Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operations Research

Does Deep Neural Network Time Series Clustering Outperform Partition-based Clustering with Elastic Distance Functions?

Willem Amesz (599668)

Supervisor:	J.Durieux
Second Assessor:	M.Welz
Date final version:	1st July 2024

Abstract

This paper aims to answer the question if deep neural networks can outperform elastic distance based clustering. To compare the distance based algorithms, a replication study of the experiments in Holder, Middlehurst and Bagnall (2024) is performed. Clustering metrics such as normalised mutual information (NMI) and clustering accuracy are used to evaluate algorithms based on nine elastic distance measures, along with an unsupervised Convolutional Neural Network (CNN). For the distance based algorithms, a distinction is made between k-means and k-medoids clustering algorithms and the differences are compared. These algorithms are used on 61 univariate datasets from the UCR archive. Wilcoxon signed ranked tests are performed to validate the significance of the methods. Results show that the move-split-merge (MSM) distance along with the edit distance with real penalty (ERP) are the best performing algorithms. Clustering algorithm k-medoids improves over k-means for almost every distance function. Both of these results are in line with the results of Holder et al. (2024). The CNN outperforms over half of the distance based algorithms, such as Dynamic Time Warping (DTW) and Euclidean Distance, but falters in light of the MSM and ERP distance algorithms. This is likely due to the use of smaller datasets and the unsupervised setting of the neural network. Neural networks show a promising avenue for time series clustering, but face challenges related to the unsupervised setting and need to be improved to outperform top-distance based algorithms.

1 Introduction

Clustering is an indispensable technique in data analysis and it serves as a cornerstone for understanding patterns within datasets. As data storage capacities and processing power continue to grow, real-world companies and institutions are now able to retain data over extended periods. Consequently, many applications now store information as time series data, such as sales figures, stock market prices, exchange rates and more. As more data is stored as time series, specific analyses can be performed on these time series data, including data mining methods such as clustering.

Time series clustering (TSCL) is an unsupervised method of grouping time series data. This suggests that there are no labels tied to the time series data. There are various methods of clustering, but this paper specifically delves deeper into two branches of the TSCL literature. First of all, this includes partition based algorithms that can use a variety of distance functions. These distance functions often entail a form of reorganisation of time series and can often enhance clustering experiments. An extensive review of set distance functions and their evaluation is given in Holder et al. (2024). For in their paper, they investigate nine different elastic distance measures and evaluate them using several clustering metrics. These algorithms include well-known distances, such as Dynamic Time Warping (DTW), and other commonly used measures, including Time Warp Edit (TWE) and Move-Split-Merge (MSM). The algorithms that are paired with these distances are clustering algorithms k-means and k-medoids, which are popular in the literature. The point of interest for Holder et al. (2024) is to determine which algorithm, k-means or k-medoids works best for TSCL, as well as which elastic distance of the nine that are selected, results in the best clustering performance. One of the goals of this paper is to perform similar experiments as Holder et al. (2024) and investigate the validity of their results.

The other branch of TSCL this paper this paper explores is deep learning. Specifically, this includes Unsupervised Deep Neural Networks (UDNN) that can be used for TSCL. This concerns a different type of clustering that with the rising popularity of machine learning, becomes interesting. Notably, neural networks, such as Convolutional Neural Networks (CNN) are often not associated with clustering, since clustering is an unsupervised procedure. On the other hand, neural networks can also be used in this setting, through methods explained in Lafabregue, Weber, Gançarski and Forestier (2022), who experiment with a large number of DNNs to examine which performs best. Since the partitional based clustering algorithms with elastic distance functions have not been compared to current deep learning methods, this paper delves deeper into the comparison of the two by using multiple clustering metrics, such as clustering accuracy and normalised mutual information. This paper essentially combines ideas from Holder et al. (2024) and Lafabregue et al. (2022).

The main research question can therefore be stated as follows: "Does Deep Neural Network Time Series Clustering Outperform Partition-based Clustering with Elastic Distance Functions?". The subquestions that revolve around this question are:

- Do the results from the replication match the paper of Holder et al. (2024)?
- What methods are there to compare neural network time series clustering to partitional clustering methods?
- What is the "best" deep neural network for time series clustering?
- What is the difference in the computation time for each of these experiments? Could one method be preferred, but simply take too long for practical use?

