# Fast Channel Selection for Scalable Multivariate Time Series Classification

Bhaskar Dhariyal, Thach Le Nguyen, and Georgiana Ifrim

School of Computer Science, University College Dublin, Ireland
{bhaskar.dhariyal, thach.lenguyen, georgiana.ifrim}@insight-centre.org

**Abstract.** Multivariate time series record sequences of values using multiple sensors or channels. In the classification task, we have a class label associated with each multivariate time series. For example, a smartwatch captures the activity of a person over time, and there are typically multiple sensors capturing aspects of motion such as acceleration, orientation, heart beat. Existing Multivariate Time Series Classification (MTSC) algorithms do not scale well with large datasets, and this leads to extensive training and prediction times. This problem is attributed to an increase in the number of records (e.g., study participants), duration of recording (time series length), and number of channels (e.g., sensors). Existing MTSC methods do not scale well with the number of channels, and only a few methods can complete their training on the medium sized UEA MTSC benchmark within 7 days. Additionally, for some problems, only a few channels are relevant for the learning task, and thus identifying the relevant channels before training may help with improving both the scalability and accuracy of the classifiers, as well as result in savings for data collection and storage. In this work, we investigate a few channel selection strategies for MTSC and propose a new approach for fast supervised channel selection. The key idea is to use channel-wise class separation estimation using fast computation on centroid-pairs. We evaluate the impact of our new method on the accuracy and scalability of a few state-of-the-art MTSC algorithms and show that our approach can dramatically reduce the input data size, and thus improve scalability, while also preserving accuracy. In some cases, the runtime for training the classifier was reduced to one third of the runtime on the original dataset. We also analyse the performance of our channel selection method in a case study on a human motion classification task and show that we can achieve the same accuracy using only one third of the data.

**Keywords:** Channel Selection · Dimension Reduction · Time Series.

## 1 Introduction

Time series are data recorded as ordered sequences of numeric values and are encountered in many applications. The proliferation of IoT and sensor technology has rapidly fuelled the collection of such sequential data. Furthermore, the onset of the covid19 pandemic has also enhanced the growth of temporal data collection. For instance, a study [1] in March 2021 reported 28% growth in the market for wearable sensing devices. Besides sensors, multimedia files like images, audio, and video can also be converted to time series to save on data storage, and thus become significant contributors to time series database growth.

Time-series applications vary across domains, e.g. sports science, agriculture, or healthcare. For example, a person lifts a barbell above their head from shoulder level in the Military Press exercise. The motion of various body parts during the exercise can be captured to analyse the correctness of the exercise execution. Sensors or video recordings can help track the movement of body parts in the form of temporal data (time series) [18]. The body parts that act as data sources are known as channels in the time series context, and these channels record time-series data simultaneously. The execution of the exercise can be classified into normal and aberrant subtypes. The task of assigning discrete labels to multi-channel time series is known as Multivariate Time Series Classification (MTSC). In the case of a single channel, the task of assigning a label is known as Univariate Time Series Classification (UTSC). Figure 1 illustrates the video capture of a person doing a Military Press exercise and the extraction of multivariate time series using body pose estimation with OpenPose [3].

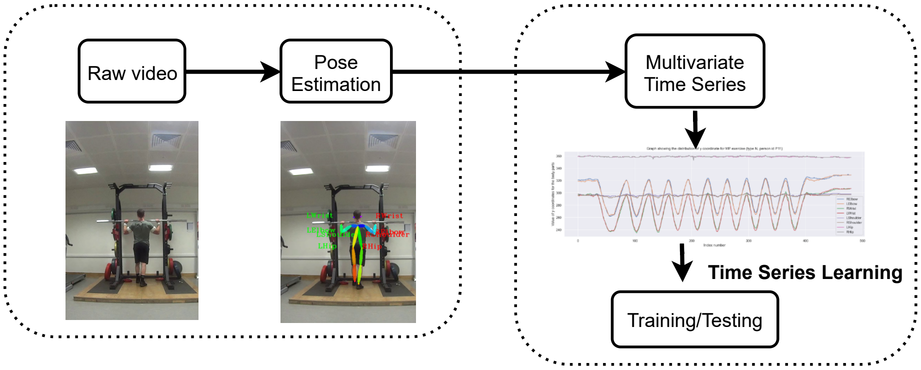


**Fig. 1.** From video to multivariate time series using OpenPose (figure from [18]).

Research in UTSC has made significant progress [2], but there is much less work done on MTSC [13]. Most literature in UTSC considers the MTSC problem as an extended version of UTSC and tends to adapt UTSC methods for MTSC. However, such methods ignore the computational components such as space and time complexity which are crucial elements for MTSC, thus rendering most of the state-of-the-art (SOTA) classifiers infeasible for practical applications. The recent studies [5, 13] highlighted scalability as a big challenge for SOTA classifiers in MTSC, with many existing algorithms not able to complete training on 26 medium-sized UCR MTSC datasets within 7 days. The scalability challenge can be analysed from three perspectives: the number of channels, length of time series and number of samples in the dataset. This study focuses on the first aspect and proposes a new method to select relevant channels from the training data, before training a classifier. The study's primary objective is to enable existing SOTA classifiers to scale better with an increase in the number of MTSC channels, by reducing the time and memory required for computation, while maintaining accuracy. In particular, we examine the impact of our channel selection approach on the recent MTSC algorithms Rocket [4], MrSEQL [10], Weasel-MUSE [16] and 1NN-DTW [13]. The main contributions of this study are:

- We propose three greedy channel selection strategies for MTSC, to scale up existing MTSC algorithms.
- We conduct extensive experiments on the UCR MTSC benchmark and report a 70% reduction in computation time for the combination of channel selection plus training MTSC algorithms, while preserving the classifier accuracy.
- We show that not all the data is useful for classification and that we achieve significant data storage savings, e.g., 70% of the original data can be discarded with our approaches.
- We present a case study of our methods on a real-world, 25-channel MTSC dataset, recorded for the Military Press strength and conditioning exercise.

