Effective Frequent Motif Discovery for Long Time Series Classification: A Study Using Phonocardiogram

Hajar Alhijailan[1,2][a] and Frans Coenen[1][b]
[1]*Department of Computer Science, University of Liverpool, Liverpool, United Kingdom*
[2]*College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia*
{*h.alhijailan, coenen*}*@liverpool.ac.uk, halhujailan@ksu.edu.sa*

Keywords: Time and Point Series Analysis, Frequent Motifs, Data Preprocessing, Classification, Phonocardiogram

Abstract: A mechanism for extracting frequent motifs from long time series is proposed, directed at classifying phonocardiograms. The approach features two preprocessing techniques: silent gap removal and a novel candidate frequent motif discovery mechanism founded on the clustering of time series subsequences. These techniques were combined into one process for extracting discriminative frequent motifs from single time series and then to combine these to identify a global set of discriminative frequent motifs. The proposed approach compares favourably with these existing approaches in terms of accuracy and has a significantly improved runtime.

1 INTRODUCTION

Time series analysis is directed at the extraction of knowledge from temporally referenced data. The usual application, and that of interest with respect to this paper, is the construction of a classification model for labelling (classifying) time series (Mueen et al., 2009) . However, the time series of interest are typically too large to be considered in their entirety; for example as a single feature vector. This issue can be addressed by identifying *motifs* within the time series (Dau and Keogh, 2017) . A motif in this context is some subsequence of points occurring within a time series which is deemed to be representative of the underlying class-label associated with the time series (Krejci et al., 2016). A representative motif can be defined in various ways; that considered in this paper is frequency of occurrence (Agarwal et al., 2015).

A number of motif discovery techniques have been proposed (Gao et al., 2017; Dau and Keogh, 2017) . However, the discovery of good motifs in time series remains computationally challenging; mainly because of the large number of candidate motifs that need to be considered. Schemes aimed at reducing this complexity (Mueen et al., 2009) remain problematic in that the classification accuracy tends to be adversely affected, because of the nature of the various proposed heuristics used to limit the time complexity.

A mechanism where by the motif discovery process complexity can be reduced is by preprocessing the data. A new technique, considered in detail in this paper, is to prune the time series by removing subsequences that are unlikely to be representative of any class-label. Three categories of time series subsequences for pruning can be identified: (i) subsequences that exist in every time series and hence not representative of any particular class, (ii) subsequences that appear so infrequently that they cannot be deemed relevant and (iii) subsequences that appear across two or more classes and thus cannot be usefully employed to discriminate between classes.

The precise nature of the most appropriate preprocessing technique to be adopted is very much dependent on the application domain under consideration. The application domain at which the work presented in this paper is directed at the classification of canine Phonocardiograms (PCGs) according to a variety of heart conditions. A PCG is a single variate time series, typically obtained using an electronic stethoscope. The advantage offered, with respect to computerised processing, is that the information contained in a PCG is more than that can be distinguished by the human ear or by visual inspection. The work presented in this paper is thus directed at preprocessing PCG data. Firstly by removing subsequences representing "silent gaps". Secondly by removing subsequences that cannot be frequent, using a novel technique involving clustering. Thirdly by discounting motifs that are not good discriminators of a class.

[a] https://orcid.org/0000-0002-4169-7911
[b] https://orcid.org/0000-0003-1026-6649

2 PREVIOUS WORK

The basic motif discovery technique is to determine the frequency of occurrence of candidate motifs by counting the number of similar subsequences that exist in the input data; in other words, by conducting a large number of comparisons (Keogh and Pazzani, 2001). Efficiency gains can be made by preprocessing/pruning the input data or using knowledge specific to the application domain to limit the number of comparisons. Popular preprocessing techniques in the context of time series include: Segmentation, Downsampling, Filtering, Decimation and De-noising.

An alternative pruning technique, applicable in the context of certain time series applications, including the PCG application considered in this paper, is "silent gap" removal. This is a technique frequently used with respect to applications that are founded on audio data. Silent gap removal was first proposed and adopted in the context of applications directed at Voice Activity Detection (VAD) and speech recognition (Sohn et al., 1999). The main motivation for removing silent gaps in VAD and speech recognition was that these subsequences were not likely to carry any information (Yang et al., 2010); by removing the silent gaps, the problem domain becomes more tractable. Of course silent gap identification also allows for the isolation of individual words, syllables and sentences (Ramírez et al., 2004). To the best knowledge of the authors, there is no work on silent gap removal in the context of PCG data.

3 FORMALISM

This section presents a formalism for the work presented in this paper.