This research is relevant when looking at the validation of the paper that will be replicated. The results of Holder et al. (2024) show that the k-medoids models perform better, as well as a worse performance of DTW and an overall improvement with the MSM distance. Since DTW is a commonly used elastic distance function, verifying this conclusion can be beneficial for further research and use of elastic distance functions in practice, as well as a motivation for researchers to use methods more similar to MSM. Furthermore, in Lafabregue et al. (2022), comparisons are made between deep neural networks and baseline models. However, this only includes the Euclidean Distance, Dynamic Time Warping and Principal Component Analysis with k-means. By comparing the time series clustering with elastic distance functions to a neural network clustering, this paper aims to better understand the taxonomy of TSCL, determining which method of TSCL leads to better results and should therefore lead to a conclusion on which methods should be more widely used in this field. Other types of networks are also briefly discussed and tested, to test the theory of Lafabregue et al. (2022) that a convolutional network based architecture performs best. Lastly, the validity of Lafabregue et al. (2022) is also tested concerning the performance of the neural network compared to the k-means DTW, since their conclusion is that the neural network yields better results compared to "simple" algorithms and distance measures.

The results found in this paper show that k-medoids based experiments perform better on almost all elastic distance measures and MSM and edit distance with real penalty are the best performing distances to go with it. These methods also do not have significantly long runtimes. These results match closely with the results found in Holder et al. (2024). When comparing different neural networks, the CNN based architectures without clustering loss perform better than a recurrent neural network. When CNN is then compared to the distance based algorithms, CNN outperforms the "weaker" algorithms, but it does not outperform the MSM and ERP measures, especially the k-medoids variants are significantly better. The reason why it cannot outperform the better distance based measures might be contributed to data structure and experiment set-up as well as lack of proper parameter tuning.

The remainder of this paper is structured as follows: Section 2 provides a review of the literature on time series clustering and the innovations that have been made in this field. The data will be specified in Section 3, while the methodology is explained in Section 4, including the elastic distance functions, deep neural networks, and clustering metrics. In Section 5 the results of the clustering experiments and evaluation are stated. Finally, in Section 6 the conclusions regarding the results will be shared

2 Literature

2.1 Time Series Classification

Firstly, a distinction between time series classification (TSC) and time series clustering (TSCL) must be made. Both fields of study share some similarities, such as the methods that are used, which include elastic distance functions (Abanda, Mori & Lozano, 2019) and deep learning (Ismail Fawaz, Forestier, Weber, Idoumghar & Muller, 2019) in TSC. However, both methods differ in the goal of experiments and the results of them. The literature of TSC is quite extensive and algorithms have been implemented well before the twenty-first century (Keogh & Kasetty, 2002). More recently however, new algorithms have been proposed in TSC by Bagnall, Lines, Bostrom, Large and Keogh (2017), who also experiment with similar distance measures used in this paper. In short, the two academic research fields share similarities, which can be used to make advancements in both fields.

2.2 Time Series Clustering

Clustering has many different approaches and many algorithms that can be used. The taxonomy of time series clustering can be seen in Figure 1 and can be divided into whole series analysis and preprocessing. In the preprocessing field, the literature is dealing with feature extraction (Räsänen & Kolehmainen, 2009) and principal components (Lafabregue et al., 2022) along with other methods. The focus of this paper is on whole series analysis and that includes the comparison of the partition based and deep learning branches of the taxonomy tree. More information on the methods will be in Section 4.