The rest of the paper is structured as follows. In Section 2, we briefly describe the SOTA MSTC approaches and existing channel selection strategies. Section 3 presents our proposed methods. Section 4 introduces the UEA MTSC benchmark used for the experiments and reports our empirical results. In Section 5 we perform a case study on the Military Press dataset. We conclude our study in Section 6.

## 2 Related Work

In this section we give a brief overview of recent MTSC methods and discuss existing approaches for channel selection.

### 2.1 Multivariate Time Series Classification

The recent empirical survey [13] provides a detailed overview of progress in MTSC. Here we describe a subset of those methods, with a specific focus on methods that were shown to complete the training and testing on the 26 equal-length UEA MTSC datasets within 7 days and do not require advanced HPC infrastructure.

**1NN-DTW** is a 1-Nearest Neighbour classifier with Dynamic Time Warping (DTW) distance and one of the most popular methods in MTSC. In [17] the authors proposed two versions of DTW for MTS data, $DTW_I$ and $DTW_D$, to study the impact of DTW on multiple channels. The main difference between the two versions is how they compute the distance between two multivariate time series. The $DTW_I$ assumes each channel of MTS as an independent univariate time-series and consequently sums up the distances for each channel pair. It calculates the optimal path $P$ based on the pointwise distance between the time series. The $DTW_D$ assumes that the correct warping is the same across all channels; it computes the distance between two time series by first summing up the distance across each channel. Unlike for $DTW_I$, in $DTW_D$ the optimal path P is based on the euclidean distance between two vectors that represent all channels. Although both versions of DTW have high accuracy and are considered a strong baseline for any MTSC task, they are computationally expensive[1] and have been outperformed in accuracy by more recent methods [13]. Both $DTW_I$ and $DTW_D$ are heavily impacted by the number of time series channels, thus optimising the number of channels can drastically help in improving their scalability.

**MrSEQL-SAX** [10] is a linear classifier that extracts symbolic features from time series. The method first transforms time-series data to multiple symbolic representations of different domains (e.g., SAX [11] in the time domain and SFA [15] in the frequency domain) and different resolutions (i.e., different window sizes). The classifier extracts discriminative subsequences from the symbolic representations and these subsequences are later combined to form a feature vector used to train a classification model. In [10] the authors showed that adding features from different representations types (e.g., SAX and SFA) boosts the accuracy of the classifier. The method was initially developed for UTSC but was also adapted for MTSC; the adapted version views each channel as an independent representation of the time series. Unlike UTSC ensemble classifiers, MrSEQL uses all the extracted features from the different representations by combining them into a single feature space to train the final model. The SAX/SFA symbolic transforms are computationally expensive. Furthermore, transformation over multiple windows iterating over full time series incurs a high cost to the scalability of the classifier. Therefore, reducing the number of channels has potential to significantly improve the scalability of MrSEQL. MrSEQL-SAX is a version of MrSEQL restricted to use only SAX features (this version is more efficient than the one using both SAX and SFA features).

**WEASEL-MUSE** [16] is an extension of the WEASEL algorithm developed for UTSC. The classifier builds a bag-of-pattern (BOP) model using the SFA transform for every channel. By rolling multiple windows of varying size on raw and derivative time series, this method transforms those segments into unigram and bigram words. The classifier links these words to their respective channel and creates a histogram for each channel separately. Since there are many features for every channel, the Chi-square feature selection method removes the irrelevant ones. The selected features are concatenated into a single feature vector which is fed to a logistic regression algorithm. Similar to MrSEQL, the WEASEL-MUSE classifier also iterates over the entire time series for every channel and performs the SFA transform for every window. This iteration and transformation increase the overall computation cost. Also, storing many unigrams and bigrams in memory is quite expensive.

[1] We have evaluated here the sktime implementation of DTW.

**ROCKET** [4] is a recent classifier initially developed for UTSC and also extended to MTSC. Models produced by ROCKET are often highly accurate while keeping the computational burden low. Inspired by CNN, ROCKET relies on convolutional kernels to extract features. Instead of learning the weights of kernels through backpropagation, ROCKET randomly generates many convolutional kernels. Additionally, kernel properties like length, dilation, stride, bias and zero-padding are sampled at random. For the MTSC task, the internal channel selection is also random. A close look at the code shows that there are at most 12 channels selected for any MTSC dataset, so the runtime of this algorithm is not significantly affected by the number of channels, especially for datasets with more than 12 channels. In principle, the kernels are just a simple linear transformation of the input time series producing a new time series. A linear classifier, ridge regression, trains on the feature vector formed by global max pooling and the proportion of positive values (PPV) features extracted from the convolution time series from every channel. ROCKET has become very popular due to its high accuracy and speed, yet the impact of the number of channels on this classifier in the MTSC task is not yet examined.

### 2.2 Channel Selection for Multivariate Time Series Classification

Channel Selection for multivariate time series is a recurring topic in the MTSC literature. However, the focus of most work has been on accuracy, rather than scalability. The most recent work on channel selection [8] tries to identify the best subset of channels. The author's method calculates a merit score based on correlation patterns of the outputs from the classifiers. The algorithm iterates through every possible subset to calculate the merit score, followed by wrapper search on the subset with top 5% merit score. 1NN-DTW is employed to perform the classification. DTW is computationally expensive [5], and using DTW over every possible subset amplifies this problem. Another notable study is CleVer [19] where the author proposed three unsupervised feature subset selection techniques employing Common Principal Component Analysis (CPCA)[9] to measure the importance of each sensor. The authors build a correlation coefficient matrix among different channels for each MTS. The principal components of each coefficient matrix are calculated, and all the principal components are aggregated together, and descriptive component principal components are calculated. The $l_2$-norm of the resulting vector generates the rank of each channel. The work [7] proposed a framework for channel selection using a voting-based method. The two criteria used were distance-based classification and confidence-based classification. These methods were proposed for streaming data, which is outside the scope of the current study.

In the recent study [6], the authors presented an algorithm for channel ranking and channel selection. The key idea is that if a channel produces similar time series with the same label and different time series with a different label, then it is an informative channel. The channel ranking algorithm assigns a relevance score for each channel. The relevance scores are constructed on a similarity graph among the channels. The authors find the largest eigenvector of the normalized adjacency matrix of the similarity graph, which reflects its cluster structure. Apart from channel ranking the authors also propose channel subset selection. From the adjacency matrix above, the algorithm finds the linear combination of matrices that approximates the similarity matrix of the labels and use the minimum number of redundant channels. Although the proposed channel ranking and selection approach performs well with regards to accuracy, it is slow. Some of the computational bottlenecks include finding the eigenvector and using DTW as the distance measure.