Definition 1, Time Series: A time/ point series P is a sequence of x data values $\{p_1, \ldots, p_x\}$ associated with a class-label c_i taken from a set of classes $C = \{c_1, c_2, \ldots\}$. In the case of the training and test data, this label is known. In the case of previously unseen data, this is what the classifier is intended to predict. A collection of z labelled point series is then given by $T = \{\langle P_1, c_1\rangle, \ldots, \langle P_z, c_z\rangle\}$ where each P_i is a point series and $c_i \in C$.

Definition 2, Pruned Time Series: A pruned time/ point series P' of P is a sequence of y data values $P' = \{p_1, \ldots, p_y\}$ such that $y \leq x$, $P' \subseteq P$.

Definition 3, Time Series Subsequence: A subsequence s of a time series P (P') is any consecutive set of points in the time series; $s = \{p_1, \ldots, p_\omega\}$ and ω is a pre-specified length of s. The set S is the set of all possible subsequences, of length ω, in P (P'); $S = \{s_1, \ldots, s_{x-\omega+1}\}$. The generation of S is computationally expensive; given any reasonably sized time series, there will $x - \omega + 1$ ($y - \omega + 1$) subsequences. This can be mitigated against by generating only a limited number of subsequences, $S = \{s_1, \ldots, s_{max}\}$.

Definition 4, Motif: A motif m is a subsequence $s_j \in S$, where S is drawn from a point series P (P'), that is representative of a class-label c_i. One way of determining whether a motif is representative or not is to consider its frequency of occurrence. For s_j to be considered frequent, the set S must include at least σ subsequences that are in some sense all similar to s_j, as define according to a similarity threshold λ. The set of frequent motifs generated from S is then given by $M = \{\langle m_1, f_1\rangle, \langle m_2, f_2\rangle, \ldots\}$, where f_i is the frequency count.

Definition 5, Top k Motifs: The k most frequent motifs in a set M are the "Top k Motifs". It is argued that the Top k Motifs are the most representative of the underlying class associated with S. The top k motifs drawn from a set of frequent motifs M is given by the set $L = \{\langle l_1, c_1\rangle, \ldots \langle l_k, c_k\rangle\}$.

Definition 6, Motif Set: Given a collection of time series T, the complete set of identified frequent motifs is given by $\mathbf{D} = \bigcup_{i=1}^{i=z} L_i$, where z is the number of records (examples) in T.

Definition 7, Pruned Motif Set: Each motif in $\mathbf{D}$ is a good *local* representative of a class. This does not necessarily mean that it is also a good *global* representative. The set $\mathbf{D}$ therefore needs to be pruned to give D', a motif set that contains only good global representatives. The complement of D' in $\mathbf{D}$ is then the set D_s, the set of motifs that are not good global discriminators. A good representative motif is defined as one where either there are no similar motifs in $\mathbf{D}$ or if there are similar motifs in $\mathbf{D}$, they are all linked to the same class; D' and D_s can be formally defined as follows:

- $D', D_s \subset \mathbf{D} \; : \; D' \cup D_s = \mathbf{D}$ and $D' \cap D_s = \emptyset$.
- $\forall \, d_i \in D' \nexists \, d_j \in \mathbf{D} \; : \; m_i \simeq m_j \; \wedge \; c_i \neq c_j$.
- $\forall \, d_i \in D_s \; \exists \, d_j \in \mathbf{D} \; : \; m_i \simeq m_j \; \wedge \; c_i \neq c_j$.

The set D' can then be used to build a classification model of some kind.

4 FREQUENT MOTIF DISCOVERY

The top-level process is given in Algorithm 1. The input is a set of time series T, and a set of parameters: (i) ω defining the size of the motifs, (ii) *max* to limit the number of candidate frequent motifs, (iii) σ to define the concept of frequency, (iv) λ to define the concept of similarity and (v) k to limit the number of

frequent motifs selected. The output is a set D' of frequent motifs. The algorithm operates by processing each pair $\langle P_i, c_i \rangle$ in T in turn. The size of each time series P_i is first reduced (line 3) by removing "silent gaps" so as to give a time series P_i'. This is then further processed (line 4) to identify a set of sets candidate frequent motifs S_i' which is then used (lines 6 to 9) to discover the k frequent motifs, within S_i', that are good local discriminators of the class c_i. In each case, the motifs generated so far are collated into a set $\mathbf{D}$ which, if not empty, is processed further (line 12) so that only motifs that are good global class discriminators are retained; these are held in a set D' to be used as the "data bank" in classification.