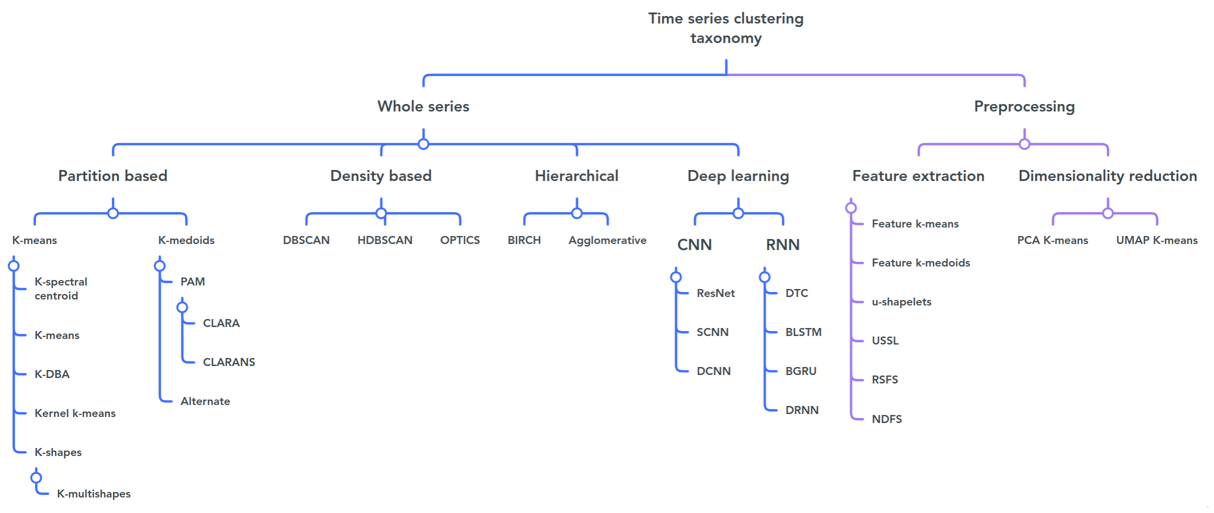


Figure 1: Taxonomy of Time Series Clustering[1]

Lastly, density-based and hierarchical TSCL are different branches that are not discussed in this paper, but are known to work well in the TSCL literature (Ester, Kriegel, Sander, Xu et al., 1996; Zhang, Ramakrishnan & Livny, 1996).

2.3 Elastic Distance measures

Aside from an algorithm for clustering such as k-means or k-medoids, distances between time series are needed for the current study. The ways in which distances are used in k-means and k-medoids are used will be highlighted in the Methodology. There exist multiple distance functions to use, which all alter the way of how distance is defined between time series. All functions used in this report are in Table 1, including relevant literature surrounding these functions. Every distance function builds upon other ideas and distances. For example, DTW was proposed to overcome Euclidean distance limitations such as non-linear distortions. Most of the functions are also used in the experiments of Bagnall et al. (2017), who compare the functions in the TSC field. Since they used these functions in TSC, Holder et al. (2024) thought of implementing them for TSCL as well. They were not the first to do this, however they did make an overview of the literature so far. In Bagnall et al. (2017) the DTW distance is seen as a benchmark model and various other distances perform better than DTW in this paper, including elastic ensemble (EE), which combines multiple of the distances used in this paper. In Holder et al. (2024), MSM emerged as the top-performing distance measure, alongside TWE and ERP, which also showed strong performance. The distances that are used in the paper will be further discussed in Section 4.2.

[1]Source: `https://www.aeon-toolkit.org/en/stable/examples/clustering/clustering.html`

Distance Function	Acronym	Literature
Euclidean Distance	ED	-
Dynamic Time Warping	DTW	Berndt and Clifford (1994)
Derivative Dynamic Time Warping	DDTW	Górecki and Łuczak (2013)
Weighted Dynamic Time Warping	WDTW	Jeong, Jeong and Omitaomu (2011)
Derivative Weighted Dynamic Time Warping	DWDTW	Jeong et al. (2011)
Longest Common Subsequence	LCSS	Paterson and Dančík (1994)
Edit Distance on Real sequences	EDR	Chen, Özsu and Oria (2005)
Edit distance with Real Penalty	ERP	Chen and Ng (2004)
Move-Split-Merge	MSM	Stefan, Athitsos and Das (2012)
Time Warp Edit	TWE	Marteau (2008)

Table 1: Distance functions used in this report

2.4 Deep learning

With the recent rise in the popularity of artificial intelligence, neural networks are now being used in numerous fields, including TSCL. In this field, specifically unsupervised neural networks are used, since clustering is an unsupervised procedure. One of the first to use the neural network approach in TSCL are Wang, Yan and Oates (2017) who introduced the baseline for deep learning experiments. This baseline consists of neural network with convolutional neural networks (CNN) architecture, which provided promising results. A great overview of different deep learning possibilities is given in Lafabregue et al. (2022). This includes recurrent neural networks (RNN) (Bandara, Bergmeir & Smyl, 2020), CNN (Zhao, Lu, Chen, Liu & Wu, 2017) along with many different archetypes and autoencoder set-ups. Another review of the different methods that can be used, can be found in Alqahtani, Ali, Xie and Jones (2021). Interestingly, the TSCL literature often does not favour neural networks that are supposed to work well with time series data. In Bandara et al. (2020) for example, RNN and its derivative long short-term memory (LSTM) variants are discussed. While LSTM is often used for time series analysis, because of the properties it has, for TSCL, it is on par or sometimes worse than other methods, as can also be read in Lafabregue et al. (2022). However, Bandara et al. (2020) concur that LSTM could be improved if the notion of similarity between the time series is accounted for. This could perhaps be possible with more intricate models.