## 3 Proposed Methods

Let $X \in R^{n \times d \times l}$ be an MTS dataset and $y$ the labels of the time series in the dataset. We denote by $n$ the number of time series in the dataset, $d$ the number of channels in the multivariate time series and $l$ the time series length. In this paper, we only consider fixed-length time series datasets.

The proposed channel selection method makes use of class centroids as representatives of the classes. Let $X_A = \{t \in X \mid y(t) == A\}$ be the subset of $X$ that contains only samples from class A. The centroid of class A is computed as the average of time series in that class:

$$C_A[i,j] = \frac{\sum_{k=1}^{k=m} X_A[k,i,j]}{m}$$

where $m$ is the number of samples in class $A$. The multi-channel centroid $C_A$ is a $d \times l$ matrix in which each row $C_{A,i}$ is the centroid of class $A$ for channel $i$.

The centroid-driven channel selection technique computes the distance matrix for every pair of class centroids, for each channel, using a distance function($\Delta$) discussed later in the section. For a dataset with $r$ classes, the total number of pairs of class centroids is $\frac{r*(r-1)}{2}$. For instance, the distance matrix for a 4-channel dataset with four classes is shown in Table 1, where channel *RElbow* has the highest distance (166.99) for the pair of class centroids $C_n$ and $C_r$. In our proposed method, we examine this distance matrix to select the channels that are most likely to be useful. The idea is that channels with larger distances between class centroids are more likely to be discriminative, since centroids behave like prototypes for time series in those classes. In this example, the distance between class centroids $C_n$ and $C_r$ is highest for channel *RElbow* therefore this channel is more likely to be useful in separating these classes than the other channels, while channel $Nose$ has small distances for all class pairs and so does not seem to be useful to separate any classes. Our method has three components: a distance measure used to compare centroids, an elbow cut heuristic used to threshold the ranked list of channels and three channel selection strategies.

**Distance Metric.** In our current work we use euclidean distance to calculate the distance ($\Delta$) between the centroid pairs for each channel. The Euclidean distance is measured as the $l2$ norm of the difference between the centroids.

$$\Delta(C_{A,i}, C_{B,i}) = \|C_{A,i} - C_{B,i}\|$$

**Table 1.** Illustration of a distance matrix with 4 channels and 4 classes: $a, arch, n, r$. More details about this dataset are provided in the case study described in Section 5.

| $Channels$ | $\Delta(C_a, C_{arch})$ | $\Delta(C_a,C_n)$ | $\Delta(C_a,C_r)$ | $\Delta(C_{arch},C_n)$ | $\Delta(C_{arch},C_r)$ | $\Delta(C_n,C_r)$ |
|---|---|---|---|---|---|---|
| Nose | 15.93 | 11.13 | 14.90 | 14.20 | 14.60 | 16.93 |
| RElbow | 34.62 | 48.95 | 157.12 | 33.04 | 153.80 | 166.99 |
| RHeel | 29.66 | 39.84 | 9.15 | 16.65 | 25.86 | 35.88 |
| LWrist | 38.557 | 42.95 | 148.48 | 47.40 | 155.56 | 157.55 |

**Channel Selection Strategies.** We propose and evaluate three different strategies for channel selection to identify useful channels.

- **KMeans.** This strategy applies k-means clustering with $k = 2$ on the distance matrix to segregate the channels. Every channel (row from distance matrix) is assigned to one cluster. The cluster centroid represents the mean distance of channels across every class pair. Thus, the mean of the cluster centroid acts as a discriminating criterion. We select the channels from the cluster whose centroid-mean is greater than the other centroid-mean, meaning that this cluster contains channels with higher separation distance, while the other cluster contains noisy channels.
- **Elbow Class Sum (ECS).** From the distance matrix, we sum all the pairwise distances for each channel (sum each row). The sum of the distances is sorted in descending order, and an elbow-point is retrieved using the elbow-cut approach described below. All the channels with a distance higher than that of the elbow point are selected as the relevant channels. A single large centroid-pair distances can bias this type of channel selection, favouring channels that separate two classes clearly, but may not be useful for separating other classes.
- **Elbow Class Pairwise (ECP).** The second strategy, ECS, can be biased towards channels that are useful for separating only a few classes. An alternative strategy iterates through every class pair, selects the best set of channels for that pair and finally takes the union of channels over all pairs. This eliminates the potential bias found in the previous strategy. In some cases, this can lead to selecting all the channels, however, there were only few instances of this behaviour in the UEA dataset.

**Elbow Cut.** The elbow cut method [14] is a method to determine a point in a curve where significant change can be observed, e.g., from a steep slope to almost flat curve. This point is often referred to as the elbow or the knee point. This is a well-known method to determine the best number of clusters when doing clustering. We apply it here to separate useful channels from noisy channels. An algorithm takes as input the sorted distances corresponding to channels and returns the elbow point. The elbow is the point at the highest distance($d$) from the line($b$) joining the initial and ending point as shown in Figure 2. The distance $d$ to any point on the distance curve is calculated as

$d = |p - (p.\hat{b})\hat{b}|$ where $\hat{b} = \frac{b}{||b||}$ and $p.\hat{b}$ is the projection of $p$ onto $\hat{b}$. The elbow-point is then elbow $= argmax(d)$. The channels that come before the *elbow* are selected as useful channels for classification and the smaller dataset with this subset of channels is used for the classification step (see Algorithm 1). It is clear from Figure 2 that the elbow point can be relaxed thus allowing a trade-off between data storage and the accuracy of classification. In our work we use the first elbow point which corresponds to channel RShoulder and select only the channels before this point.