Algorithm 1 Frequent Motif Discovery

Input: T, ω, max, σ, λ, k
Output: D'
1: $\mathbf{D} \leftarrow \emptyset$, Set to hold "good" frequent motifs
2: **for** $\forall \langle P_i, c_i \rangle \in T$ **do**
3: $P_i' \leftarrow silentGapRemoval(P_i)$
4: $S_i' \leftarrow candidateFrequentMotifGen(P_i', \omega)$
5: $max = \frac{max}{|S_i'|}$, $k = \frac{k}{|S_i'|}$
6: **for** $\forall S_{i_j}' \in S_i'$ **do**
7: $L \leftarrow localMotifDiscov(S_{i_j}', max, \sigma, \lambda, k, c_i)$
8: $\mathbf{D} \leftarrow \mathbf{D} \cup L$
9: **end for**
10: **end for**
11: **if** $\mathbf{D} \neq \emptyset$ **then**
12: $D' \leftarrow globalMotifSelection(\mathbf{D})$
13: **end if**
14: **return** (D')

The process, as shown in Algorithm 1, comprises four subprocesses: (i) Silent gap removal, (ii) Candidate frequent motif generation, (iii) Frequent motif discovery for local class discrimination and (iv) Frequent motif selection for global class discrimination.

4.1 Silent Gap Removal

PCGs feature cycles (heartbeats) spaced by "silent gaps". These gaps, because they appear in all the time series, cannot contribute to class discrimination and therefore should be removed. The adopted mechanism for removing silent gaps is founded on the MathWorks mechanism[1], which operates using a sliding window, of a pre-specified length, to identify a collection of non-overlapping subsequences S. For each subsequence s_j in S, two parameters are calculated: (i) the signal energy (e_j) and (ii) the spectral centroid (c_j) which are then used for pruning.

[1]https://uk.mathworks.com/matlabcentral/fileexchange/28826-silence-removal-in-speech-signals

The silent gap removal process is presented in Algorithm 2. The input is a point series P and a window size w measured in milliseconds. The output is a pruned point series P'. The process commences (line 1) by segmenting P into a set of non-overlapping subsequences S. The parameters e_j and c_j are then computed for each subsequence s_j in S and stored in E and C respectively (lines 5 to 10). The thresholds, t_e and t_c are then calculated; the calculation process is described below. Then (lines 13 to 17) for each subsequence s_j in S, its parameters (e_j and c_j) must fulfil the conditions to be appended to the end of P', the set of retained subsequences to be returned at the end.

The function *findThreshold* finds a threshold for a collection of values. It starts by generating a histogram of the input values; this is then smoothed (H'). The local maxima in H' are then identified and stored in a set M. The threshold is then calculated according to the number of maxima in M, which is then returned.

Algorithm 2 Silent Gap Removal

Input: P, w
Output: P'
1: $S \leftarrow$ Set of subsequences of length w in P
2: $P' \leftarrow \emptyset$, Empty set to hold pruned point series P
3: $E \leftarrow \emptyset$, Empty set to hold signal energy values
4: $C \leftarrow \emptyset$, Empty set to hold spectral centroid values
5: **for** $\forall s_j \in S$ **do**
6: $e_j \leftarrow$ Signal energy calculated for s_j
7: $E \leftarrow E \cup e_j$
8: $c_j \leftarrow$ Spectral centroid calculated for s_j
9: $C \leftarrow C \cup c_j$
10: **end for**
11: $t_e \leftarrow findThreshold(E)$
12: $t_c \leftarrow findThreshold(C)$
13: **for** $j = 1$ **to** $|E|$ **do**
14: **if** $e_j \geq t_e$ **and** $c_j \geq t_c$ **then**
15: $P' \leftarrow append(P', s_j)$
16: **end if**
17: **end for**
18: **return** (P')

4.2 Candidate Frequent Motif Generation

Given a pruned point series P', this can be pruned further by removing subsequences that cannot be considered to be frequent so as to retain a set of candidate frequent motifs S' ($S' \subset S$), the complete set of subsequence in P' of length ω. To do this, a novel algorithm was proposed using a hypothetical motif r referred to as the "zero motif". The similarity between each subsequence $s_i = \{s_{i_1}, \ldots, s_{i_\omega}\}$, $s_i \in S$, and r is calculated using Euclidean Distance similarity. The

obtained similarity values were used to cluster the set of subsequences S in P' into an ordered set of clusters, $CL = \{CL_1, CL_2, \dots\}$, one cluster per unit distance value, ordered according to size. The subsequences in S contained the largest cluster, CL_1, were retained; however, if the difference between the size of CL_1 and CL_2 was proportionally small, then CL_2 was also retained. Whether one or two clusters are retained was defined by a parameter θ, which was calculated using Equation 1:

$$\theta = \begin{cases} 1 & \text{if } \frac{|CL_1|-|CL_2|}{|S|} \times 100 > \alpha \\ 2 & \text{otherwise} \end{cases} \quad (1)$$

where, α is a user-defined percentage of the total number of subsequences in S. $\alpha = 5$ is suggested.