Lastly, since TSCL is a unsupervised part of machine learning, employing neural networks could work differently, since the majority of use cases for neural network are supervised. An example of where a supervised algorithm is used in this field is exlplored by Tavakoli, Siami-Namini, Adl Khanghah, Mirza Soltani and Siami Namin (2020), where they propose a two step method that uses labelled data.

2.5 Contribution

This paper contributes to the literature in the following way: Unsupervised deep neural networks (UDNN) are compared to TSCL with elastic distance functions. This has not been done in the literature before, since Lafabregue et al. (2022) provided only comparisons with the DTW and

ED measures. Aside from this contribution, this paper functions as a confirmation of the results of Holder et al. (2024) and partly also the results of Lafabregue et al. (2022). In effect, these results could lead to more clarity in which field more research is more beneficial and can provide insights. Lastly, a subset of the data is taken, as will be explained in the following section. This means that another contribution of this paper is to validate the results on this smaller database, which includes smaller datasets. This could give insights into performance on specifically smaller datasets.

3 Data

This paper will focus on the time series data originating from the University of California, Riverside (UCR) archive (Dau et al., 2018). This archive comprises 128 datasets spanning various domains and sizes, including some simulated datasets. The datasets are divided into multiple categories such as “Household devices”, “Images” and “Spectograms”. Datasets with missing values or unequal length are dismissed. The data is split into training and testing data for every dataset, which are predetermined.

Along with the datasets with unequal length and missing values, more datasets will be excluded from this paper compared to the analysis performed in Holder et al. (2024). This is due to the size of some of the datasets, which entails either a large number of clusters or a sizeable number of data points in the time series or a combination of both. Through an investigation of runtime during experiments, a rather arbitrary threshold is set for determining which datasets will be included. This involves datasets with at most 10 clusters and a time series length of at most 1000, since run times beyond these thresholds become exponentially long. Applying these conditions results in 61 datasets from the UCR Archive that will be used in experiments. The average number of classes is 2.92. The average size of the training set is 256.2, while the length of the time series is on average 270.37. The UCR archive contains multiple bigger datasets, but these are dismissed in this research.

Next, the question of normalisation arises within this research, since Holder et al. (2024) opts to look at both non-normalised and normalised data, while Lafabregue et al. (2022) use normalised data. The comparison and tests can therefore only be performed on the normalised data. However according to Rakthanmanon et al. (2013), normalised data is better for comparison in this field of study.

Lastly, there exists some discourse surrounding the use of the UCR Archive datasets. Specifically, these are discussed in Hu, Chen and Keogh (2016), where they look at a set of different datasets to perform their experiments on, since they believe that in realistic settings, results derived from UCR archive are limited. Using predefined labels is also a point of discussion. Namely, labels can be put different than what a clustering would come up with, while still clustering accurately. But since the only goal is to compare these different measures, the UCR Archive is a valuable and reliable source in this sense.

4 Methodology

In this section, the methods regarding the distance functions, performance measures and comparison options are displayed, as well as the specific neural networks that are selected. Furthermore specific details surrounding the initialisation and running procedures will be specified.

4.1 Clustering Algorithms

In this paper, the k-means and k-medoids clustering algorithms will be used, since they are used in Holder et al. (2024) and they are the partition based methods that are known to work well in the literature. In order to correctly replicate the paper, k-medoids must be added.

Defined in MacQueen et al. (1967), k-means clustering is a well known clustering algorithm set on minimising the variance of each cluster. TSCL can be seen as partitioning a time series set $T = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}\}$ where a time series $\mathbf{x}$ is a sequence of n observations $(x_1, x_2, \ldots, x_n)$ into k clusters: $\mathbf{T} = \{T_1, T_2, \ldots, T_k\}$. The variance of each cluster is then minimised, as seen in this equation.

$$\arg\min_{\mathbf{T}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in T_i}^{m} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \tag{1}$$

In this equation μ_i is the mean of a cluster, also defined as a centroid. The norm that is taken in this equation is the L2 norm, also known as the Euclidean distance. This paper delves deeper into different distance functions that can be used in this minimization along side the general L2 norm. The question still remains what the k-means algorithm specifically does to solve this minimisation problem.

