A scalable shapelet discovery for time series classification

Mr. Nattakit Vichit

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2018

การค้นพบเชพเลทอย่างรวดเร็วสำหรับการจำแนกอนุกรมเวลา

นายณัฐกิตติ์ วิชิต

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2561

Thesis Title	A scalable shapelet discovery for time series classification
By	Mr. Nattakit Vichit
Field of Study	Computer Science
Thesis Advisor	Associate Professor CHOTIRAT RATANAMAHATANA, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Science

..... Dean of the Faculty of Engineering
(Professor SUPOT TEACHAVORASINSKUN, Ph.D.)

THESIS COMMITTEE

..... Chairman
(Duangdao Wichadakul, Ph.D.)

..... Thesis Advisor
(Associate Professor CHOTIRAT RATANAMAHATANA, Ph.D.)

..... External Examiner
(Haemwaan Sivaraks, Ph.D.)

ณัฐกิตติ์ วิชิต : การค้นพบเชพเลทอย่างรวดเร็วสำหรับการจำแนกอนุกรมเวลา. (A scalable shapelet discovery for time series classification) อ.ที่ปรึกษาหลัก : รศ. ดร.โชติรัตน์ รัตนามหัทธนะ

เนื่องจากข้อมูลอนุกรมเวลามีความซับซ้อนเพิ่มขึ้นและผู้ใช้คาดหวังประโยชน์จากการวิเคราะห์มากขึ้น อัลกอรึทึมทั้งหลายจึงถูกนำเสนอเพื่อนำมาแก้ปัญหา อัลกอริทึมเชพเล็ทหรือการจำแนกส่วนของอนุกรมเวลาเป็นหนึ่งในอัลกอรึทึมที่สามารถจำแนกอนุกรมเวลาที่ให้ผลลัพธ์ที่ดีทั้งในด้านของความแม่นยำและสามารถมอบข้อมูลเชิงลึกที่สามารถนำไปประยุกต์ใช้งานจริงได้ แต่ทั้งนี้ ในอดีตที่ผ่านมาอัลกอริทึมเชพเล็ทยังประสพปัญหาเนื่องจากความต้องการเวลาประมวลผลมากจึงมีข้อจำกัดในการใช้งานกับชุดข้อมูลทีมีขนาดใหญ่ จากปัญหาดังกล่าววิทยานิพนธ์นี้จึงเสนออัลกอริทึมเชพเลทตัวใหม่ ซึ่งจะปรับปรุงเรื่องความรวดเร็วในการประมวลผลเป็นหลัก อัลกอริทึมที่นำเสนอจะใช้ชื่อว่า Dual Increment Shapelets (DIS) เกิดจากการรวมกันของ Incremental neural network สองชั้นและกระบวนการคัดเลือกผลลัพธ์จากลักษณะส่วนอนุกรมเวลา ผลลัพธ์จากการทดลองเชิงประจักษ์ใน 40 ชุดข้อมูลแสดงให้เห็นว่า อัลกอริทึมนี้มีความเร็วในการประมวลผลเพิ่มขึ้นอย่างมากอีกทั้งสามารถคงความแม่นยำไว้ในระดับสูง อัลกอริทึมที่ถูกนำเสนอที่ผ่านมาจะเน้นด้านการปรับปรุงความเร็วของกระบวนการค้นหาเชพเลทเป็นหลัก แต่ในวิทยานิพนธ์นี้จะใช้กระบวนการลดจำนวนตัวเลือกจากลักษณะส่วนอนุกรมเวลาแทน ผลการทดลองแสดงให้เห็นว่า อัลกอริทึมนี้จะสามารถเพิ่มประสิทธิภาพในด้านความเร็วมากกว่า 1,000 เท่าเมื่อเปรียบเทียบกับอัลกอริทึมพื้นฐานอีกทั้งสามารถคงความแม่นยำไว้ได้

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่อนิสิต .....
ปีการศึกษา 2561 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6070178221 : MAJOR COMPUTER SCIENCE

KEYWORD: Time series, Data mining, Classification, Shapelet

Nattakit Vichit : A scalable shapelet discovery for time series classification. Advisor: Assoc. Prof. CHOTIRAT RATANAMAHATANA, Ph.D.

As time series data become more complex and users expect more sophisticated information, numerous algorithms have been proposed to solve these challenges. Among those algorithms to classify time series data, shapelet – a discriminative subsequence of time series data – is considered a practical approach due to its accurate and insightful classification. However, previously proposed shapelet algorithms still suffer from exceedingly high computational complexity, as a result, limiting its scalability to larger datasets. Therefore, in this work propose a novel algorithm that speeds up shapelet discovery process. The algorithm so called “Dual Increment Shapelets (DIS)” is a combination of two-layered incremental neural network and filtering process based on subsequence characteristics. Empirical experiments on forty datasets evidently demonstrate that the proposed work could achieve large speedup while maintaining its accuracy. Unlike the previous algorithm that mainly emphasizes speedup of the search algorithm, DIS essentially reduces the number of shapelet candidates based on subsequence characteristics. As a result, The DIS algorithm could achieve more than three orders of magnitude speedup, comparing with the baseline algorithms, while preserving the accuracy of the state-of-the-art algorithm.

Field of Study:	Computer Science	Student's Signature .....
Academic Year:	2018	Advisor's Signature .....

ACKNOWLEDGEMENTS

First of all, I would like to thank Associate Professor Chotirat Ratanamahatana. She is the best thesis advisor that I have met. She give me many challenges and constructive critics. Without her advice the thesis could not become successful. I want to say that she makes me a better person. I am feeling to sincerely thanks to Duangdao Wichadakul, Ph.D. and Haemwaan Sivaraks, Ph.D for being my thesis committee. I am also so thankful to my fellow labs researcher in Mind lab for giving me many advice about this research. And teach me how to be a better researcher. Many knowledge and idea are came from classroom which I have registered. I want to thank all of professors that giving me an broad overview of many topic in computer science. I really like to listen to them. Furthermore, I also thank Chulalongkorn University for gives me an opportunity to study in master degree in computer science. Lastly, I also thank my family who support me throughout the time of my research.

Nattakit Vichit

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Chapter 1

Introduction

A time series is a sequence of data point, which represents data in a time interval. Apart from classical time series data such as stock market data, other types of data have recently been shown to work effectively and efficiently for various data mining tasks once transformed into time series data, including shape classification, movement tracking, medical diagnosis (ECG/EKG, EEG, etc.), motif discovery, anomaly detection, classification, clustering, etc.

In working with time series data, many challenges that affect the performance of the time series mining tasks include occlusion, distortion of warping, uniform scaling, uncertainty, and wandering baselines (Keogh et al., 2009). Therefore, the algorithms must be designed to overcome the complexity of a time series with such variances.

A large amount of studies has been conducted to improve time series classification tasks. A well-known approach is based on distance, such as Euclidean Distance (ED) and Dynamic Time Warping (DTW) combined with 1-NN (Keogh & Ratanamahatana, 2005). However, these algorithms have been shown to be sensitive to missing data that may lead to misclassification. To solve this problem, many researchers opted to classify time series data by discovering and using local features rather than the whole time series sequence, e.g., interval-based classifiers, dictionary-based classifiers, and shapelet-based classifiers.

Above all three aforementioned algorithms, a shapelet-based classifiers have been recently shown to provide the most accurate and interpretable results with the speediest testing. Due to its flexibility, shapelet-based classifiers are applicable for both local features and whole time series. This flexible quality allows extensive improvement of the classification accuracy. Comparing with lazy algorithm, such as DWT based 1-NN classifier, eager algorithms like shapelet-based classifiers could save more time to classify the target classes. Moreover, shapelet-based classifiers provide more interpretable results, which facilitate the domain experts in the fields to effectively interpret the results. For example, the shapelets that are generated by the shapelet-based classifiers could evidently provide prominent features of '*Urtica Dioica*' and '*Verbena Urticifolia'* (Rakthanmanon & Keogh, 2013). Shapelet algorithm will select the most discriminative feature such as in Figure 1; important feature are located at in the tweek of each class. As a result, shapelets will locate around those tweeks. Based on these shapelet-based classifier

results, the botanists could identify the types of the skull by having only a quick glance of the shapelets.

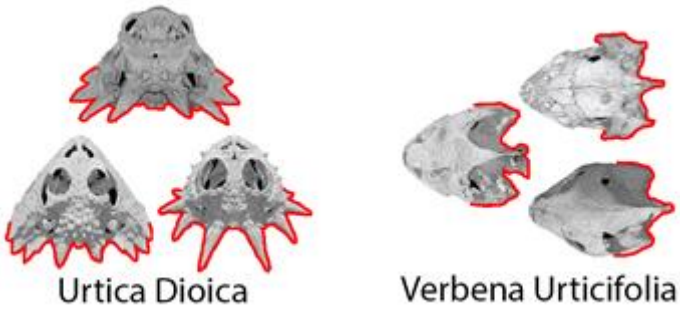

Figure 1 Difference of position of important features between Uritca Dioica class and Verbena Urticifolia class as shown in bold lines.

Another application of shapelet-based classifiers are classification of video clips of actions; In Figure 2, a hand's centroid position in each frame is tracked and transformed into a time series. The main difference of the first half of the action between two classes are the position of the hand to pick up or not to pick up the gun. Figure 3 compares time series sequences of a 150-frame video clip between the actions of a person grasping a gun from a holster, pointing it to a target, then putting it back to the holster versus another person resting his hand on the side, only pointing his finger to a target, then putting his hand back to the resting position. As seen in the figure, the $100^{th} - 130^{th}$ time interval clearly shows the difference between the person holding a gun and the one without. Even though the two time series sequences demonstrate the discrepancies between the two actions in the $50^{th} - 90^{th}$ time interval (different heights of the hand pointing a finger vs. pointing a gun), the shapelet-based classifiers see them as irrelevant invariances and successfully differentiate the two actions using only the essential intervals. It can be observed that the overly captured invariances could mislead the results. Thus, shapelet-based classifiers are considered more appropriate and effective.

