001.

[5] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The UCR time series classification archive, October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

[6] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

[7] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 248–257, 2021.

[8] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[9] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.

[10] Mojtaba A Farahani, MR McCormick, Robert Gianinny, Frank Hudacheck, Ramy Harik, Zhichao Liu, and Thorsten Wuest. Time-series pattern recognition in smart manufacturing systems: A literature review and ontology. *arXiv preprint arXiv:2301.12495*, 2023.

[11] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–401, 2014.

[12] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data mining and knowledge discovery*, 28:851–881, 2014.

[13] Shenda Hong, Meng Wu, Yuxi Zhou, Qingyun Wang, Junyuan Shang, Hongyan Li, and Junqing Xie. Encase: An ensemble classifier for ecg classification using expert features and deep neural networks. In *2017 Computing in cardiology (cinc)*, pages 1–4. IEEE, 2017.

[14] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

[15] Guozhong Li, Byron Koon Kau Choi, Jianliang Xu, Sourav S Bhowmick, Kwok-Pan Chun, and Grace LH Wong. Efficient shapelet discovery for time series classification. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[16] Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29:565–592, 2015.

[17] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–297, 2012.

[18] Jason Lines, Sarah Taylor, and Anthony Bagnall. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data*, 12(5), 2018.

[19] Qianli Ma, Wanqing Zhuang, and Garrison Cottrell. Triple-shapelet networks for time series classification. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1246–1251. IEEE, 2019.

[20] Qianli Ma, Wanqing Zhuang, Sen Li, Desen Huang, and Garrison Cottrell. Adversarial dynamic shapelet networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5069–5076, 2020.

[21] Matthew Middlehurst, James Large, Gavin Cawley, and Anthony Bagnall. The temporal dictionary ensemble (tde) classifier for time series classification. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*, pages 660–676. Springer, 2021.

[22] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. Hive-cote 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110(11):3211–3243, 2021.

[23] Thanawin Rakthanmanon and Eamonn Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *proceedings of the 2013 SIAM International Conference on Data Mining*, pages 668–676. SIAM, 2013.

[24] Peter L Rousseau and Dadanee Vuthipadadorn. Finance, investment, and growth: Time series evidence from 10 asian economies. *Journal of macroeconomics*, 27(1):87–106, 2005.

[25] Patrick Schäfer. Scalable time series classification. *Data Mining and Knowledge Discovery*, 30(5):1273–1298, 2016.

[26] Ahmed Shifaz, Charlotte Pelletier, François Petitjean, and Geoffrey I Webb. Ts-chief: a scalable and accurate forest algorithm for time series classification. *Data Mining and Knowledge Discovery*, 34(3):742–775, 2020.

[27] Lexiang Ye and Eamonn 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*, pages 947–956, 2009.

[28] Hanbo Zhang, Peng Wang, Zicheng Fang, Zeyu Wang, and Wei Wang. Elis++: a shapelet learning approach for accurate and efficient time series classification. *World Wide Web*, 24(2):511–539, 2021.

Publications

Top-teir Conferences

Xuan-May Le, Minh-Tuan Tran, Van-Nam Huynh. *Learning Perceptual Position-Aware Shapelets for Time Series Classification.* In Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part VI (pp. 53-69). Cham: Springer Nature Switzerland, 2023.

International Journals

Minh-Tuan Tran *, **Xuan-May Le** *, Sung-Eui Yoon, Van-Nam Huynh. *PISD: A Linear Complexity Distance beats Dynamic Time Warping on Time Series Classification and Clustering.* Pattern Recognition. (Under review)

Xuan-May Le, Minh-Tuan Tran, Van-Nam Huynh. *PPSN++: A Stronger Shapelet-based Time Series Classifier using Distance Learning and Binary Auxiliary Head. Data Mining and Knowledge Discovery (DMKD).* (Under review)

*These authors contributed equally to this work.