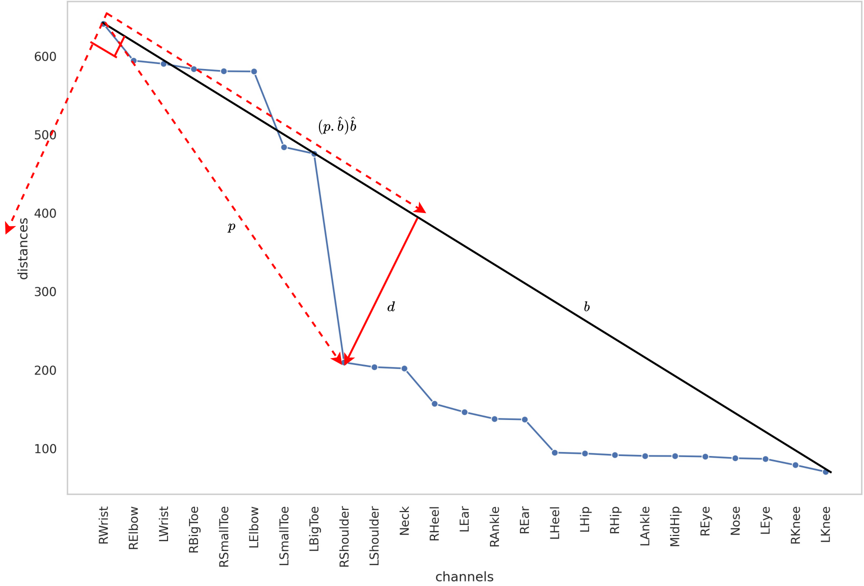


**Fig. 2.** Elbow-point channel selection. All the channels up to RShoulder are selected.

## 4 Evaluation

All the experiments were conducted using the popular Python library sktime [12]. Our primary objective in designing experiments is to understand the relative gain or loss in computational aspects of MTSC algorithms using the proposed channel selection strategies for data reduction. We release all our code in our Github repository[2].

### 4.1 Datasets

The UEA/UCR Time Series Classification archive [2] is a collection of univariate and multivariate time series data. The repository contains 30 multivariate

[2] https://github.com/mlgig/Channel-Selection-MTSC

**Algorithm 1:** Channel Selection for an MTSC dataset.

```
Input: Train dataset: X,y
Output: Selected channels
Initialization;
For each channel in X and each class, compute class centroid ;
Compute distance matrix for all pair of centroids;
if Channel-Selection is KMeans;
    Create 2 clusters using KMeans;
    Selected channels = cluster with higher centroid mean;
elif Channel-Selection is ECS ;
    Sum the distance matrix by rows;
    Rank channels by sum distance;
    Find the elbow on the ranked channels;
    Selected channels = channels with sum distances > elbow point ;
elif Channel-Selection is ECP;
    For each pair of classes;
        Rank the channels by the distances in the corresponding column of the
          distance matrix;
        Find the elbow on the ranked channels;
        Selected channels = Selected channels ∪ channels with class pair-wise
          distance > elbow point;
Return Selected channels;
```

datasets from a variety of application domains, e.g., ECG, motion classification, spectra classification. These heterogeneous datasets vary regarding the number of channels (from 2 to 1,345), number of time series (12 to 30,000) and time series length (8 to 17,894). Here we work with the subset of 26 datasets with equal-length time series.

### 4.2 MTSC Algorithms

All algorithms described in Section 2, except ROCKET, utilise all the channels from the MTSC datasets. Table 2 gives the hyperparameter settings used for all the classifiers in this study.

**Table 2.** Hyperparamter setting used for various SOTA methods

| Classifiers | Hyperparameter-setting |
|---|---|
| WEASEL-MUSE | MUSE(random_state=0) |
| MrSEQL-SAX | MrSEQLClassifier(seql_mode=fs, symrep= ['sax']) |
| ROCKET | Rocket(random_state=0) |
| ROCKET* | ROCKET with all channels |
| 1NN-DTW | KNeighborsTimeSeriesClassifier(n_neighbors=1, distance=”dtw”) |

Table 3 presents the results of these MTSC algorithms on the 26 datasets when no explicit channel selection is implemented. The time is shown in hours and is the total time taken by the algorithm for training and prediction. We observe that ROCKET and WEASEL-MUSE have almost similar mean accuracy, however ROCKET is much faster. ROCKET implements a random channel selection strategy which allows it to keep the runtime bounded, no matter how many channels the dataset has; we discuss this in more detail later in this section. ROCKET also uses a multi-threaded implementation, while all the other algorithms are single-thread implementations, hence the significant difference in runtime. The baseline 1NN-DTW is the least accurate and the slowest method among the four. MrSEQL-SAX does not use the SFA representations in this study, due to the SFA implementation in sktime being too computationally expensive, so its accuracy is lower than ROCKET and WEASEL-MUSE in this experiment. Both WEASEL-MUSE and MrSEQL are impacted by the runtime taken by the symbolic transform, while 1NN-DTW is impacted by the DTW computation. In the Appendix we provide detailed results with each algorithm on each dataset.

**Table 3.** Mean Accuracy and total time of SOTA on 26 UEA MTSC datasets.

| Classifier | Accuracy | Time(in hrs) |
|---|---|---|
| ROCKET | 71.59 | 0.1 |
| WEASEL-MUSE | 70.28 | 73.22 |
| MrSEQL-SAX | 66.99 | 141.40 |
| 1NN-DTW | 65.38 | 152.07 |

### 4.3 MTSC with Channel Selection

Our proposed channel selection strategies (KMeans, ECS, and ECP) are evaluated on the same 26 datasets as above. The channel selection algorithm is run before the MTSC algorithm and it typically results in a reduced dataset for training/testing. We investigate how these strategies impact the classification accuracy, running time (training and testing) and data storage size.