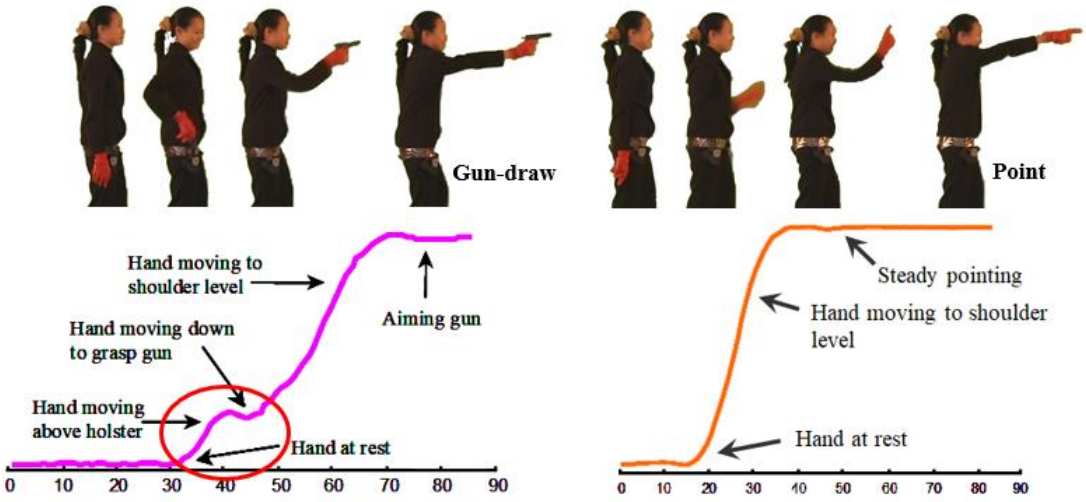

Figure 2 A comparison between time series sequences that created from motion of the actions in the video of a person holding and the pointing a gun (Gun time series) and a person pointing a finger (No gun time series).

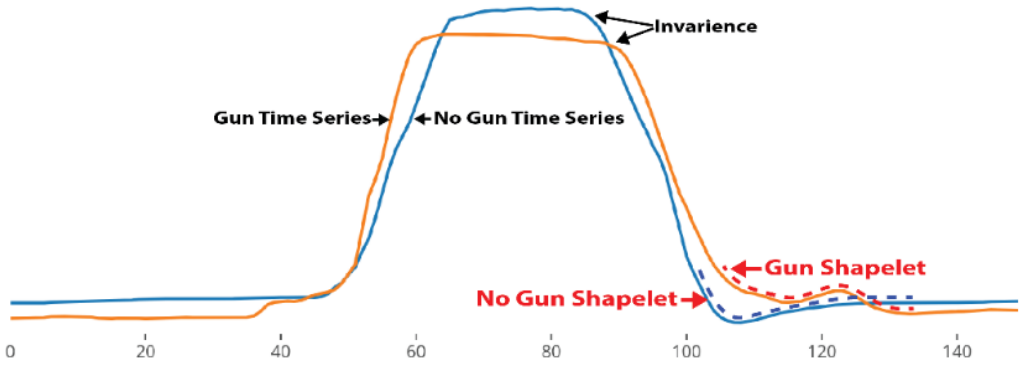

Figure 3 A comparison between time series sequences of the actions of a person holding and then *pointing a gun (Gun time series) and a person pointing a finger (No gun time series) and shapelets, which represent gun time series and no gun time series.*

In spite of its benefits, shapelet-based classifiers still suffer from a major drawback; they are too slow that they become infeasible with large datasets. Even in state-of-the-art approach, the algorithm still requires to generate a large number of shapelet candidates. A group of shapelet candidates could be as large as $O(n^2 m^4)$ when n is the number of time series in the dataset, and m is the length of each time series. For instance, if a dataset contains only 70 instances, where each instance is 500 data points long, the shapelet candidates could accumulate to more than

10^{14} candidates, which become infeasible for the classifier to give prediction results within reasonable amount of time unless adequate number of candidates are pruned out.

To resolve the issues of infeasibility and practicality of the existing shapelet-based classifiers, I have designed a new algorithm, which optimizes the shapelet candidate selection process, while preserving the accuracy of the results and no-false-dismissal property. More specifically, proposed algorithm can effectively prune out shapelet candidates using a so-called "Dual Incremental Shapelets (DIS)" that is a combination of two-layered incremental neural network and filtering process based on subsequence characteristics.

1.1 Objective

The objective of this thesis is to design the a new shapelet algorithm that is faster in terms of the running time but also comparable in terms of accuracy when compared to the best-known shapelet baselines.

1.2 Scope

The datasets employed in this thesis will be the 40 datasets taken from UCR 2015 repository (Chen et al., 2015). The proposed algorithm will measure the performance by comparing the algorithm to the 3 baselines: fast shapelet, shapelet transform, and learning shapelet.

1.3 Expected benefit of this thesis

This work aims to create a faster shapelet algorithm which maintains the comparable level of accuracy to the 3 baselines. The proposed shapelet algorithm is expected to find the shapelet in the datasets in a feasible time frame. This ability is expected to improve the weaknesses of the 3 mentioned baselines.

The rest of the document is organized as follows. Chapter 2 gives fundamental background on shapelets and reviews on its related work. Chapter 3 presents the research methodology and the algorithm details. Chapter 4 presents the experiment results. Chapter 5 presents the conclusion of this thesis.

Chapter 2

Literature review

2.1 Terminology

Time Series: Time series (*T*) is a sequence of data points, which represent data in a time order. Every data point in a time series sequence is a real value. The notation of a time series sequence is defined as follows:

$$T = \{t_1, t_2, \ldots, t_l\} \tag{2.1}$$

where *l* is the length of the time series.

Time Series Subsequence: Subsequences can be selected from the time series. The length of subsequences (m) can vary from 3 to the length *l*. The starting position of selection (*p*) can vary from 0 to *l*-*m*+1. The notation of a subsequences is defined as follows:

$$S = \{t_p, t_{p+1}, \ldots, t_{p+m-1}\} \tag{2.2}$$

Distance between two time series: In this work, the distances between two time series are $T = \{t_0, t_1, \ldots, t_m\}$ and $R = \{r_0, r_1, \ldots, r_m\}$ where *T* and *R* are equal in length. The distance is calculated by Euclidean distance as presented below:

$$Dist(T,R) = \sum_{i=1}^{m} (t_i - r_i)^2 \tag{2.3}$$

Distance from the time series to the subsequence: SubsequenceDist(*T*, *S*) is a distance between time series $T = \{t_0, t_1, \ldots, t_l\}$ and subsequence $S = \{s_0, s_1, \ldots, s_m\}$, where subsequence *T* has length *l,* and subsequence *S* has length *m.* One by one, algorithm will calculate the length of each subsequence in order to locate the minimum distance, which will be later marked as SubsequenceDist. The notation of SubsequenceDist is defined as follows:

$$SubsequenceDist(T,S) = min(Dist(s, s^*)) \tag{2.4}$$

Entropy: Given that there are two classes, namely Class A and Class B, in the dataset *D,* a proportion of classes will be referred as *p*(*A*) and *p*(*B*), respectively. The entropy (*I*) will be obtained using the following formula:

$$I(D) = -(p(A)\log p(A) + p(B)\log p(B)) \tag{2.5}$$

Information gain: To examine the information gain (Gain(*D*)), which is the difference between the entropy before and after splitting dataset *D*, the dataset *D* will be split into two groups. Then, the entropy of the two groups will be calculated. After that, the difference between two entropy values will be calculated. The more information gain at the split point is, the better split point quality it will be. The notation of an information gain is defined as follows:

$$Gain(D) = I(D) - (f(D1)I(D1) + f(D2)I(D2)) \tag{2.6}$$

where *f* is a proportion of the number of data which is split to each side.

Optimal split distance: Optimal split distance is the value of information gain at the best split point in which the information gain reaches the highest value. The notation of an optimal split distance is defined as follows:

$$Gain(S,d) \geq Gain(S,d_i) \tag{2.7}$$

for all possible d_i.

Shapelet: Shapelet is a subsequence which obtains the minimum distance among all time series sequences in a dataset, as indicated by the result of subsequenceDist. In other words, shapelet is the subsequence that best discriminates classes. The notation of a shapelet is defined as follows:

$$Gain(S,d) \geq Gain(S',d) \tag{2.8}$$

where S' is any subsequence that can be generated by a time series in a dataset.

2.2 Original shapelet

Shapelet was first introduced by Keogh et al. (Ye & Keogh, 2009), who suggested that shapelet could be obtained by scanning every subsequence to figure out the best subsequence among the time series. To do so, (Ye & Keogh, 2009) suggested the information gain, which is capable of separating classes at a certain split point (See Figure 4), should be used as a criterion. As a result, the subsequence with the best quality could be retrieved. As shown in Figure 4, there is a set of two different data classes (as represented by the circles and squares). The split point on the line effectively classifies the set of mixed data into two classes: the left group containing only circles, and another containing mostly squares with only one circle. Thus, it can be concluded that the group of data on the left side is better than the one on the right.

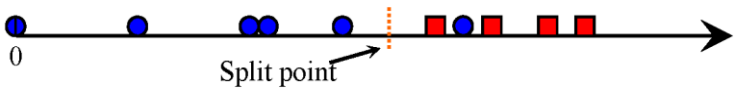

Figure 4 Quality of each side when data have been split at the split point (Rakthanmanon & Keogh, 2013)

The lengths of shapelet can be varied from 3 to the entire length of the time series sequence. With this variety, the number of shapelet candidates in the dataset can be massive. The number of shapelet candidates can be calculated using $O(n^2m^4)$ formula, where *n* is the number of time series in the dataset, and *m* is the length of a time series. It can be observed that the original shapelet algorithm introduced by (Ye & Keogh, 2009) runs very slowly because the quality of each subsequence is calculated one by one. The pseudo code of (Ye & Keogh, 2009) is presented in Table 1.