Firstly, the algorithm needs to be initialised. This includes picking example cases that characterize a cluster. This initialisation stage can be differently executed. Methods include random partition and Forgy's method, along with other more intricate methods, such as the binary-split method (Linde, Buzo & Gray, 1980) and a more robust method called ROBIN (Al Hasan, Chaoji, Salem & Zaki, 2009). The Random Partition method randomly assigns a cluster to each observation, while Forgy's method randomly chooses k observations from the dataset and uses these as the initial means. Important to note is that the number of clusters k is known beforehand in this research, since it is assumed to be equal to the number of classes in the classification dataset for each of the datasets that are analysed. The topic of determining the number of clusters in a data set is a different research subject that is not investigated in this paper.

After the initialisation, an update step is performed, where now that the centroids have been selected, the new centroid means are calculated to find the new centroids. This algorithm terminates eventually, since this process is nonnegative monotonically decreasing. It could be that this takes a considerable amount of time, so a stopping condition can be set.

The difference in k-medoids lies in the update step. Instead of calculating the mean for the observations in the cluster, k-medoids picks the observation that has the minimal distance to the other observations in the cluster. This means that only the distances between observations are needed, compared to new average calculations.

In these experiments, k-means and k-medoids will both be used with random initialisation. The choice of initialisation is made through findings from Holder et al. (2024), who concluded that among the investigated initialisation methods, no significant differences were found between them.

4.2 Distance Functions

As mentioned, distance measures are needed to perform the update step in the k-means and k-medoids algorithms. Since the simple Euclidean distance is not the only way of calculating distances between time series, nine other functions will be discussed. The rest of the methodology of the distance functions resembles the methods and techniques used in Holder et al. (2024). It is important to note that specific algorithms that find the distances can be found in either Holder et al. (2024) or in the specific originating literature in Table 1. The next subsections elaborate more on each elastic distance.

4.2.1 Dynamic Time Warping and variations

The elastic measure that is most extensively used is Dynamic Time Warping (DTW). The idea is to warp two series to align with each other better by using a warping path.

Consider the sequences $\mathbf{a} = \{a_1, a_2, \ldots, a_n\}$ and $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}$. Let $N(\mathbf{a}, \mathbf{b})$ represent the $n \times n$ pointwise distance matrix between $\mathbf{a}$ and $\mathbf{b}$. A warping path through this matrix can be defined as:

$$P = \langle (e_1, f_1), (e_2, f_2), \ldots, (e_s, f_s) \rangle \tag{2}$$

where the path is constrained to avoid backtracking. The objective is to find the path P^* of length k that minimises the total distance:

$$D^*_P(\mathbf{a}, \mathbf{b}, N) = \sum_{i=1}^{k} N(e_i, f_i) = d_{dtw}(\mathbf{a}, \mathbf{b}) \tag{3}$$

While many potential paths exist, the primary focus is on the path P^* that defines the Dynamic Time Warping (DTW) distance. This optimal path can be determined using a dynamic programming approach, which can be made more efficient with techniques such as the Sakoe-Chiba band (Sakoe & Chiba, 1978).

The first modification is Derivative Dynamic Time Warping (DDTW). The idea is to first transform the series into a differential series. For a given series $\mathbf{a}$, the differential series is $\mathbf{a}' = \{a'_2, a'_3, \ldots, a'_{m-1}\}$. Here, a'_i is the average of the slopes between a_{i-1} and a_i, and a_i and a_{i+1}. This modification involves computing the DTW distance of the differenced series:

$$d_{ddtw}(\mathbf{a}, \mathbf{b}) = d_{dtw}(\mathbf{a}', \mathbf{b}'). \tag{4}$$