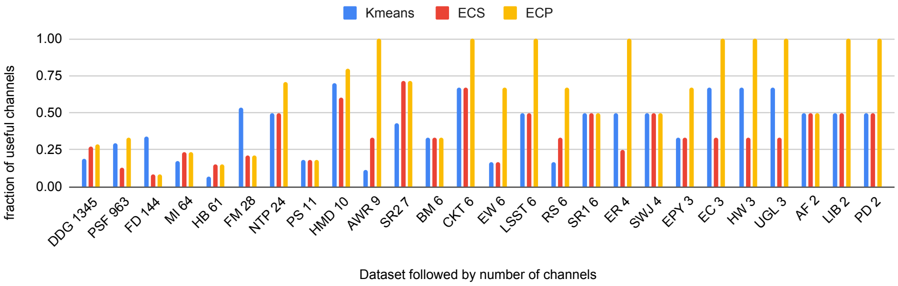


**Fig. 3.** Fraction of channels selected by each of three channel selection strategies.

**Ratio of Channels Selected.** Figure 3 reports the ratio of channels selected by our methods for each dataset (1.0 means no channel is discarded). The acronyms and details of the datasets can be found in the Appendix. The ECP ratios appear to be higher in general, as expected and mentioned in Section 3. However, this mostly occurs in datasets with a small number of channels (the right side of Figure 3). On the left side, where the large numbers of channels can become an issue, ECP appears to be just as efficient as the other methods. We also observe that all three methods are more effective on datasets with a larger number of channels that usually pose a significant scalability challenge to existing MTSC algorithms.

In Table 4 we show the total time taken by the three channel selection strategies. This includes the time taken by each method to compute centroids and create the distance matrix. All three techniques are run only on the training dataset and the output is a selected subset of channels. Since these methods only require the distance matrix for centroid-pairs, they are extremely fast even for large datasets, as the time complexity is only affected by the number of classes, and not by the number of samples. The subset of selected channels is then used to create a reduced dataset as input to MTSC algorithms.

**Table 4.** Total time taken by three channel selection strategies on 26 UEA datasets.

| Channel Selection Strategy | KMeans | ECS | ECP |
|---|---|---|---|
| Total Time (minutes) | 0.34 | 0.33 | 0.35 |

**Table 5.** Loss/Gain in mean accuracy ($\Delta$Acc) vs percentage time saved (%Time) with respect to All channels (Table 3) for our three channel selection techniques on 26 UCR datasets. The red and blue color indicates loss and gain in accuracy respectively. Higher value for %Time or %Storage indicates more time or storage saved.

| Channel Selection→ | KMeans | ECS | ECP |
|---|---|---|---|
| Classifiers↓ | $\Delta$Acc \| %Time | $\Delta$Acc\|%Time | $\Delta$Acc \| %Time |
| ROCKET | -4.01 \| 33.62 | -4.40 \| 29.23 | +0.13 \| 21.43 |
| WEASEL-MUSE | -4.63 \| 70.46 | -3.80 \| 79.90 | -1.53 \| 73.21 |
| MrSEQL-SAX | -3.33 \| 72.68 | -3.80 \| 84.00 | +0.45 \| 77.06 |
| 1NN-DTW | -4.28 \| 68.30 | -6.08 \| 68.82 | +0.67 \| 44.80 |
| Mean $\Delta Acc$\| Mean %Time | -4.06 \| 61.26 | -4.52\| 65.48 | -0.07 \| 54.12 |
| Mean %Storage Saved | 73.95% | 82.59% | 74.38% |

**Performance of Channel Selection.** Table 5 shows the change in accuracy and the percentage of time saved by the MTSC algorithms when run on the reduced datasets after applying the three channels selection strategies.

The comparison reveals that there is a massive gain in computation time for a minimal drop in accuracy. The time taken to find the subset (Table 4) is insignificant in comparison. Out of the three channels selection strategies, ECP seems to be the best choice for channel selection. It significantly reduces the computation time and at the same time eliminates noisy channels, thus increasing the accuracy for ROCKET, 1NN-DTW and MrSEQL-SAX. The method WEASEL-MUSE takes a small hit on accuracy (1.5%), at the benefit of saving 73.21% in runtime. Considering that WEASEL-MUSE requires 73.2h to complete training and prediction on this benchmark (see Table 3), this is a significant time saving. A similar result holds across all classifiers, and all channel selection strategies: for a small loss in accuracy, there is a high gain in runtime. In the case of ECP, the accuracy is preserved or even increased, with a significant saving in runtime. We also calculate the average amount of memory saved by the channel selection techniques over the 26 datasets. The comparison of dataframe size in memory, before and after channel selection is used to compute these values. Overall, this MTSC archive uses about 1.6Gb memory and when using our channel selection strategies, this is reduced to less than 30% of the original size. When stored on disk this dataset is about 3.3Gb total, and with the channel selection techniques this is reduced to about 900Mb.

### 4.4 Effectiveness of Channel Selection

In this experiment we test whether our best strategy (ECP) selects useful channels and how good the selection is compared to selecting optimal channel subsets.

**Optimal Channel Subset Selection.** We evaluate every possible subset of channels on the test set to discover the optimal subset. Naturally, this brute-force approach is very expensive and impractical for datasets with a high number of channels as the possible combination for a dataset with $d$ channels will be $2^d - 1$. Nevertheless, in this study, all the subsets for datasets with a number of channels <4 are analysed. These datasets are: *AtrialFibrillation, Libras, PenDigits, EthanolConcentration, Epilepsy, Handwriting, UWaveGestureLibrary.* In this experiment, we choose the state-of-the-art ROCKET classifier to quickly evaluate all the subsets. However, because ROCKET internally randomly samples the channels, it can select a good subset by chance and mask the issue of selecting bad channels. Therefore, we modify its code to get ROCKET to use all channels in each kernel, i.e., we use the ROCKET* variant. By doing so, the impact of a good channel subset and a bad channel subset on classification accuracy becomes more pronounced.