Table 1 A pseudo code of original shapelet algorithm (Ye & Keogh, 2009)

Algorithm 1 Original Shapelet

Input *Dataset*

1: $candidateShapelets \leftarrow \text{SelectAllCandidates}(Dataset)$
2: $maxGain \leftarrow 0.0$
3: $shapelet \leftarrow \text{null}$
4: **for all** *candidate* in *candidateShapelets* **do**
5: $\quad candidateGain \leftarrow \text{EstimateCandidate}(candidate, Dataset)$
6: $\quad$ **if** $candidateGain > maxGain$ **then**
7: $\quad\quad candidateGain \leftarrow maxGain$
8: $\quad\quad shapelet \leftarrow candidate$

return *shapelet*

First, all shapelet candidates of every length are generated (Line 1) as shown in Figure 5. Then, the subsequence with a better quality is identified and marked as a temporary shapelet (Lines 2 and 3). Then, the algorithm loops through every shapelet candidate to see whether there is another subsequence with better quality to the temporary shapelet. The quality of each shapelet candidate is estimated by the information gain (Line 5). If the better one is detected, it will replace the current temporary shapelet and become a new temporary shapelet (Lines 6-8). When it comes to the last comparison, the last temporary shapelet standing will be identified as a shapelet.

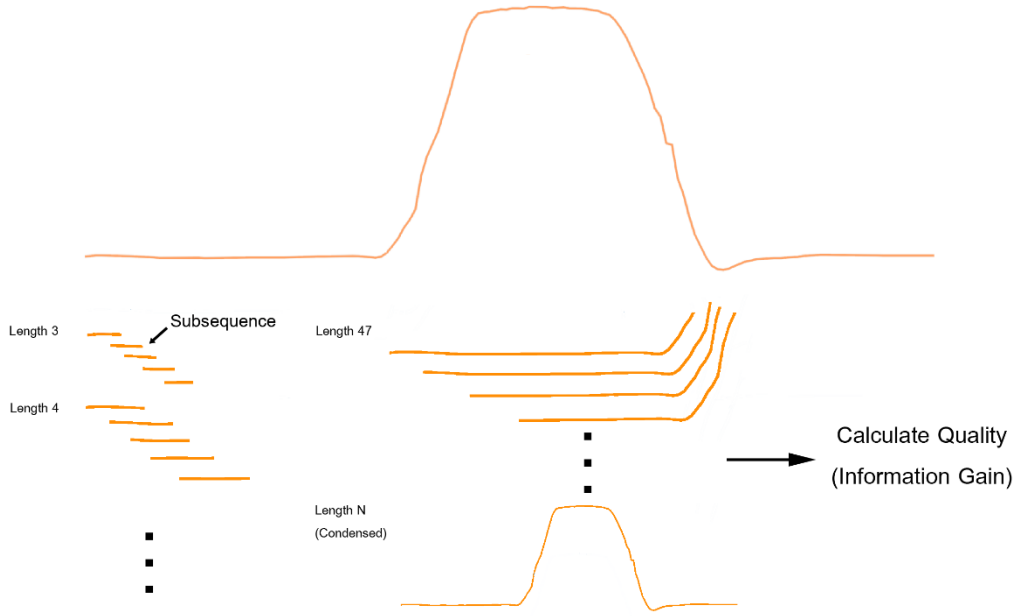

Figure 5 An example of subsequence generation process in the original shapelet algorithm.

To sum up, the algorithm starts by comparing the quality of one subsequence against another to get a better subsequence. After the better one is obtained, it is used as a baseline to compare with the next one. By repeating this process, the algorithm will eventually obtain the subsequence with the best quality.

2.3 Fast Shapelet (FS)

Fast Shapelet was introduced in 2012 by (Rakthanmanon & Keogh, 2013), which mainly improved the speed of algorithm in finding the best shapelet candidates. To do so, they suggested that the algorithm must preprocess the time series in order to roughly eliminate the low quality shapelet candidates and select the prospective shapelet candidates. The preprocessing consisted of Symbolic Aggregate Approximation (SAX) (Lin, Keogh, Wei, & Lonardi, 2007) combined with Random Projection (Bingham & Mannila, 2001). The pseudo code of (Rakthanmanon & Keogh, 2013) is presented in Table 2.

Table 2 A pseudo code of fast shapelet algorithm (Rakthanmanon & Keogh, 2013)

Algorithm 2 Fast Shapelet

Input *Dataset*

1: $[TS,Label] \leftarrow \text{ReadData}(Dataset)$
2: **for all** $len \leftarrow 1$ to m **do**
3: $SAXList \leftarrow \text{CreateSAXList}(TS,len)$
4: $Score \leftarrow [\,]$
5: **for all** $i \leftarrow 1$ to r **do**
6: $Count \leftarrow \text{RandProjection}(SAXList,TS)$
7: $Score \leftarrow \text{UpdateScore}(Score,Count)$
8: $SAXCand \leftarrow \text{FindTopKSAX}(SList,Score,k,r)$
9: $TSCand \leftarrow \text{Remap}(SAXCand,TS)$
10: $maxGain \leftarrow info$
11: $minGap \leftarrow 0$
12: **for all** $i \leftarrow 1$ to $|TSCand|$ **do**
13: $cand \leftarrow \text{TSCand}(i)$
14: $DList \leftarrow \text{NearestNeighbor}(TS,cand)$
15: $[gain,gap] \leftarrow \text{CalInfoGain}(DList)$
16: **if** $(gain > maxGain)$ || $((gain \leftarrow maxGain)$ && $(gap > minGap))$ **then**
17: $maxGain \leftarrow gain$
18: $minGap \leftarrow gap$
19: $shapelet \leftarrow cand$

return *shapelet*

The process of Fast Shapelet algorithm consists of 4 steps. First, the dimensions of the time series are reduced through SAX representation. This reduces the length of the time series and keeps the prospective shapelet candidates appended to the list (Line 3). Second, the shapelet candidates from Line 3 are randomly projected among the groups, using their similar shapes as a criterion (Line 6). Simultaneously, the score of each group is calculated using class discrimination power (Line 7). This process is repeatedly conducted until the last shapelet candidates in the list are projected into their groups. Third, the top *k* groups sorted by the class discrimination power are obtained (Line 9). Every shapelet candidate in the top *k* groups is remapped to the original time series (Line 10). Fourth, the quality of all remapped shapelet candidates from Line 10 is examined using the same process as the original shapelet algorithm. Finally, the best shapelet candidate among the remapped ones will be identified as the shapelet.

2.4 Shapelet Transform (ST)

Shapelet Transform was introduced by (Hills, Lines, Baranauskas, Mapp, & Bagnall, 2014) who noted the problem of using information gain as a criterion for measuring shapelet quality. According to (Hills et al., 2014), the original shapelet algorithm was time-consuming because it

examines the quality of shapelet candidates by recursively calculating the quality of shapelet candidates in every split point. Even though the algorithm processes only one single quality measurement, it is time consuming due to the repetition. Therefore, (Hills et al., 2014) rejected the use of information gain and opted to other quality measurements. Shapelet Transform, as introduced by (Hills et al., 2014), distinguishes classifiers from shapelet discovery process by transforming the time series into distance features and then using any classifier to classify the distance feature. By using this strategy, the decision tree classifiers can be replaced by any superior classifiers, for example, SVM and rotation forest. Elimination of decision tree leads to the absence of the information gain, which is used as shapelet quality measurement; therefore, the new practical quality measurement must be created.

(Hills et al., 2014) suggested F-stat, mood's-median, or Kruskal-Wallis as quality measurements. With F-stat quality measurement combined with SVM classifier, shapelet candidate discovery achieves about an order of magnitude speedup.

Table 3 A pseudo code of shapelet selection section in shapelet transform algorithm (Hills et al., 2014)

Algorithm 3 Shapelet Selection

Input T, min, max, k

1: $kShapelets \leftarrow [\,]$
2: **for all** Ti in T **do**
3: $shapelets \leftarrow [\,]$
4: **for all** $l \leftarrow min$ to max **do**
5: $W_{il} \leftarrow \text{generateCandidates}(T_i, l)$
6: **for all** subsequence S in W_{il} **do**
7: $D_s \leftarrow \text{findDistances}(S, T)$
8: $\text{quality} \leftarrow \text{assessCandidate}(S, D_S)$
9: $\text{shapelets.add}(S, quality)$
10: $\text{sortByQuality}(shapelets)$
11: $\text{removeSelfSimilar}(shapelets)$
12: $kShapelets \leftarrow \text{merge}(k, kShapelets, shapelets)$

return $kShapelets$

The process of Shapelet selection algorithm that presented in Table 3 can be describe shortly by these steps. First, all shapelet candidates are generated and measured for their quality. Then, they are sorted by their quality, and the similar shapelet candidates are removed. Finally, *k* shapelets are selected. The number of *k* are defined by $\frac{n}{2}$, where *n* is the length of the time series.

After the *k* shapelets are identified, they are transformed to the distance features using the following algorithm presenting in Table4.

Table 4 A pseudo code of shapelet transform section in shapelet transform algorithm (Hills et al., 2014)

Algorithm 4 Shapelet Transform

Input Shapelets S,Dataset T

1: $T' \leftarrow [\,]$
2: **for all** T_i in T **do**
3: **for all** shapelets $s_j$in S **do**
4: $dist \leftarrow \text{subsequenceDist}(s_j, T_i)$
5: $T_{ij} = dist$
6: $T' = T' \cup T_i$

return T'

First, the *k* shapelets are compared against all time series in the dataset (Line 4). After that, the distance features obtained from the previous process are classified by any classifier: SVM, Rotation forest, or Naïve Bayes.

Referring to Algorithm 3, the similar shapelet candidates are clustered. The representative of each group is nominated so the number of the shapelet candidates is reduced by using the algorithm presented in Table 5.