Next is Weighted Dynamic Time Warping (WDTW). WDTW introduces a weight penalty based on the warping distance between points in the warping path. When constructing the distance matrix N, a weight penalty $w(|i-j|)$ for a distance of $|i-j|$ is applied, such that:

$$N^{w}_{i,j} = w(|i-j|) \cdot (a_i - b_j)^2. \tag{5}$$

Various weighting functions can be used, but a logistic weight function is suggested in (Jeong et al., 2011). The problem is then analogous to other dynamic time warping problems, with a modified N matrix:

$$d_{\text{wdtw}}(\mathbf{a}, \mathbf{b}) = D^{*}_{P}(\mathbf{a}, \mathbf{b}, N_w). \tag{6}$$

WDTW also has a derivative version, known as Weighted Derivative Dynamic Time Warping (WDDTW), which is expressed as:

$$d_{\text{wddtw}}(\mathbf{a}, \mathbf{b}) = d_{\text{wdtw}}(\mathbf{a}', \mathbf{b}'). \tag{7}$$

4.2.2 Longest Common Subsequence

The next distance can be seen as different way of looking at DTW. Instead of aligning two time series by warping points onto each other to form a path, the Longest Common Subsequence (LCSS) distance creates a common series from the input sequences. As the name suggests, it finds the subsequence that is obtained through edit operations, such as deletion, with a certain cost. It does not provide a path from start to end, in the same way as DTW. Instead, LCSS describes the edit operations. The LCSS algorithm would have to identify matches between series and since this is a continous case, a distance treshold must be considered in the algorithm.

4.2.3 Edit distances

The previous distance LCSS was modified to introduce the Edit Distance on Real sequences (EDR), which also has a distance threshold to determine when two elements of a series match. Unlike LCSS, EDR assigns a constant penalty for elements that do not match, resulting in deletions (or gaps) in the alignment.

Edit distance with real penalty (ERP) was developed to address gaps in sequences in a different manner. While LCSS allows for gaps between elements, ERP assigns penalties to these gaps using a specific parameter. The primary difference from EDR is in how the penalty is determined. For ERP, the penalty is based on the distance to a predefined parameter rather than a fixed value. Furthermore, a cost matrix is employed that accounts for matching costs, incorporating a term for edit operations, such as inserting or deleting elements. ERP is therefore seen as an combination of earlier methods, as it includes the penalty mechanism of EDR but integrates it with a cost function linked directly to edit penalties. Additionally, ERP uses the squared distance for pointwise comparisons in the distance computation.

4.2.4 Move-Split-Merge

Move-Split-Merge (MSM) is in theory quite similar to ERP. Unlike ERP however, where the cost of an edit is based on the distance to a parameter, MSM focuses on the absolute difference between values.

As the name suggests, MSM has a split operation, which is where the main difference lies. Namely, the cost of an edit is defined differently to ERP. The cost of this operation depends on whether the value being inserted falls between the values being split. If it does, the cost is a fixed constant; if not, it includes an additional cost based on the deviation from the surrounding points. To summarize it, MSM uses a different cost system for insertions and deletions compared to ERP.

4.2.5 Time Warp Edit

Lastly, Time Warp Edit (TWE) is used in experimentation. TWE uses a parameter called stiffness, which controls the penalty applied to the distance between matched points. This makes TWE similar to weighted warping measures, where no penalty is applied when stiffness is zero. For deletion and insertion operations, TWE applies a constant stiffness penalty along with an additional edit penalty. This approach ensures that the warping is considered only for consecutive points within the same series.

Overall, TWE integrates the flexibility of warping with the specificity of editing into one distance measure.

4.3 Extension: Deep Neural Networks

The extension of this paper regards the use of unsupervised deep neural networks (UDNN) for TSCL. The paper of Lafabregue et al. (2022) delves deeper into this subject. As stated before, this study aims to compare the best performing UDNN clustering methods from this paper to the elastic distance functions based TSCL. To find the best performing model, Lafabregue et al. (2022) tests a large number of different UDNNs. Each UDNN proposed in Lafabregue et al. (2022) is constructed from three main components: architecture, parameter training method and lastly clustering training method. The authors conclude that a model with CNN architecture, reconstruction loss and no clustering loss performs best and this model will therefore be used for comparison. However, to validate the results, five other models are introduced with differing components. The remaining part of the section explains the components in more detail and the differences between the possibilities, along with the models that will be tested.