The candidate frequent motif generation subprocess is given in Algorithm 3. The input is a pruned time series P' and the window size ω. The output is a set of sets of candidate frequent motifs S'; this may consist of one or two sets. The algorithm commences by defining the zero motif. Next, the set S is generated and a corresponding set D. The set S is then processed (lines 4 to 7) so as to populate D; there is a one-to-one correspondence between S and D. The set S is then clustered (line 8) according to D. The largest cluster is retained (line 9) and the second largest might also be retained depending on the θ value (lines 10 to 13). The retained set of clusters S' is then returned.

Algorithm 3 Candidate Frequent Motif Generation

Input: P', ω
Output: S'
1: $r \leftarrow \{r_j : r_j = 0, j = 1\ to\ j = \omega\}$
2: $S \leftarrow$ Set of subsequences of length ω in P'
3: $D \leftarrow \emptyset$, Empty set to hold distance values
4: **for** $\forall s_i \in S$ **do**
5: $\quad d \leftarrow$ Euclidean similarity between s_i and r
6: $\quad D \leftarrow D \cup d$
7: **end for**
8: $CL \leftarrow$ Ordered set of clusters obtained by clustering all $s_i \in S$ according to $d_i \in D$
9: $S' \leftarrow CL_1$
10: $\theta \leftarrow$ Parameter calculated using Equation 1
11: **if** $\theta = 2$ **then**
12: $\quad S' \leftarrow S' \cup CL_2$
13: **end if**
14: **return** (S')

4.3 Frequent Motif Discovery for Local Class Discrimination

The next subprocess, given a set of candidate frequent motifs S', is to identify the most frequent motifs that, by definition, are deemed to be good local discriminators. However, even after the removal of silent gaps and infrequent subsequences, the number of remaining subsequences in S' is still likely to be large. It is therefore proposed to limit the number of candidate frequent motifs considered using a user-defined threshold *max*. The idea is to randomly select *max* candidates from the set S'. The process is given in Algorithm 4. The inputs are: a set of candidate frequent motifs S'; the thresholds *max*, σ, λ and k; and the class-label c associated with the given time series. The output is a set L of identified frequent motifs. The first step (lines 1 to 6) is to create a subset S'' from S' comprised of only *max* subsequences. Next, the frequency count f_i for each subsequence s_i in S'' is calculated using Euclidean Distance and the λ parameter. Where the count is greater than or equal to σ% of $|S'|$, the subsequence and count are stored in $M = \{\langle m_1, f_1\rangle, \langle m_2, f_2\rangle, \dots\}$ which is then ordered according to frequency count (line 15). If k is greater than the number of subsequences in M, k is adjusted to $|M|$. Next, the set L of the k most frequently occurring motifs is generated. The set L is then returned.

Algorithm 4 Local Frequent Motif Discovery

Input: S', *max*, σ, λ, k, c
Output: L
1: $S'' \leftarrow \emptyset$
2: **if** $max \geq |S'|$ **then**
3: $\quad S'' \leftarrow S'$
4: **else**
5: $\quad S'' \leftarrow$ Set of *max* subsequences from S'
6: **end if**
7: $M \leftarrow \emptyset$, Empty set to hold motifs
8: **for** $\forall s_i \in S''$ **do**
9: $\quad f_i \leftarrow$ The number of subsequences in S' that are similar to s_i according to the threshold λ
10: $\quad$ **if** $f_i \geq \frac{\sigma \times |S'|}{100}$ **then**
11: $\quad\quad M \leftarrow M \cup \langle s_i, f_i\rangle$
12: $\quad$ **end if**
13: **end for**
14: $L \leftarrow \emptyset$, Empty set to hold top k frequent motifs
15: $M \leftarrow$ The set M ordered by frequency
16: **if** $k > |M|$ **then**
17: $\quad k \leftarrow |M|$
18: **end if**
19: **for** $i = 1$ **to** k **do**
20: $\quad L \leftarrow L \cup \langle s_i, c\rangle$ $(\langle s_i, \_\rangle \in M)$
21: **end for**
22: **return** (L)

The silent gap removal, candidate frequent motif generation and local frequent motif discovery subprocesses are applied to each point series in the input set $T = \{P_1, P_2, \dots\}$ and a sequence of sets L generated, $\{L_1, L_2, \dots\}$. These are collated into a set **D**.

4.4 Frequent Motif Selection for Global Class Discrimination

Although the motifs in **D** were deemed to be good local class discriminators, this did not automatically mean that they were also good global class discriminators; **D** may contain motif-class pairs that contradict each other. Thus, the final step was to derive a set $D' \subset \mathbf{D}$ that comprised only good global class discriminators. The process is presented in Algorithm 5. The input is the set **D**. Each motif-class pair in **D** is then compared with all other pairs in **D** and if no similar motif associated with a different class is found, the motif-class pair is added to the set D' (line 11). Similarity is again measured using Euclidean Distance and the λ parameter. The dataset D' can then be used as a Nearest Neighbour Classifier (NNC) "data bank".