In conclusion, involving every similar shapelet candidate in the computation is considered redundant and a waste of time. As observed from the dataset, the shapelet candidates which have similar quality tend to cluster together. Thus, finding shapelet candidates should be done in cluster based on their similarity, rather than using the one-by-one process.

2.5 Self-Organizing Incremental Neural Network (SOINN)

Self-Organizing Incremental Neural Network was introduced by (Okada & Hasegawa, 2008). The outstanding feature of this algorithm is an ability to cluster all time series in the dataset in one pass, and to self-adjust the number of clusters by the distance of the data. So, this algorithm achieves satisfied speed, although it clusters a large dataset. In addition, it can remove the predefined cluster number parameter. The overview process of SOINN is presented in Figure 6.

Table 5 A pseudo code of cluster shapelet section in shapelet transform algorithm (Hills et al., 2014)

Algorithm 5 ClusterShapelets(Shapelets S, $noClusters$)

$C \leftarrow \emptyset$ *//C is a set of sets.*
for all $s_i \in S$ **do**
  $C \leftarrow C \cup \{s_i\}$
while $|C| > noClusters$ **do**
  $\mathbf{M} \leftarrow |C| \times |C|$ matrix
  for all $C_i \in C$ **do**
    for all $C_j \in C$ **do**
      $distance \leftarrow 0$
      $comparisons \leftarrow |C_i| \times |C_j|$
      for all $c_l \in C_i$ **do**
        for all $c_k \in C_j$ **do**
          $dist \leftarrow dist + d_S(c_l, c_k)$
      $\mathbf{M}_{i,j} \leftarrow \frac{dist}{comparisons}$ *//store average linkage distances in distance map.*
  $best \leftarrow \infty$
  $position \leftarrow \{0, 0\}$
  for all $\mathbf{M}_{i,j} \in \mathbf{M}$ **do**
    if $\mathbf{M}_{i,j} < best \wedge i \neq j$ **then**
      $x \leftarrow i$
      $y \leftarrow j$
      $best \leftarrow \mathbf{M}_{i,j}$ *//Find smallest distance in distance map.*
  $C' \leftarrow C_x \cup C_y$
  $C \leftarrow C - \{C_x\} - \{C_y\}$
  $C \leftarrow C \cup C'$ *//Update set of clusters, merging the closest pair.*
return C

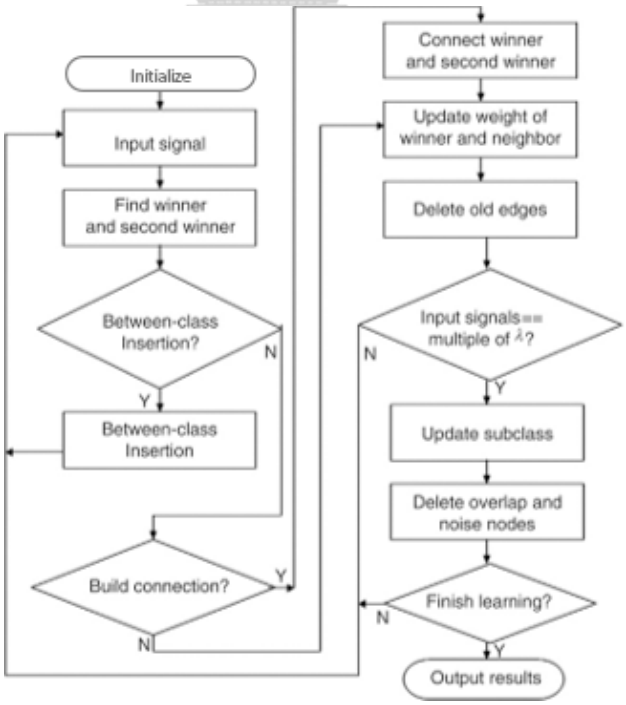

Figure 6 A process flow of Self-Organizing incremental neural network algorithm (Okada & Hasegawa, 2008).

First, the process starts with an empty cluster in a neural network. Next, the data in the dataset is passed one-by-one into a neural network. As the data is being fed in, the neural network finds the nearest preexisting cluster in the neural network. In case that the distance of the newly fed data is below the threshold, it will be merged with the preexisting group and the weight of the neural network is automatically self-adjusted. Vice versa, the neural network will automatically create a new group if the distance of the newly fed data is higher than the threshold. Eventually, the cluster(s) can be extracted from the neural network after all data is processed.

2.6 Piecewise Linear Representation (PLR)

When consider the subsequences in the dataset, it can observe the number of shapelet candidates. However, not every shapelet candidate is useful. Therefore, some processes must be created in order to filter the useless shapelet candidates out of the dataset. Hence, dimension reduction was applied upon the idea that some points of the time series can be removed. Despite its shortened length, the time series still preserves its main structure as well as achieves faster classification.

(Zhu, Wu, & Li, 2007) created the bottom-up Piecewise Linear Representation (PLR) by selecting the points in the time series that preserve the reconstruction error. First, the algorithm spots the starting and ending points and marks them as the first two points. Second, other points which are not yet spotted are counted in the evaluation as they are prescreened. As shown in figure 7, those candidates are used to create new time series candidates, which are later used to measure against the original time series. This process is repeatedly performed until the errors are below the threshold.

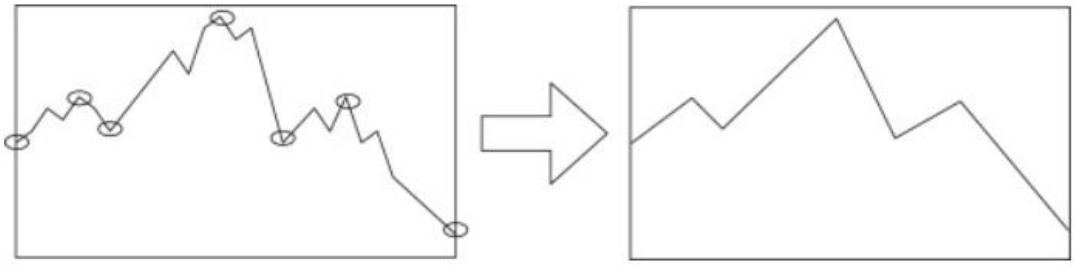

Figure 7 A time series constructed from the selected points (Zhu et al., 2007).

2.7 Local Farthest Deviation Points (LFDP)

LFDP was introduced by (Ji et al., 2016). This algorithm solves the errors found in PLR which are the result of the extreme points. This algorithm tends to deal with the time series that contain

extreme points in a special way. Despite its similarity to PLR, LFDP counts the extreme points into its special measurement.

Figure 8 The time series where its dimensions are reduced by LFDP(Ji et al., 2016).

As seen in Figure 8, the time series are concluded into 3 points, as spotted at the indices 1, 10, and 15. The resulting time series sequence is obtained by connecting these 3 points. The full details of this algorithm will be described in chapter 3.

2.8 Fast Shapelet Selection (FSS)

Employing the points from LFDP, Fast Shapelet Selection (FSS) generates shapelet candidates [11]. From the pool of the reference points from LFDP, FSS marks the starting and ending points of the selected shapelet candidates. The intuition of shapelet candidate generating process is presented in Figure 9.

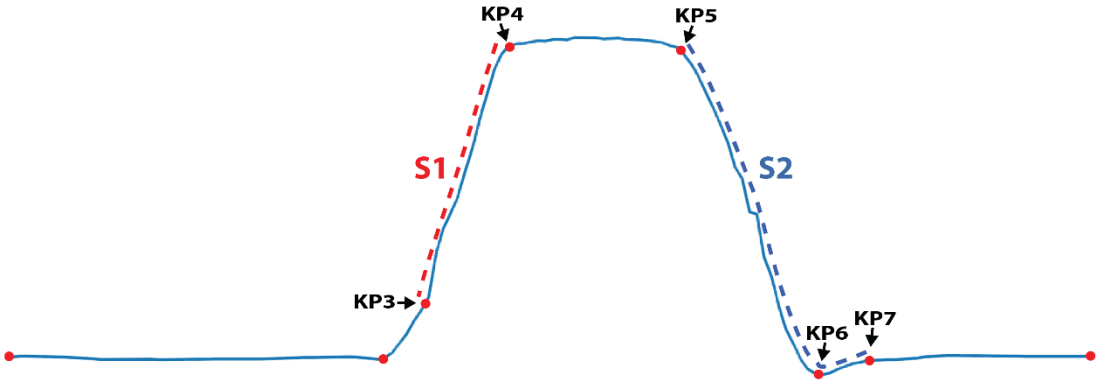

Figure 9 Examples of shapelet candidates generated by key points. Candidate S1 is generated from key point 3 (KP3) to key point 4 (KP4), and candidate (S2) is generated from key point 5 (KP5) to key point 7 (KP7).

The left dashed line (S1) is selected using the key point 3 (KP3) and 4 (KP4) as starting and ending points, respectively. The right dashed line (S2) is selected using key point 5 (KP5) as a starting point and key point 7 (KP7) as the ending one. This way, algorithm can prune the useless shapelet candidates out of the calculation.

2.9 Learning Shapelet (LS)

Learning shapelet (Grabocka, Schilling, Wistuba, & Schmidt-Thieme, 2014) is a shapelet selection algorithm which selecting the shapelet by learning the candidate score which is ability to separate class. Instead of calculating candidate quality one by one, this algorithm iteratively adjusts the weight of each candidate by a tiny fraction. Overview of this algorithm step can be described in these 3 steps. First algorithm initializes all the weight to all shapelet candidate. Second, most contributed subsequences are learned by adjust the weight parameter using gradient descent algorithm. Finally, best contribute subsequence will be selected as a shapelet.

Chapter 3

Research Methodology

3.1 Proposed method: Dual Increment Shapelet (DIS)

From literature review, the main reason why shapelet algorithms are time consuming is the enormous number of shapelet candidates, that shapelet algorithms generating. Thus, trying to prune out some useless shapelet candidates will improve the running time of the algorithm. This strategy will be used in the proposed algorithm, and the results are very promising.