First of all, there is the architecture of a UDNN. This refers to the number of layers and the way the network is set up, including the hyperparameters in these layers. There are multiple architecture types discussed in Lafabregue et al. (2022), including CNN, recurring neural networks (RNN) and fully connected neural networks (FCNN). These architectures all consist of multiple layers which include an input layer, followed by hidden layers and lastly the output layer. Each architecture distinguishes itself by the way its hidden layers operate. Architectures such as CNN and FCNN are feedforward networks, since information between layers can only go in one direction, while archetypes such as RNN work with information that can be two-directional.

If the architectures are compared, CNN can deal with hierarchical data better than other neural network archetypes, since they start with relatively simple patterns in the first layers and then look at more intricate patterns and assemble patterns in the later layers. The convolution part of the network can be seen as applying and sliding a filter over the time series. Specifically 1D-convolutions are used. This means that each layer applies a certain number of filters of a certain kernel size to the input sequence. FCNN archetypes could be described as “vanilla” neural networks. As the name suggests, all neurons are fully connected. These models are particularly good at distinguishing non-linearly separable data patterns. Lastly, RNN architectures, which also includes the Long short-term memory (LSTM) network, are specifically useful when looking at time series data or other sequenced data, since existing patterns can be more easily detected. As mentioned in Section 2.4, RNN architectures are interestingly not always the best performing choice for TSCL, while they do excel at finding patterns in sequenced data.

The two other architectures that are examined other than CNN are RNN and the multilayered perceptron (MLP), where the latter is an example of an FCNN.

The second component is the training of the encoder’s parameters. Since this study uses unsupervised networks, meaning without labels, a different way of training must be examined. For UDNNs it is common to not use the raw data, but transform it in the latent space using an encoding function. Afterwards, patterns in this latent space are found and captured by a decoding function. This process is often referred to as a proxy task, since this is a side objective to train the UDNNs to learn a representation that will favor a good clustering. Autoencoders (AEs) can be used as one of the methods for unsupervised training. To train this AE, the mean squared error is minimised.

$$L_r = \frac{1}{n}\sum_{i=1}^{n}(x_i - g(f(x_i)))^2 \tag{8}$$

where f is the encoding function, g the decoding function and $[x_1, ..., x_n]$ the data. This is referred to as the reconstruction loss. Interestingly, this is one of the easiest ways to train the encoder’s parameters, while it performs the best in the results of Lafabregue et al. (2022). There exist multiple other methods to train the model explored in Lafabregue et al. (2022), including generating realistic data to train the encoder. This paper will only include the reconstruction loss in experiments.

The last ingredient is the possibility of specific training of parameters on clustering loss. Previous training methods aim at capturing relevant patterns in the data. However, this does not necessarily mean that a suitable clustering is prioritised. Therefore, Lafabregue et al. (2022) introduce an optional complementary clustering loss. These losses include popular methods such as deep embedded clustering (Xie, Girshick & Farhadi, 2016) and also losses specifically designed for time series, such as the Deep Temporal Clustering Representation (DTCR). This representation, introduced by Ma, Zheng, Li and Cottrell (2019) uses a three layer RNN encoder and a single RNN decoder. The training objective consists of three parts: a reconstruction loss, a real/fake loss and a k-means loss.

Despite there being various other clustering losses stated in Lafabregue et al. (2022), their results show that the existing and explained clustering losses do not contribute significantly to the performance of the clustering. To test this hypothesis, the DTCR is included as one of the options in experiments, alongside the option of no clustering loss.

So in the end, this leads to six models that will be tested: CNN-None, CNN-DTCR, MLP-None, MLP-DTCR, RNN-None and RNN-DTCR, where “None” refers to the absence of clustering loss. To reiterate, this selection is just a small part of the possibilities presented in

Lafabregue et al. (2022) and the idea is to validate the use of CNN-None.

The UDNN experiments work as follows in practice: The 61 datasets are selected from the total archive. Each model is trained through both the training and clustering phases if applicable. After the initial training phase, the trained model is used as a starting point for subsequent clustering methods. The models are trained on the training set and then tested on the test set, following the approach used by Ma et al. (2019). This means that the minimisation of the losses, both the reconstruction loss and the clustering loss, is performed on the training set, which leads to a model that performs the clustering on the actual test set. This ensures that the learned latent space can be effectively applied.

4.4 Clustering Metrics and Comparison

To compare the clustering experiments in an organised and viable way, the following three clustering metrics are used for each experiment. Note that these measures are also used in Holder et al. (2024).