Algorithm 5 Global Motif Selection

Input: **D**, λ
Output: D'
1: $D' \leftarrow \emptyset$, Set to hold "good" frequent motifs
2: **for** $\forall \langle m_i, c_i \rangle \in \mathbf{D}$ **do**
3: isGoodGlobalDisctimiator $\leftarrow$ **true**
4: **for** $\forall \langle m_j, c_j \rangle \in \mathbf{D}$, $i \neq j$ **do**
5: **if** $c_i \neq c_j$ **and** $m_i \simeq m_j$ **then**
6: isGoodGlobalDisctimiator $\leftarrow$ **false**
7: **break**
8: **end if**
9: **end for**
10: **if** isGoodGlobalDisctimiator **then**
11: $D' \leftarrow D' \cup \langle m_i, c_i \rangle$
12: **end if**
13: **end for**
14: **return** (D')

5 EVALUATION

For the evaluation, a dataset of canine PCGs was used (described in Subsection 5.1). Five sets of experiments were conducted. The first two were designed to evaluate the operation of the first two subprocess. The third set was designed to identify the most appropriate values for the parameters ω, *max*, λ and σ. The fourth was directed at an investigation of the runtime complexity of the proposed method, and the fifth at the quality of the generated motifs.

5.1 Evaluation Data

The data used for the evaluation was a set of canine PCGs encapsulated as WAVE files. It was collected using an electronic stethoscope.The resulted point series were 72 series; the average length of a single series was 740,550 points. Each point series had an associated class-label selected from the class attribute set $\{B_1, B_2, C, Control\}$. The first three are stages of Mitral Valve disease that appear in the data collection.

5.2 The Silent Gap Removal Evaluation

Experiments were conducted to determine the most appropriate value for w with respect to PCG signals. To this end, a range of window sizes from $w = 10$ ms to $w = 90$ ms, increasing in steps of 10 ms, was experimented with. It was found that a window size of 10 ms produced the best result;this was the value therefore used in this paper. Using $w = 10$, the input time series was reduced by a little less than half as a result of applying silent gap removal.

5.3 The Candidate Frequent Motif Generation Evaluation

This subprocess takes as input a pruned point series P' and a window size ω. Experiments were conducted using $\omega = \{100, 200, 300\}$ to identify the most appropriate parameter settings. The results indicated that when $\omega = 200$, the point series size is reduced by approximately a further 45% (69% of the original size); when $\omega = 100$ and $\omega = 300$, the point series size is reduced by a further 27% (59% of the original size).

5.4 Parameter Setting for Frequent Motifs Discovery

Recall that the process required five parameters: (i) ω: desired frequent motif length expressed in terms of a number of points, (ii) *max*: maximum number of motifs to be considered, (iii) λ: similarity threshold expressed in terms of a maximum distance between two motifs, (iv) σ: frequency threshold expressed in terms of a minimum percentage of possible motifs and (v) k: maximum number of motifs to be selected ($k < max$).

The selected values for these parameters all affect on the number of frequent motifs identified and consequently the quality of any further utilisation of the motifs. Clearly, the higher the σ value, the fewer the number of motifs that would be identified because the criteria for frequency would become stricter as σ increased. Inversely, the higher the λ value, the greater the number of motifs that would be identified because the criteria for similarity would become less strict as λ increased. It was anticipated that as ω increased, the number of frequent motifs would decrease as there would be fewer subsequences to choose candidate frequent motifs from. The values for *max* and k would also impact on the number of identified, and then selected, candidate frequent motifs.

To identify the most appropriate parameter settings, a range of values for ω and *max* were considered, $\{100, 200, 300\}$ and $\{20, 40, 60\}$ respectively.

The value for k was set to $k = \frac{max}{2}$ although any value less than *max* could be used. A range of values for λ and σ was also experimented with, from 0.005 to 0.100 increasing in steps of 0.005. Both λ and σ were considered to be more significant with respect to the performance of the proposed approach and thus a greater number of values was consider compared to the range of values considered for ω and *max*. The experiments indicated that there were clear "peaks" in the number of motifs identified when $\omega = 100$, while in the case of $\omega = 200$ and $\omega = 300$ a "plateaux" was reached before the number of motifs discovered started to decrease as σ and λ increased. As expected, the number of motifs discovered increases as *max* (k) increased, and tended to decrease as ω decreased. The best general setting for λ and σ, regardless of the value of *max* or ω, was in the region of 0.025; although it can be concluded that λ had more influence on the number of motifs discovered than σ.