Proposed algorithm is divided into 3 steps as follows:

3.1.1 Shapelet candidate selection

Not all shapelet candidates have the same quality. A Candidate with large variation is considered having higher quality than a candidate with small variation (e.g., straight line). The proposed method adopt a Local Farthest Deviation Points (LFDP) algorithm (Ji et al., 2016) to select only candidates with high variation. The pseudo code of LFDP algorithm are present in Table 6. For each time series sequence, LFDP algorithm first marks the start and end of the set of the selected points and then set the distance to infinity (lines 2-4). LFDP algorithm continues selecting a point until its distance to the original time series falls below the threshold (lines 5-8). The selected points are all appended together to the list of important points (line 9). The example of important points are shown in Figure 10. These reference points are used to mark the starting and ending points of a shapelet candidate, as shown in Figures 11 - 15. As a result, the algorithm could filter out a large number of shapelet candidates, while maintaining shapelet candidate's quality.

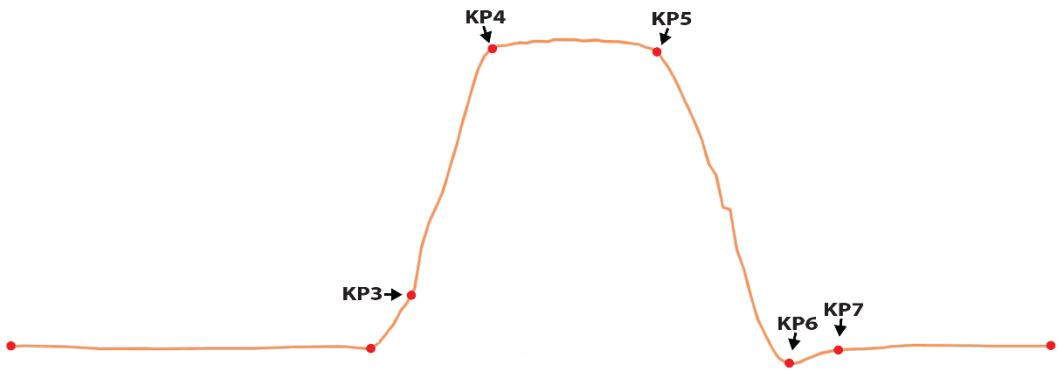

Figure 10 Important points in a gun-point time series sequence generated from LFDP algorithm.

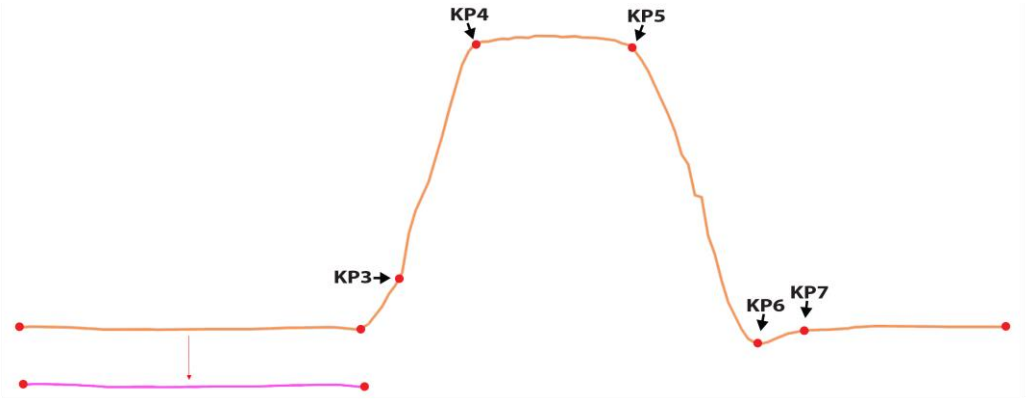

Figure 11 A subsequence created by connecting the first key point with the second key point.

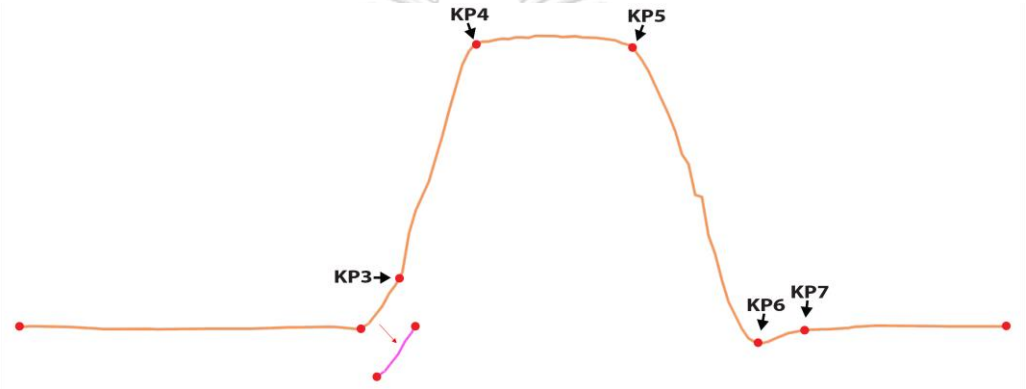

Figure 12 A subsequence created by connecting the second key point with the third key point

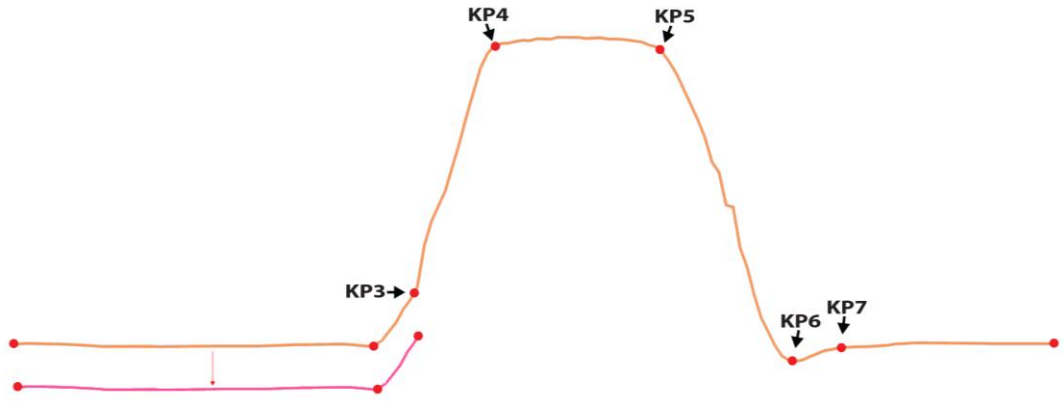

Figure 13 A subsequence created by connecting the first, second, and third key points.

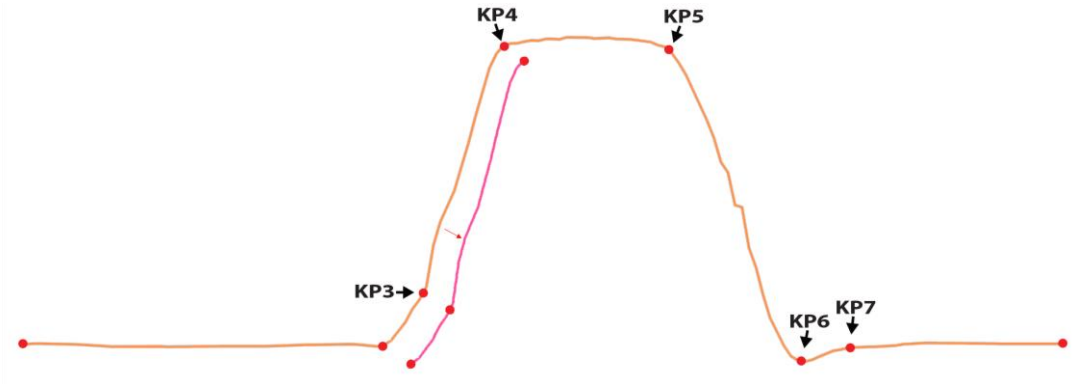

Figure 14 A subsequence created by connecting the second, third, and fourth key points.

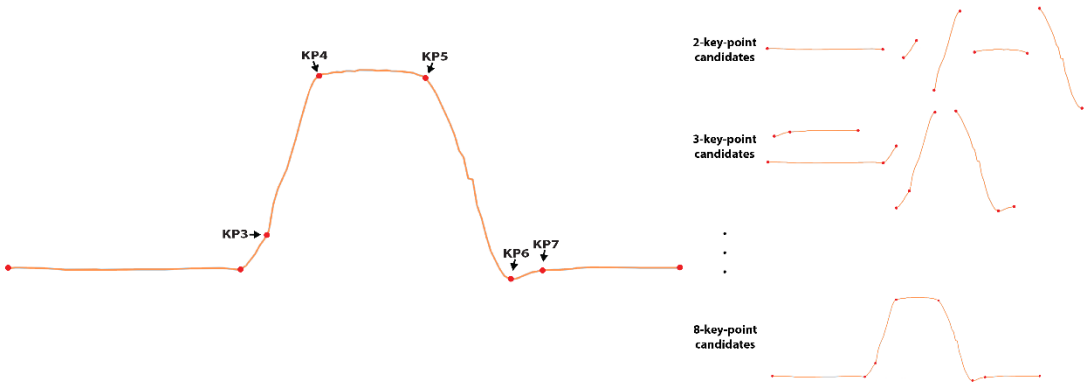

Figure 15 Example of subsequences generated from every key point

Table 6 A pseudo code of LFDP algorithm

Algorithm 6 LFDP

Input List of timeSeries *timeSeries*

List *importantPointList* ← [ ]
for all *timeSeries* **do**
    importantPoints ← [*start*,*end*]
    distance ← *infinity*
    while *distance*>*threshold* **do**
        tempTimeSeries ← createTimeSeries(*importantPoints*)
        distance ← distance(*tempTimeSeries*,*timeSeries*)
        importantPoints ← findNextBestCandidatePoint(*timeSeries*)
    importantPointList.append(*importantPoints*)
return *importantPointList*

3.1.2 Incremental Neural Network

An Incremental Neural Network is a preferable choice of clustering algorithm as it is a one-pass algorithm, which is suitable for large datasets. Moreover, it has an ability to self-adjust the

number of groups by candidate's variation. The algorithm adopts this idea from the work in (Ji et al., 2018). However, instead of one single layer, I have modified the algorithm to become a two-layered network, i.e., Candidates Averaging layer, and Class Heterogeneity layer.