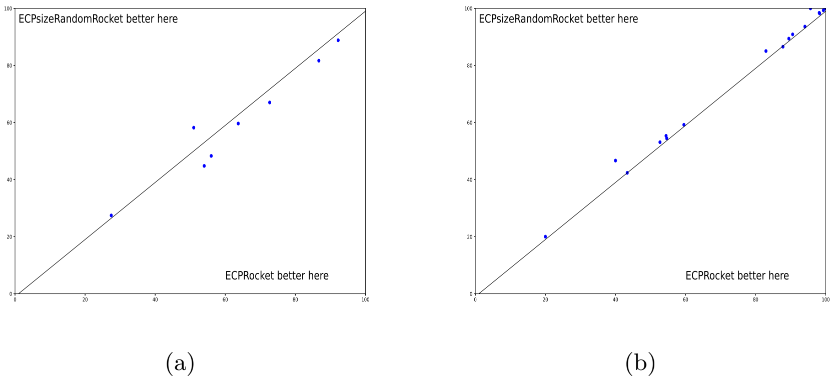
**Table 6.** Accuracy of ROCKET* on datasets with channels < 4. Bold indicates the optimal subset. Underscore indicates the subset selected by ECP. Empty spaces are for datasets with less channels, e.g., dataset AF only has 2 channels, 0 and 1.

| DT | 0 | 1 | 2 | (0,1) | (0,2) | (1,2) | (0,1,2) |
|---|---|---|---|---|---|---|---|
| AF | **20** | 6.67 | | 13.33 | | | |
| LB | 73.9 | 77.78 | | **93.89** | | | |
| PD | 89.59 | 88.45 | | **98.26** | | | |
| EC | 54.4 | 49.8 | **53.6** | 38.0 | 44.9 | 39.2 | 36.1 |
| EP | 97.82 | **100** | 94.93 | 98.55 | 98.55 | 97.83 | 99.28 |
| HW | 38.12 | 32.35 | 42.12 | 45.76 | 59.76 | 50.35 | **57.06** |
| UW | 79.37 | 71.25 | 71.88 | 87.5 | 93.12 | 84.06 | **93.75** |

Table 6 shows that ECP successfully identified the optimal subset five out of seven times. With the Epilepsy (EP) problem, it also correctly identified channel

0 as a potential issue (the classification accuracy is only 97.82% with channel 0 alone) and excluded it from the selection. However, for this dataset it seems to be better to use either only channel 1 or all the channels. It is important to remind the reader that this setting is evaluated directly on the test data, and in practice we do not have perfect knowledge of the best subset of channels for the test data. ECP selects this channels based on the training data alone, and it seems to be effective at finding the useful channels for each task using only training data.

**Random Channel Subset Selection.** In order to further understand the effect of the ECP channel selection method, we compare the accuracy of the ROCKET classifier, when using channels selected with different strategies. We compare ECPRocket (ECP combined with ROCKET) with ECPsizeRandomRocket, a simple baseline where the number of channels is set using ECP, but the actual channels are picked randomly. We repeated the experiment 10 times for each dataset and report the average accuracy in Figure 4. We observe that for the majority of the large datasets (number of channels $> 10$), ECPRocket is better, while for datasets with less number of channels (number of channels $\leq 10$) the ECPSizeRandomRocket works similar to ECPRocket. Note that for half of the datasets with number of channels $\leq 10$, ECP does not reduce the number of channels (i.e., it keeps all the channels as shown in Figure 3), hence the two variants ECPRocket and ECPSizeRandomRocket simply reduce to ROCKET, since ECP has no effect in this case. For datasets with a higher number of channels, ECP often reduces the full channel set to a subset of good channels, and the variant ECPRocket constrains ROCKET to work with this pool of good channels, resulting in storage savings and improvements in accuracy. Hence, for either small or large number of channels, ECP is fast and leads to storage savings without resulting in loss of accuracy.



(a)

(b)

**Fig. 4.** Comparison of ECPRocket with ECPsizeRandomRocket. Figure 4(a) represents datasets with number of channels $> 10$ and Figure 4(b) represents datasets with number of channels $<=10$.

# 5 Case Study: Channel Selection for the Military Press MTSC Dataset

## 5.1 Dataset

A total of 56 healthy volunteers (34 males and 22 females; age: 26 ± 5 years, height: 1.73 ± 0.09 m and body mass: 72 ± 15 kg) participated in a study aimed at analysing the execution of the Military Press strength and conditioning exercise. The participants completed ten repetitions of the normal form and ten repetitions of induced forms. The NSCA guidelines were applied under the guidance of sports physiotherapists and conditioning coaches to ensure standardisation. The dataset was extracted from the video of individuals performing the exercise with the help of the human body pose estimation OpenPose[3]. There are four classes in the dataset, namely: Normal (N), Asymmetrical (A), Reduced Range (R) and Arch (arch). The N refers to the correct execution of the exercise; A refers to when the barbell is lopsided and asymmetrical, R refers to the form where the bar is not brought down completely to the shoulder level and Arch refers to when participants arch their back. A total of 25 body parts were tracked, as seen in Figure 2. These 25 body parts act as channels for the MTSC task. The train and test size for this dataset is 1452 and 601 respectively and the length of time-series is 160.

## 5.2 Channel Selection

Table 7 illustrates the selected channels for the Military Press dataset. The Elbows and Wrists are actively involved in the exercise, as the participant is required to lift a barbell over the shoulders. However, the Toes do not seem to contribute to the exercise. We tried to investigate this and think that the issue might be related to data pre-processing when the time series is extracted from the video; investigating this aspect further is interesting but outside the scope of this study.

**Table 7.** Channel Selection using our three strategies. All strategies select the same 8 body parts as relevant for this classification task.

| Channel Selection | Body Parts |
|---|---:|
| KMeans | Elbows, Wrists, BigToes, SmallToes |
| ECS | Wrists, Elbows, BigToes, SmallToes |
| ECP | Elbows, Wrists, BigToes, SmallToes |

## 5.3 Results & Discussion

Table 8 reports the results for ECP with different SOTA MTSC classifiers. ROCKET is the fastest and most accurate classifier in this experiment. The data normalisation which is turned on by default in ROCKET, is turned off

[3] https://github.com/CMU-Perceptual-Computing-Lab/openpose

in the current experiment. This is due to the fact that the signal magnitude contains important information for this task, so normalisation should not be used in this case. For WEASEL-MUSE and MrSEQL-SAX, data normalisation is done internally in the algorithm during the symbolic transform (SFA/SAX), so we cannot de-activate the data normalisation step. This affects the accuracy of these methods in this task, since the magnitude of the signal is important to differentiate between classes. As in the previous experiments, in the case study we also find that ECP saves a large amount of time and memory, with minimal or no loss in accuracy. For WEASEL-MUSE, it saves about 71.6% of computation time, while for MrSEQL-SAX and 1NN-DTW it saved about 74% and 68%, respectively. Moreover, the memory required for computation is reduced to 32%, thus a saving of 68% on the original dataset.