5.5 Runtime Evaluation

To determine the runtime complexity, nine sets of experiments were conducted using $\omega = \{100, 200, 300\}$ and $max = \{20, 40, 60\}$. The adopted values for both λ and σ were 0.025; $k = \frac{max}{2}$ was used. The runtime results are presented in Table 1, these are average runtime obtained by running each experiment ten times. In the table, runtimes are presented in seconds for: (i) Silent Gap Removal (SGR), (ii) Candidate Frequent Motif Generation (CFMG), (iii) Local Frequent Motif Discovery (LFMD) and (iv) Global Frequent Motif Selection (GFMS). The last column presents the average runtime to process a single PCG (the sum of the values in the previous four columns divided by 72, the number of records in the test dataset). From the table, it can be clearly seen that the larger the ω value, the less time that was required to generate a set of candidate frequent motifs because when using a large ω, there were fewer subsequences to consider. However, the runtime required for LFMD increased with ω because larger subsequences require more processing. The runtime required for GFMS is negligible.

The total runtime required for the proposed process to undertake the nine experiments was roughly 23 minutes. This compared very favourably with the PCGseg technique reported in (Alhijailan et al., 2018) which adopted a segmentation approach to reducing the complexity of the motif generation and the MK algorithm (Mueen et al., 2009) to extract motifs. In (Alhijailan et al., 2018), a runtime of 14 hours per experiment was recorded (versus 2:59 minutes in this paper) using the same dataset and similar hardware as used in this paper. Using the best accuracy result, the proposed process required only 0.46 second to mine one record whereas PCGseg required 7 minutes.

Overall, the proposed algorithm reduced the generation time by a factor of 343 compared with PCGseg, without adversely affecting the quality of the motifs discovered as discussed further in Subsection 5.6 below. The algorithms were implemented using the Java object oriented programming language and run on an iMac Pro (2017) computer with 8-Cores, 3.2GHz Intel Xeon W CPU and 19MB RAM.

Table 1: Runtime for Frequent Motif Discovery process.

ω	*max*	SGR	CFMG	LFMD	GFMS	Avg.
100	20	4.01	8.40	20.36	0.02	0.46
	40			40.57	0.09	0.74
	60			60.90	0.18	1.02
200	20		6.82	62.84	0.01	1.02
	40			126.11	0.06	1.90
	60			190.63	0.14	2.80
300	20		6.02	132.44	0.00	1.98
	40			264.22	0.01	3.81
	60			398.44	0.04	5.67

5.6 Classification Accuracy

In the previous subsection, it was demonstrated that the proposed algorithm speeds up the discovery process to a matter of seconds per time series; much faster than the runtimes presented in (Alhijailan et al., 2018). However, any speed up in runtime must not be offset against a loss in the quality of the identified motifs. The authors also wished to investigate how the various parameters used by the proposed approach might influence the quality of the identified motifs. To quantify the quality of the motifs, a classification scenario was considered. The idea was to use a subset of the selected motifs to define a classification model and the remaining motifs to test the model. More specifically, the well-known Nearest Neighbour Classification (NNC) approach (Dasarathy, 1991) was adopted because NNC is frequently used in the context of time/point series analysis (Wu and Chau, 2010). Two values for the number of nearest neighbours to be identified were used, $k_{NNC} = \{1, 3\}$. The NNC bank comprised a set of motifs, $\mathbf{D} = \{\langle m_1, c_1\rangle, \langle m_2, c_2\rangle, \dots\}$ where m_i is a frequent motif and c_i is a class-label taken from a set of classes C. The time series to be labelled, the query time series, will then be processed, using the proposed Frequent Motif Discovery algorithm, so that it is represented as a set of motifs, $Q = \{m_1, m_2, \dots\}$ each of which was to be matched with the motifs held in $\mathbf{D}$.

Given that the proposed process will typically identify more than one frequent motif in each time series, $|Q|$ classification labels will be identified; only one is required, the most appropriate label. To select it, three different methods were used: Shortest Distance (SD), Shortest Total Distances (STD) and Highest Votes (HV). The SD simply chooses the class associated with the most similar motif. The STD chooses

the class associated with the lowest accumulated distance; the total similarity distances is calculated for each class and the class with the shortest total distance selected. The HV chooses the most frequent class, the class with the highest number of votes is selected. In each case, if there is more than one winner, one of the other class selection methods is applied. For the experiments, the same parameter values as used in the runtime experiments reported above was used here.

Ten-cross validation (TCV) was used throughout. Average results are presented in Table 2; best results highlighted in bold font. Inspection of the results indicates that accuracy was around 70% regardless of the k_{NNC} value and class selection method used. It can also be observed that there was little to choose between methods. Best accuracy, precision, recall and f-score values were 73.0%, 0.386, 0.445 and 0.390, this compares favourably with the results reported in (Alhijailan et al., 2018) where the same dataset and an alternative motif-based approach was used, where accuracy, precision, recall and f-score values of 70.8%, 0.191, 0.312 and 0.218 were reported using segmented data, and 71.9%, 0.218, 0.308 and 0.247 using unsegmented data. In other words, the proposed algorithm identifies effective motifs.