3.1.2.1 Candidates Averaging Layer

A large number of candidates are similar in shape and size. Similar candidates would result in similar shapelet quality. If clustering similar shapelets together, the algorithm would be able to estimate the quality of shapelets groupwise rather than individually. The algorithm uses the length and shapelet average within the same group to calculate shapelet quality.

The first layer will average candidates of the same length from the same original time series sequence whose distance between the candidates was below the threshold. As shown in Table 7, the inputs are candidates that are grouped by their length. For all candidates with the same length (line 4), the algorithm inserts them one by one into an incremental neural network (line 5). After inserting all data with the same length, the algorithm then averages all of the candidates in each node and appends it to the list of filtered candidates (line 6). The algorithm continues until all lengthwise candidate groups are completed (line 2). The outputs are the averaged filtered candidate, based on their lengths.

Table 7 A pseudo code of candidate average layer in dual increment shapelet algorithm

Algorithm 7 Candidate Averaging Layer

Input List of tuples(*length*,*candidates*)

1: *filteredCandidates* ← [ ]
2: **for all** *length* get *candidates* **do**
3: *network* ← create IncrementalNetwork()
4: **for all** *candidates*.get(*length*) **do**
5: *network.insertNewData(candidate)*
6: *filteredCandidates* ← *network*.average()

return *filteredCandidates*

3.1.2.2 Class Purity Layer

The measurement criteria for the best shapelet rely on the power of class discrimination (Bostrom & Bagnall, 2017). To measure shapelet quality, this work adopts the idea from (Rakthanmanon & Keogh, 2013) and (Bostrom & Bagnall, 2017) in measuring the quality of the group using class heterogeneity of a shapelets group. During the shapelet grouping process, the algorithm inserts the class metadata into the groups to help in group heterogeneity calculation. The algorithm used this approximated group quality to filter out some low-quality candidate

groups instead of selecting each candidate one by one. Regardless of the length, this second layer could effectively group similar shapelets together. For example, in Figure 16, shapelet candidates that are being close to a horizontal line are clustered in cluster 1, but shapelet candidates that are shorter and have greater slopes are clustered in cluster 4.

Table 8 A pseudo code of class purity layer in dual increment shapelet algorithm

Algorithm 8 Class Purity Layer

Input List of candidates *candidates*

1: *network* ← create IncrementalNetwork()
2: **for all** *candidate* in *candidates* **do**
3: *network*.insertNewData(*candidate*)
4: *filteredCandidates* ← *network*.quality()

return *filteredCandidates*

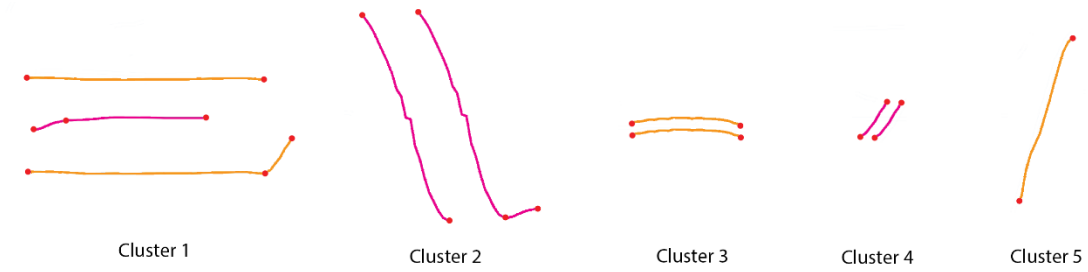

Figure 16 A cluster of similar shape shapelet candidate

3.1.3 Dual Increment Shapelet

With those strategies above, it could actually reduce the number of candidates to be evaluated by a few orders of magnitude. The remaining groups have good candidate potentials. All of the remaining candidates are then used to find the distance to all time series in the dataset. The underlying algorithm is very similar to the shapelet transform. Lastly, the results of the transform step are classified by SVM algorithm with a propose to find the best shapelet candidates. With L1 regularization, weights are consolidated with some candidates that contribute the best accuracy. As a result, an acceptable number of shapelets is acquired. A pseudo code of Dual Increment shapelet presented in Table 9.

Table 9 A pseudo code dual increment shapelet algorithm

Algorithm 9 Dual Increment Shapelets

1: $importantPoints \leftarrow \text{LFDP}()$
2: $shapeletCandidates \leftarrow \text{candidateGeneration}(importantPoints)$
3: $averageShapelets \leftarrow \text{AveragingLayer}(shapeletCandidates)$
4: $(shapeletGroup, quality) \leftarrow \text{PurityLayer}(averageShapelets)$
5: $passedCandidates \leftarrow [\,]$
6: **for all** $(shapeletsGroup, quality)$ **do**
7: **if** $quality > threshold$ **then**
8: $selectedCandidates \leftarrow shapeletGroup$
9: $shapelets \leftarrow \text{SVM-L1reg}(selectedCandidates)$
return $shapelets$

The dual increment shapelets algorithm starts by first selecting important points obtained from LFDP algorithm. Then, those selected points are used to generate shapelet candidates (lines 1-2). The generated candidates were transferred to the first layer of the incremental neural network to average candidates that share the same length (line 3). After that, these candidates were transfered to the second layer of the incremental neural network which hosted a group of similar shapelet candidates (line 4). The quality of every group in class heterogeneity layer is measured by the class heterogeneity measurement (lines 6-7). The selected groups above the threshold are kept in the selected candidate list (line 8). Then, the candidates are transformed to a distance matrix; before they are classified by SVM with L1 regularization to the best shapelets (line 9).

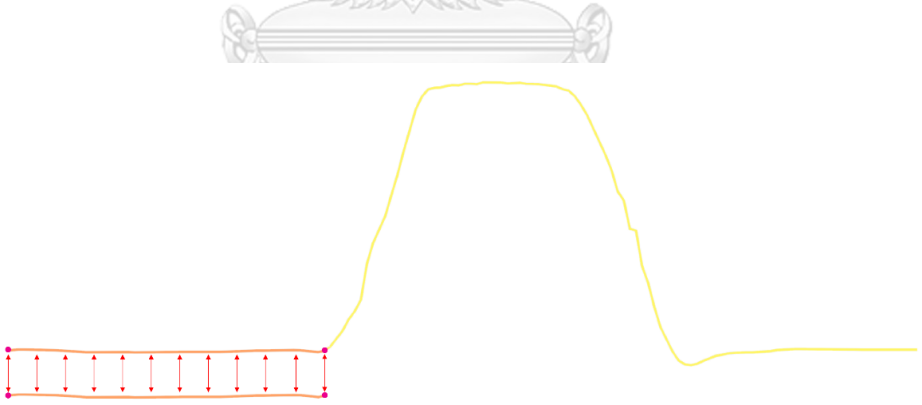

Figure 17 Euclidean distance measurement between the time series sequence and the first candidate

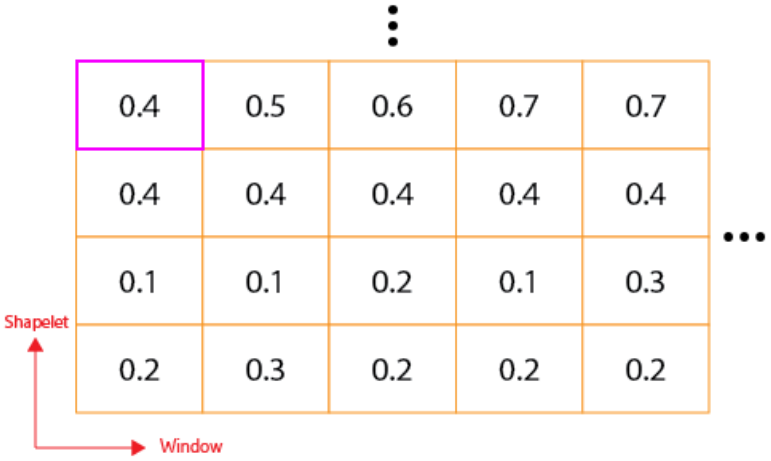

Figure 18 The top left cell of the distance matrix shows the resulting distance obtained from the operation in Figure 17.

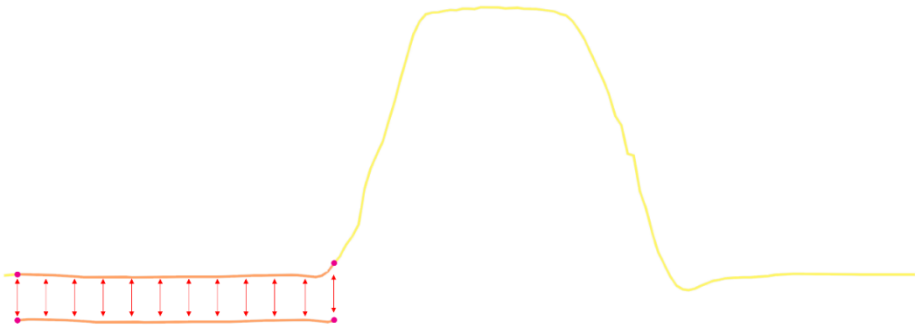

Figure 19 Euclidean distance measurement between the time series sequence and the first shapelet candidate slided to the right by 1 data point.