**Table 8.** Performance of ECP on the Military Press Exercise.

| | Accuracy | Time (minutes) |
|---|---|---|
| Classifiers | ECP \| All | ECP\| All |
| ROCKET | 76.26 \| 77.53 | 2.14 \| 2.25 |
| WEASEL-MUSE | 57.57 \| 57.57 | 30.29 \| 107.02 |
| MrSEQL-SAX | 58.23 \| 61.56 | 139.53 \| 516.79 |
| 1NN-DTW | 48.58 \| 47.25 | 10.39 \| 24.36 |
| Data Size (MB) Reduced\|Original | 15.77 \| 49.29 | |

## 6 Conclusion

In this study we have shown that not all the channels for MTSC are helpful. Data noise in the form of uninformative channels can prevent the classifier from achieving its maximum potential. We have observed that channel selection can remove some of the noise and drastically reduce the required computation time for existing MTSC methods. In the current study, we showed that the distance between the class centroids of various channels plays a crucial role in identifying the noisy channels. Our three-channel selection strategies ECP, ECS and Kmeans, can select the useful channels based on this distance. All three techniques significantly reduced the runtime and memory required to run SOTA classifiers. The ECS and KMeans techniques also reduced the accuracy, while ECP resulted in accuracy gains for MrSEQL-SAX, ROCKET and 1NN-DTW and marginal accuracy loss for WEASEL-MUSE. We believe that with a more robust elbow selection heuristic the performance can be improved further. Our channel selection techniques significantly reduced the data size on disk for most of the MTSC datasets, thus enabling significant storage savings for large MTSC datasets where several channels are not useful for the classification task.

## Acknowledgments

This publication has emanated from research supported in part by a grant from Science Foundation Ireland through the VistaMilk SFI Research Centre

(SFI/16/RC/3835) and the Insight Centre for Data Analytics (12/RC/2289_P2). For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. We would like to thank the reviewers for their constructive feedback. We would like to thank all the researchers that have contributed open source code and datasets to the UEA MTSC Archive and especially, we want to thank the groups at UEA and UCR who continue to maintain and expand the archive.

## References

1. Consumer enthusiasm for wearable devices drives the market to 28.4% growth in 2020 (2021), https://www.idc.com/getdoc.jsp?containerId=prUS47534521
2. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **31**, 606–660 (2017)
3. Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y.A.: Openpose: Realtime multi-person 2d pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
4. Dempster, A., Petitjean, F., Webb, G.I.: Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery pp. 1–42 (2020)
5. Dhariyal, B., Le Nguyen, T., Gsponer, S., Ifrim, G.: An examination of the state-of-the-art for multivariate time series classification. In: 2020 International Conference on Data Mining Workshops (ICDMW). pp. 243–250 (2020). https://doi.org/10.1109/ICDMW51313.2020.00042
6. Han, S., Niculescu-Mizil, A.: Supervised feature subset selection and feature ranking for multivariate time series without feature extraction. arXiv preprint arXiv:2005.00259 (2020)
7. Hu, B., Chen, Y., Zakaria, J., Ulanova, L., Keogh, E.: Classification of multidimensional streaming time series by weighting each classifier's track record. In: 2013 IEEE 13th International Conference on Data Mining. pp. 281–290 (2013). https://doi.org/10.1109/ICDM.2013.33
8. Kathirgamanathan, B., Cunningham, P.: A feature selection method for multidimension time-series data. In: International Workshop on Advanced Analytics and Learning on Temporal Data. pp. 220–231. Springer (2020)
9. Krzanowski, W.: Between-groups comparison of principal components. Journal of the American Statistical Association **74**(367), 703–707 (1979)
10. Le Nguyen, T., Gsponer, S., Ilie, I., O'Reilly, M., Ifrim, G.: Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. Data Mining and Knowledge Discovery **33**(4), 1183–1222 (2019)
11. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. Data Mining and Knowledge Discovery **15**(2), 107–144 (Oct 2007). https://doi.org/10.1007/s10618-007-0064-z, https://doi.org/10.1007/s10618-007-0064-z
12. Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., Király, F.J.: sktime: A unified interface for machine learning with time series. arXiv preprint arXiv:1909.07872 (2019)

13. Ruiz, A.P., Flynn, M., Large, J., Middlehurst, M., Bagnall, A.: The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery pp. 1–49 (2020)
14. Satopaa, V., Albrecht, J., Irwin, D., Raghavan, B.: Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In: 2011 31st international conference on distributed computing systems workshops. pp. 166–171. IEEE (2011)
15. Schäfer, P., Högqvist, M.: Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In: Proceedings of the 15th international conference on extending database technology. pp. 516–527 (2012)
16. Schäfer, P., Leser, U.: Multivariate time series classification with weasel+ muse. ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (AALTD18), arXiv preprint arXiv:1711.11343 (2017)
17. Shokoohi-Yekta, M., Wang, J., Keogh, E.J.: On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In: SDM (2015)
18. Singh, A., Le, B.T., Le Nguyen, T., Whelan, D., O'Reilly, M., Caulfield, B., Ifrim, G.: Interpretable classification of human exercise videos through pose estimation and multivariate time series analysis. In: 5th International Workshop on Health Intelligence(W3PHIAI-21) at AAAI21. Springer (2021)
19. Yoon, H., Yang, K., Shahabi, C.: Feature subset selection and feature ranking for multivariate time series. IEEE transactions on knowledge and data engineering **17**(9), 1186–1198 (2005)