The best accuracy was obtained using SD combined with $\omega = 100$ and $max = 20$, giving 73.0% and 72.2% for $k_{NNC} = 3$ and $k_{NNC} = 1$ respectively. For SD and HV, the smaller the window size, the better the accuracy while for STD the pattern for the accuracy values seemed unclear. With respect to the *max* parameter, $max = 20$ produced the best result. It should also be noted that the recorded standard deviation for accuracy was good (in the region of 0.05.) Overall, the combination of $\omega = 100$ and $max = 20$, coupled with SD, gave best result; a combination that was also the fastest, 0.46 second to classify a single record.

Given the $\omega = 100$ had produced the best results, it was hypothesised that with an even smaller window sizes, accuracy might increase. Hence, an extra experiment was conducted using $\omega = 50$ combined with the best performing parameter settings and class selection method ($max = 20$ with $K_{NNC} = 3$ coupled with SD class selection). However, it was found that accuracy dropped to 70.2%. It was also hypothesised, given that best results were obtained when $max = 20$, that accuracy might be improved if a lower value for *max* was considered. An extra experiment with $max = 10$ was therefore undertaken, with all other parameters as before, but it was found that this significantly reduced the accuracy to 59.6%.

5.7 Comparison of Pruning Techniques

Further experiments were conducted to determine the affect on accuracy when either the silent gap removal phase or the candidate frequent motif generation phase was omitted, and when both were omitted. In the first case, line 3 in Algorithm 1 was removed and the *candidateFrequentMotifGen* function (line 4) called with P_i instead of P'_i. In the second case, line 4 was replaced with:

$S'_i \leftarrow$ Set of subsequences in P'_i of length *omega*

In the third case, both lines 3 and 4 in Algorithm 1 were replaced with:

$S'_i \leftarrow$ Set of subsequences in P_i of length *omega*

The results obtained using the best performing parameters are given in Table 3. The accuracy results

Table 2: Classification performance measures.

k_{NNC}	Method	ω	*max*	Acc.	Prec.	Rec.	F-S
1	SD	100	20	**0.722**	**0.338**	**0.428**	**0.374**
			40	0.669	0.207	0.283	0.234
			60	0.644	0.179	0.237	0.201
		200	20	**0.688**	**0.386**	**0.403**	**0.390**
			40	0.680	0.286	0.374	0.313
			60	0.660	0.262	0.308	0.270
		300	20	0.619	0.164	0.183	0.166
			40	**0.630**	**0.212**	0.237	0.220
			60	0.629	0.210	**0.241**	**0.222**
	STD	100	20	**0.660**	**0.266**	**0.312**	**0.283**
			40	0.611	0.199	0.208	0.197
			60	0.611	0.208	0.245	0.217
		200	20	0.633	0.226	0.253	0.227
			40	0.652	0.268	0.333	0.281
			60	**0.673**	**0.373**	**0.345**	**0.355**
		300	20	0.598	0.112	0.145	0.122
			40	**0.670**	**0.289**	**0.324**	**0.298**
			60	0.581	0.137	0.116	0.122
	HV	100	20	0.680	**0.240**	0.283	**0.255**
			40	**0.703**	0.225	**0.299**	0.249
			60	0.694	0.172	0.278	0.208
		200	20	0.667	0.264	0.328	0.290
			40	**0.671**	**0.337**	**0.387**	**0.347**
			60	0.645	0.252	0.295	0.262
		300	20	0.643	0.205	0.233	0.210
			40	**0.672**	**0.304**	**0.329**	**0.304**
			60	0.609	0.178	0.237	0.194
3	SD	100	20	**0.730**	**0.341**	**0.445**	**0.381**
			40	0.676	0.201	0.287	0.231
			60	0.658	0.185	0.254	0.211
		200	20	**0.687**	**0.360**	**0.412**	**0.380**
			40	0.665	0.274	0.358	0.300
			60	0.686	0.320	0.378	0.331
		300	20	**0.626**	**0.176**	0.191	0.178
			40	0.622	0.174	**0.212**	**0.189**
			60	0.599	0.147	0.195	0.159
	STD	100	20	**0.660**	**0.270**	**0.333**	**0.287**
			40	0.613	0.166	0.220	0.184
			60	0.625	0.225	0.262	0.227
		200	20	**0.653**	**0.291**	**0.370**	**0.316**
			40	0.637	0.272	0.304	0.279
			60	0.645	0.267	0.349	0.289
		300	20	0.605	0.114	0.154	0.127
			40	**0.685**	**0.331**	**0.362**	**0.338**
			60	0.574	0.114	0.129	0.112
	HV	100	20	**0.695**	**0.219**	**0.299**	**0.250**
			40	0.689	0.188	0.291	0.224
			60	0.687	0.145	0.266	0.185
		200	20	**0.680**	**0.306**	**0.395**	**0.340**
			40	0.671	0.239	0.362	0.278
			60	0.638	0.213	0.270	0.228
		300	20	**0.658**	**0.247**	0.254	0.246
			40	0.645	0.243	**0.279**	**0.250**
			60	0.588	0.139	0.212	0.161