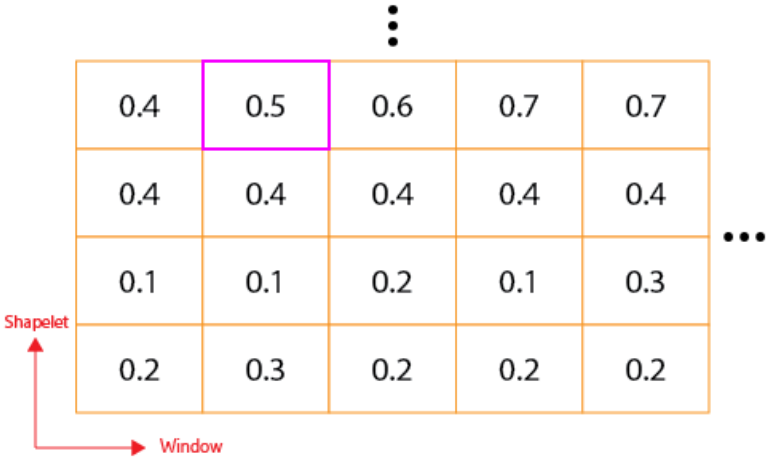

Figure 20 The marked cell shows the resulting distance obtained from the operation in Figure 19

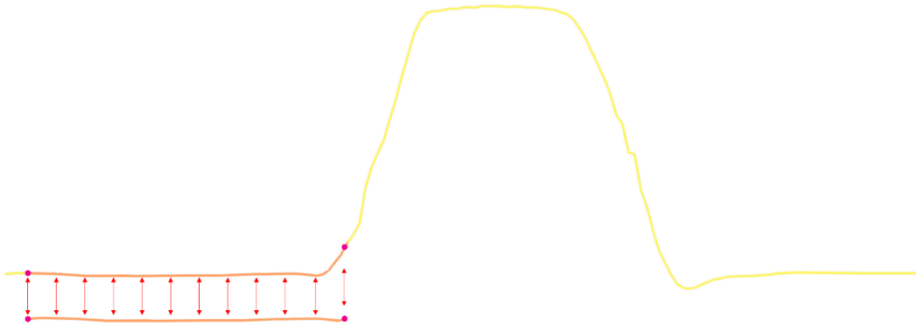

Figure 21 Euclidean distance measurement between the time series sequence and the first shapelet candidate slided to the right by 2 data point.

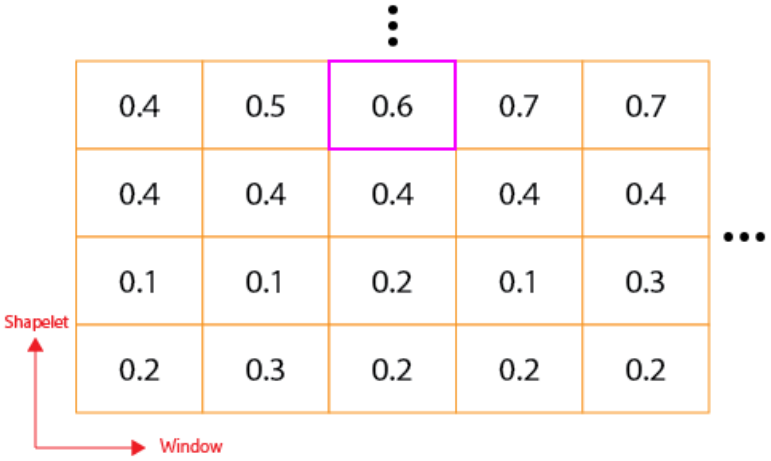

Figure 22 The marked cell shows the resulting distance obtained from the operation in Figure 21

Figure 23 Euclidean distance measurement between the time series sequence and the second candidate

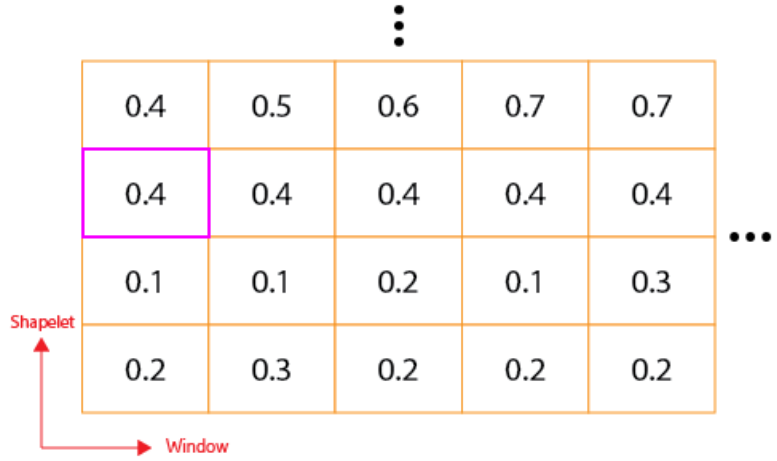

Figure 24 The marked cell shows the resulting distance obtained from the operation in Figure 23

Figure 25 Euclidean distance measurement between the time series sequence and the second shapelet candidate slided to the right by 1 data point.

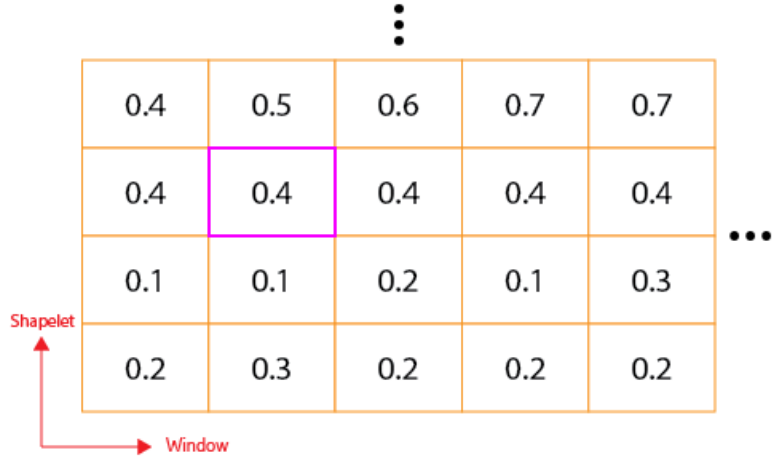

Figure 26 The marked cell shows the resulting distance obtained from the operation in Figure 25

Figure 27 Euclidean distance measurement between the time series sequence and the second shapelet candidate slided to the right by 2 data point.

Figure 28 The marked cell shows the resulting distance obtained from the operation in Figure 27

Chapter 4

Experiment and Results

4.1 Experiment setup

All experiments were run on AMD 6 Core CPU.

All experiments were written in JAVA using WEKA framework (Hall et al., 2009).

The DIS algorithm are designed and written by myself but the baseline codes are took from (Bagnall, Lines, Bostrom, Large, & Keogh, 2017) to recreate the results.

4.2 Baselines

The baselines are chosen from 3 different method of finding shapelet: Shapelet Transform (ST), Learning Shapelets (LS), and Fast Shapelets (FS). Shapelet Transform represents the brute force method of finding shapelet. Learning Shapelet represents shapelet discovery using gradient descent. Fast Shapelet represents approximated but faster shapelet discovery algorithm.

4.3 Datasets

The data in UCR repository (Chen et al., 2015) are selected to be used in this thesis. The approximate number of shapelet candidates was obtained by using the formula $O(n^2m^4)$ to calculate shapelet candidate size. This formula was also employed to measure the training feasibility of the algorithms. However, due to the exceedingly large time complexity, some baseline algorithms are unable to complete the classification in some large datasets within 24-hour period. Specifically, only 40 datasets from the total of 76 datasets could be run by Shapelet Transform algorithm. Therefore, even though the proposed DIS algorithm could be successfully trained within the time limit, for fair comparison, only these 40 datasets are chosen for the experiments

4.4 Running time comparison to the baselines

To evaluate the performance, every algorithm is repeated 5 times, and the average running time is reported. Table 10 reports the average running time for all shapelet-based classifiers.

Table 10 Average running time in millisecond for shapelet-based classifiers

Datasets	FS	LS	ST	**DIS**
Adiac	101177	99454633	10692757	**14834**
ArrowHead	10815	228300	4841654	**1596**
Beef	75734	1915378	3123344	**9364**
BeetleFly	22934	179908	5866064	**9474**
BirdChicken	17178	159999	6281603	**8060**
Car	119352	3926827	50858285	**52557**
CBF	3301	49966	213188	**516**
ChlorineConcentration	191170	1820407	52458469	**12248**
Coffee	6244	76629	302256	**1013**
DiatomSizeReduction	6882	309793	93089	**831**
DistalPhalanxOutlineAgeGroup	10385	410264	15641115	**2609**
DistalPhalanxOutlineCorrect	23501	287386	10530021	**3377**
DistalPhalanxTW	13456	1933645	5874907	**2045**
ECGFiveDays	2267	15433	159845	**224**
FaceAll	190672	37933408	78029678	**26308**
FaceFour	24148	500485	7792586	**1994**
FacesUCR	54225	12490998	19879931	**5122**
GunPoint	2578	45133	895632	**679**
ItalyPowerDemand	1166	4777	2730	**295**
Lightning7	87039	4702008	54935741	**19362**
MedicalImages	46871	7818573	28067449	**6739**
MiddlePhalanxOutlineAgeGroup	10184	412895	8862617	**1526**
MiddlePhalanxTW	14331	1917648	14721367	**3744**

MoteStrain	1738	6538	10036	**175**
OliveOil	47847	1623811	3082948	**8836**
PhalangesOutlinesCorrect	88116	968687	99125785	**16083**
Plane	14181	1658518	10914489	**2583**
ProximalPhalanxOutlineAgeGroup	9819	410964	8685065	**1355**
ProximalPhalanxOutlineCorrect	18283	269331	7089362	**2377**
ProximalPhalanxTW	10817	1938259	9705892	**1767**
ShapeletSim	31187	210767	542623	**7698**
SonyAIBORobotSurface1	1504	4862	8410	**220**
SonyAIBORobotSurface2	1561	6323	12257	**281**
SwedishLeaf	104532	36721071	86573650	**16996**
Symbols	24673	1648839	15207771	**4569**
SyntheticControl	10487	934441	2413403	**3405**
ToeSegmentation1	11377	109016	8114483	**3792**
ToeSegmentation2	14778	197764	17057920	**6911**
Trace	45370	1624364	87329377	**15214**
TwoLeadECG	1468	7206	4381	**175**

Table 10 shows that the proposed DIS algorithm evidently and significantly outperforms all other algorithms in terms of the running time. Especially in multiclass problems, DIS has an extra ability to prune more candidates than the binary class problems, being as many as 6,400 times faster in some datasets. These results indicate that the characteristics of the datasets (length, number of instances, number of classes, etc.) could be influential factors affecting the speed of DIS. In other words, the characteristics of the dataset directly affect the pruning power of the shapelet candidates, e.g., ItalyPowerDemand vs. FaceAll.