# Appendix

**Table 9.** Detailed description for the 26 MTSC datasets used in this study.

| Dataset | Acronym | TrainSize | TestSize | NumChannels | SeriesLength | NumClasses | ClassCounts |
|---|---|---|---|---|---|---|---|
| ArticularyWordRecognition | AWR | 275 | 300 | 9 | 144 | 25 | 11 |
| AtrialFibrillation | AF | 15 | 15 | 2 | 640 | 3 | 5 |
| BasicMotions | BM | 40 | 40 | 6 | 100 | 4 | 10 |
| Cricket | CKT | 108 | 72 | 6 | 1197 | 12 | 9 |
| DuckDuckGeese | DDG | 50 | 50 | 1345 | 270 | 5 | 10 |
| EigenWorms | EW | 128 | 131 | 6 | 17984 | 5 | 55 |
| Epilepsy | EP | 137 | 138 | 3 | 206 | 4 | 34 |
| ERing | ER | 30 | 270 | 4 | 65 | 6 | 5 |
| EthanolConcentration | EC | 261 | 263 | 3 | 1751 | 4 | 65 |
| FaceDetection | FD | 5890 | 3524 | 144 | 62 | 2 | 2945 |
| FingerMovements | FM | 316 | 100 | 28 | 50 | 2 | 159 |
| HandMovementDirection | HMD | 160 | 74 | 10 | 400 | 4 | 40 |
| Handwriting | HW | 150 | 850 | 3 | 152 | 26 | 8 |
| Heartbeat | HB | 204 | 205 | 61 | 405 | 2 | 57 |
| Libras | LB | 180 | 180 | 2 | 45 | 15 | 12 |
| LSST | LSST | 2459 | 2466 | 6 | 36 | 14 | 34 |
| MotorImagery | MI | 278 | 100 | 64 | 3000 | 2 | 139 |
| NATOPS | NTP | 180 | 180 | 24 | 51 | 6 | 30 |
| PEMS-SF | PSF | 267 | 173 | 963 | 144 | 7 | 32 |
| PenDigits | PD | 7494 | 3498 | 2 | 8 | 10 | 780 |
| PhonemeSpectra | PS | 3315 | 3353 | 11 | 217 | 39 | 85 |
| RacketSports | RS | 151 | 152 | 6 | 30 | 4 | 39 |
| SelfRegulationSCP1 | SR1 | 268 | 293 | 6 | 896 | 2 | 135 |
| SelfRegulationSCP2 | SR2 | 200 | 180 | 7 | 1152 | 2 | 100 |
| StandWalkJump | SWJ | 12 | 15 | 4 | 2500 | 3 | 4 |
| UWaveGestureLibrary | UW | 120 | 320 | 3 | 315 | 8 | 15 |

**Table 10.** The amount of memory (MB) used by each dataset when using all channels and after applying our channel selection strategies.

| Dataset | OriginalSize | KMeansReduced | ECSReduced | ECPReduced | KMeansSaved% | ECSSaved% | ECPSaved% | Channels |
|---|---|---|---|---|---|---|---|---|
| DuckDuckGeese | 147.25 | 28.03 | 40.40 | 42.37 | 80.97 | 72.56 | 71.23 | 1345 |
| PEMS-SF | 315.83 | 92.81 | 41.00 | 104.29 | 70.61 | 87.02 | 66.98 | 963 |
| FaceDetection | 511.20 | 173.95 | 42.60 | 42.60 | 65.97 | 91.67 | 91.67 | 144 |
| MotorImagery | 409.53 | 70.39 | 95.98 | 95.98 | 82.81 | 76.56 | 76.56 | 64 |
| Heartbeat | 40.06 | 2.63 | 5.91 | 5.91 | 93.44 | 85.25 | 85.25 | 61 |
| FingerMovements | 4.52 | 2.42 | 0.97 | 0.97 | 46.43 | 78.57 | 78.57 | 28 |
| NATOPS | 2.24 | 1.12 | 1.12 | 1.59 | 50.00 | 50.00 | 29.17 | 24 |
| PhonemeSpectra | 65.10 | 11.84 | 11.84 | 11.84 | 81.82 | 81.82 | 81.82 | 11 |
| HandMovementDirection | 5.09 | 3.56 | 3.05 | 4.07 | 30.00 | 40.00 | 20.00 | 10 |
| ArticularyWordRecognition | 3.04 | 0.34 | 1.01 | 3.04 | 88.89 | 66.66 | 0.00 | 9 |
| SelfRegulationSCP2 | 12.49 | 5.35 | 8.92 | 8.92 | 57.14 | 28.57 | 28.57 | 7 |
| BasicMotions | 0.21 | 0.07 | 0.07 | 0.07 | 66.63 | 66.63 | 66.63 | 6 |
| Cricket | 6.00 | 4.00 | 4.00 | 6.00 | 33.33 | 33.33 | 0.00 | 6 |
| EigenWorms | 105.47 | 17.58 | 17.58 | 70.32 | 83.33 | 83.33 | 33.33 | 6 |
| LSST | 5.97 | 2.98 | 2.98 | 5.97 | 50.00 | 50.00 | 0.00 | 6 |
| RacketSports | 0.32 | 0.05 | 0.11 | 0.22 | 83.30 | 66.64 | 33.32 | 6 |
| SelfRegulationSCP1 | 11.20 | 5.60 | 5.60 | 5.60 | 50.00 | 50.00 | 50.00 | 6 |
| ERing | 0.08 | 0.04 | 0.02 | 0.08 | 49.92 | 74.88 | 0.00 | 4 |
| StandWalkJump | 0.92 | 0.46 | 0.46 | 0.46 | 49.99 | 49.99 | 49.99 | 4 |
| Epilepsy | 0.70 | 0.23 | 0.23 | 0.47 | 66.66 | 66.66 | 33.33 | 3 |
| EthanolConcentration | 10.56 | 7.04 | 3.52 | 10.56 | 33.33 | 66.67 | 0.00 | 3 |
| Handwriting | 0.58 | 0.39 | 0.19 | 0.58 | 33.33 | 66.65 | 0.00 | 3 |
| UWaveGestureLibrary | 0.91 | 0.61 | 0.30 | 0.91 | 33.33 | 66.66 | 0.00 | 3 |
| AtrialFibrillation | 0.15 | 0.08 | 0.08 | 0.08 | 49.96 | 49.96 | 49.96 | 2 |
| Libras | 0.17 | 0.09 | 0.09 | 0.17 | 49.96 | 49.96 | 0.00 | 2 |
| PenDigits | 2.86 | 1.43 | 1.43 | 2.86 | 50.00 | 50.00 | 0.00 | 2 |