are average results obtained using TCV. Considering silent gap removal (SGR) first, the first two rows in the table, it can be seen that SGR has a slight adverse effect on accuracy. It did improve runtime although this is not obvious from this table because different ω and *max* values produced the best results (recall that low ω and *max* values result in efficiency gains because they entail less calculation). Candidate frequent motif generation (CFMG), on the other hand, had a positive effect on accuracy and resulted in significant speed up (although again it should be noted that the results reported in Table 3 were obtained using different ω values). When the two are run together, as in the case of the earlier experiments, accuracy was slightly reduced, because of the negative effect of SGR, but runtime is enhanced considerably.

Table 3: The best classification accuracy for finding frequent motifs with different preprocessing techniques.

Preproc. Tech.		Attributes				Results	
SGR	CFMG	Method	ω	*max*	k_{NNC}	Acc.	Runtime (Sec.)
X	X	HV	300	20	3	0.721	8.23
✓	X	HV	100	60	1	0.714	16.44
X	✓	SD	300	20	3	0.736	3.69
✓	✓	SD	100	20	3	0.730	0.46

6 CONCLUSIONS

An approach to Frequent Motif Discovery, applicable to PCG time series, has been proposed. The proposed method addresses the challenge of finding discriminative motifs in long time series by proposing two pruning mechanisms: (i) silent gap removal and (ii) candidate frequent motif generation. The motivation for the first was that little useful information could be extracted from "silent gaps". The second mechanism featured a novel way of clustering subsequences, without comparing all subsequences with all other subsequences, to identify the most frequently occurring subsequences. The performance of the proposed approaches was ascertained in the context of runtime and the quality of the motifs identified; the latter analysed in terms of a classification scenario. The results indicated a classification accuracy comparable with other motif-based approaches but offering significant runtime advantages.

REFERENCES

Agarwal, P., Shroff, G., Saikia, S., and Khan, Z. (2015). Efficiently discovering frequent motifs in large-scale sensor data. In *Proceedings of the Second ACM IKDD Conference on Data Sciences (CoDS'15)*, pages 98–103.

Alhijailan, H., Coenen, F., Dukes-McEwan, J., and Thiyagalingam, J. (2018). Segmenting sound waves to support phonocardiogram analysis: The pcgseg approach. In Geng, X. and Kang, B.-H., editors, *PRICAI 2018: Trends in Artificial Intelligence*, pages 100–112, Cham. Springer International Publishing.

Dasarathy, B. V. (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press tutorial. IEEE Computer Society Press.

Dau, H. A. and Keogh, E. (2017). Matrix profile v: A generic technique to incorporate domain knowledge into motif discovery. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 125–134, NY, USA. ACM.

Gao, Y., Lin, J., and Rangwala, H. (2017). Iterative grammar-based framework for discovering variable-length time series motifs. In *IEEE International Conference on Data Mining*, pages 111–116. IEEE.

Keogh, E. J. and Pazzani, M. J. (2001). Derivative dynamic time warping. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–11.

Krejci, A., Hupp, T. R., Lexa, M., Vojtesek, B., and Muller, P. (2016). Hammock: a hidden markov model-based peptide clustering algorithm to identify protein-interaction consensus motifs in large datasets. *Bioinformatics*, 32(1):9–16.

Mueen, A., Keogh, E., Zhu, Q., Cash, S., and Westover, B. (2009). Exact discovery of time series motifs. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 473–484.

Ramírez, J., Segura, J., Benítez, C., Ángel Torre, and Rubio, A. (2004). Efficient voice activity detection algorithms using long-term speech information. *Speech Communication*, 42(3):271 – 287.

Sohn, J., Kim, N. S., and Sung, W. (1999). A statistical model-based voice activity detection. *IEEE Signal Processing Letters*, 6(1):1 – 3.

Wu, C. and Chau, K. (2010). Data-driven models for monthly streamflow time series prediction. *Engineering Applications of Artificial Intelligence*, 23(8):1350 – 1367.

Yang, X., Tan, B., Ding, J., Zhang, J., and Gong, J. (2010). Comparative study on voice activity detection algorithm. In *Proceedings of the 2010 International Conference on Electrical and Control Engineering*, ICECE '10, pages 599–602, Washington, DC, USA. IEEE Computer Society.