The types of algorithms also influence the running time. The results from Table 11 revealed that DIS outperformed all the baselines since it is a one-pass candidate filtering algorithm. To be more specific, comparing to DIS, Shapelet Transform is the slowest one (2,460.567 times slower), followed by Learning Shapelet (525.664 times slower), and Fast Shapelet (6.180 times slower),

respectively, as shown in Table 11. It can be explained that Brute Force algorithm like Shapelet Transform requires very large running time to run. The Gradient descent algorithm underlying the Learning Shapelet is faster than the Brute Force algorithm, and the dimensionality reduced algorithm could greatly help reduce the running time, making Fast Shapelet the fastest one among all the baselines.

Table 11 A comparison of the results of average speedup of DIS and the baselines

Fast Shapelet	Learning Shapelet	Shapelet Transform
6.180	525.664	2460.567

4.5 Accuracy comparison to the baselines

In terms of accuracy, the proposed DIS and the baselines were compared in order to compare the accuracy of DIS against all of the baselines. It should be noted that the Shapelet Transform employed in this session was parameterized using the same parameters reported in (Bagnall et al., 2017). The results are provided in Table 12.

Table 12 Classification accuracies for shapelet-based classifiers

Dataset	FS	LS	ST	DIS
Adiac	0.550	0.519	0.130	**0.729**
ArrowHead	0.577	**0.823**	0.720	0.777
Beef	0.567	**0.800**	0.567	0.767
BeetleFly	0.650	0.750	**0.800**	**0.800**
BirdChicken	**0.900**	0.800	0.750	**0.900**
Car	0.733	0.800	0.633	**0.883**
CBF	0.919	**0.990**	0.956	0.941
ChlorineConcentration	0.591	0.591	0.616	**0.648**
Coffee	0.964	**1.000**	**1.000**	0.964
DiatomSizeReduction	0.879	**0.967**	0.765	0.941
DistalPhalanxOutlineAgeGroup	0.640	**0.719**	0.691	0.698
DistalPhalanxOutlineCorrect	0.728	**0.786**	0.670	0.775

DistalPhalanxTW	**0.655**	0.626	0.647	0.647
ECGFiveDays	0.995	**1.000**	**1.000**	**1.000**
FaceAll	0.620	0.775	0.653	**0.778**
FaceFour	0.920	**0.966**	0.750	0.943
FacesUCR	0.738	**0.944**	0.671	0.878
GunPoint	0.940	**1.000**	0.953	0.993
ItalyPowerDemand	0.906	**0.963**	0.943	0.958
Lightning7	0.630	**0.808**	0.425	0.740
MedicalImages	0.605	0.686	0.471	**0.689**
MiddlePhalanxOutlineAgeGroup	0.539	**0.584**	0.532	**0.584**
MiddlePhalanxTW	0.461	0.506	**0.526**	0.494
MoteStrain	0.798	0.858	0.839	**0.874**
OliveOil	0.633	0.700	0.767	**0.867**
PhalangesOutlinesCorrect	0.724	0.748	0.685	**0.833**
Plane	0.990	**1.000**	0.924	**1.000**
ProximalPhalanxOutlineAgeGroup	0.776	0.815	0.737	**0.834**
ProximalPhalanxOutlineCorrect	0.838	0.849	0.715	**0.887**
ProximalPhalanxTW	0.727	**0.810**	0.654	0.790
ShapeletSim	**1.000**	0.978	**1.000**	0.972
SonyAIBORobotSurface1	0.686	0.827	**0.947**	0.429
SonyAIBORobotSurface2	0.790	**0.890**	0.876	0.824
SwedishLeaf	0.789	0.917	0.755	**0.928**
Symbols	**0.937**	0.930	0.823	0.930
SyntheticControl	0.937	**0.997**	0.957	0.990
ToeSegmentation1	0.943	0.930	0.934	**0.956**
ToeSegmentation2	0.692	**0.923**	0.892	0.800
Trace	**1.000**	**1.000**	0.980	**1.000**

TwoLeadECG	0.946	**0.997**	0.970	0.991

As the tradeoff for speed, DIS accuracies are expected to be lower. However, DIS does surprisingly well, being a winner among all baselines in as many as 18 datasets. The Gradient descent algorithm in Learning Shapelet could yield good results as expected, but the Fast Shapelet is the least accurate despite its fastest speed, due to the dimensionality reduced data.

Table 13 reports the performance of the proposed DIS algorithm in terms of accuracy, comparing to the three baselines. DIS generally outperforms Fast Shapelet and Shapelet Transform but is slightly less accurate than the Learning Shapelet. One possible explanation involves their underlying behavior of the algorithms. The Gradient descent algorithm could effectively search for the qualified shapelets. The DIS might prune some qualified shapelet candidates out, leading to the possible deviated result.

Table 13 A comparison of the results of accuracy of DIS and each baseline algorithm

Result	Fast Shapelet	Learning Shapelet	Shapelet Transform
Win	33	15	30
Lose	4	20	7
Tie	3	5	3

Chapter 5

Conclusions and Future work

In this work, a novel algorithm so-called “Dual Increment Shapelets (DIS)” is introduced. This thesis evaluates its performance in terms of speed and accuracy against the three baselines: Fast Shapelet, Learning Shapelet, and Shapelet Transform. The results reveal that DIS is the fastest algorithm comparing with all the baselines. With the significantly improved speed, the accuracy of DIS has only been slightly sacrificed. The key to its success is good candidate filtering ability, which is a result of LFDP candidate generation method combined with two-layered Incremental Neural Network. With these capacities, DIS offers a promising algorithm to handle large datasets while maintaining satisfactorily high accuracy. However, as a future work, The algorithm’s performance could be improved to pin-point qualified shapelets, as a result, speeding up the training time.

REFERENCES

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery, 31*(3), 606-660.

Bingham, E., & Mannila, H. (2001). *Random projection in dimensionality reduction: applications to image and text data.* Paper presented at the Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining.

Bostrom, A., & Bagnall, A. (2017). Binary shapelet transform for multiclass time series classification. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXII* (pp. 24-46): Springer.

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., & Batista, G. (2015). The ucr time series classification archive. In: July.

Grabocka, J., Schilling, N., Wistuba, M., & Schmidt-Thieme, L. (2014). *Learning time-series shapelets.* Paper presented at the Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter, 11*(1), 10-18.

Hills, J., Lines, J., Baranauskas, E., Mapp, J., & Bagnall, A. (2014). Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery, 28*(4), 851-881.

Ji, C., Liu, S., Yang, C., Pan, L., Wu, L., & Meng, X. (2018). A Shapelet Selection Algorithm for Time Series Classification: New Directions. *Procedia Computer Science, 129*, 461-467.

Ji, C., Liu, S., Yang, C., Wu, L., Pan, L., & Meng, X. (2016). *A piecewise linear representation method based on importance data points for time series data.* Paper presented at the Computer Supported Cooperative Work in Design (CSCWD), 2016 IEEE 20th International Conference on.

Keogh, E., & Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowledge and information systems, 7*(3), 358-386.

Keogh, E., Wei, L., Xi, X., Vlachos, M., Lee, S.-H., & Protopapas, P. (2009). Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures. *The VLDB journal, 18*(3), 611-630.

Lin, J., Keogh, E., Wei, L., & Lonardi, S. (2007). Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery, 15*(2), 107-144.

Okada, S., & Hasegawa, O. (2008). *Motion recognition based on dynamic-time warping method with self-organizing incremental neural network.* Paper presented at the Pattern Recognition, 2008. ICPR 2008. 19th International Conference on.

Rakthanmanon, T., & Keogh, E. (2013). *Fast shapelets: A scalable algorithm for discovering time series shapelets.* Paper presented at the proceedings of the 2013 SIAM International Conference on Data Mining.

Ye, L., & Keogh, E. (2009). *Time series shapelets: a new primitive for data mining.* Paper presented at the Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.

Zhu, Y., Wu, D., & Li, S. (2007). *A piecewise linear representation method of time series based on feature points.* Paper presented at the International Conference on Knowledge-Based and Intelligent Information and Engineering Systems.

VITA

NAME	Nattakit Vichit
DATE OF BIRTH	2 August 1992
PLACE OF BIRTH	Bangkok
INSTITUTIONS ATTENDED	Bachelor degree in computer science at Thammasat University
HOME ADDRESS	28/16 Ramintra6 Ramintra rd. Anusawari Bangkhen Bangkok 10220
PUBLICATION	Vichit, N., & Ratanamahatana, C. A. (2019, July). Dual Increment Shapelets: A Scalable Shapelet Discovery for Time Series Classification. In International Conference on Computing and Information Technology (pp. 3-14). Springer, Cham. The international conference was located at Arnoma Grand hotel, 99 Ratchadamri Rd, Khwaeng Lumphini, Khet Pathum Wan, Bangkok, Thailand Conference website: https://ic2it.org/