The first and arguably the most important metric that is used in the evaluation is clustering accuracy (CL-ACC). As the name implies, clustering accuracy tests how accurate the clustering is of the experiment. Since the data has predefined labels, the accuracy can be calculated and summed up in the following equation:

$$\text{CL-ACC}(y, \hat{y}) = \max_{s \in S_k} \frac{1}{|y|} \sum_{i=1}^{|y|} \begin{cases} 1, & \text{if } y_i = s(\hat{y}_i) \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

This equation validates if a cluster prediction $\hat{y}$ is correct in the sense that it matches with the best matching class value S_k. It is a maximization and to avoid calculating every permutation to find the maximum, an optimization algorithm is implemented, which includes constructing a cost matrix and assigning clusters using the Hungarian Algorithm (Kuhn, 1955) to validate the correctness of the cluster assignment. The indicator function in the equation is true when the cluster value is equal to the class value it is associated and zero otherwise.

The Rand index (RI) (Rand, 1971) looks at similarity between two sets of labels. The following equation encapsulates the idea behind the Rand Index:

$$RI = \frac{\text{number of pairs in both sets}}{\text{total pairs}} \tag{10}$$

This essentially boils down to a similarity measure between the cluster labels and the predefined labels. A larger RI index indicates a better clustering. Since this method looks at the partition of "correct" pairs, the value lies between 0 and 1. The clustering metric that will be used is an altered form of the RI, which compensates for randomness, namely the adjusted rand index (ARI). Instead of taking the actual values of the pairs of the labels, a contingency matrix is set up, where pairwise comparisons are made between clusterings made by a random model. Note that the value of the ARI can now also be negative, since the expected index can be larger than the actual index (Wagner & Wagner, 2007). ARI eliminates any randomness that could be present when comparing the labels. This is especially useful when comparing datasets with a large number of clusters, since random equalities can occur more often. However, this research focuses on datasets with a smaller number of clusters, so one could argue that the ARI measure is not necessary in this research and RI should be used. This is not the case, seeing that the deep neural network extension of Lafabregue et al. (2022) also uses ARI in their analysis. In order to conduct a comparison, ARI is chosen. A last thing to note is that RI and ARI do not have permutation problem that needs be accounted for, since a direct comparison is made between the sets.

Lastly, Normalised Mutual Information (NMI) is also a measure used to evaluate the similarity between two clustering results by quantifying the shared information between two sets. Mathematically, NMI between two clusters U and V is defined as:

$$\text{NMI}(U,V) = \frac{2 \cdot I(U;V)}{H(U) + H(V)} \tag{11}$$

where $I(U;V)$ is the mutual information between clusters U and V, and $H(U)$ and $H(V)$ are the entropies of U and V, respectively. Mutual information $I(U;V)$ measures the amount of information obtained about one cluster by knowing the other by using the marginal and joint distributions of data items in U and V, while entropy quantifies the uncertainty or disorder within each cluster. A higher value of the NMI indicates a better alignment between results. There are different methods of defining NMI as Vinh, Epps and Bailey (2009) point out. Nevertheless the key part is that the normalisation results in a stable range $[0,1]$ within which NMI values can fall. Furthermore, Lafabregue et al. (2022) also use NMI when comparing clustering results.

Comparing these metrics within experiments is the last step in obtaining the results. These metrics can be compared between the methods using pairwise Wilcoxon signed-rank tests and cliques resulting from a Holm correction (Benavoli, Corani & Mangili, 2016). For all tests that will be performed, $\alpha = 0.05$ will be used.

4.5 Practical Implementation and tuning

The authors Holder et al. (2024) have provided example code and a guide on how to replicate the research. As stated before, due to long runtime, some of the datasets will not be used. In Lafabregue et al. (2022), code is also provided for replicating the neural networks.

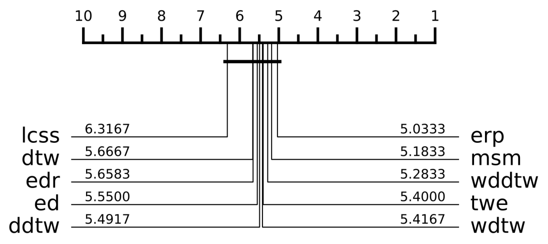
All experiments are run on an AMD Ryzen 7 4700U, 2000 MHz in Visual Studio Code with Python.

5 Results

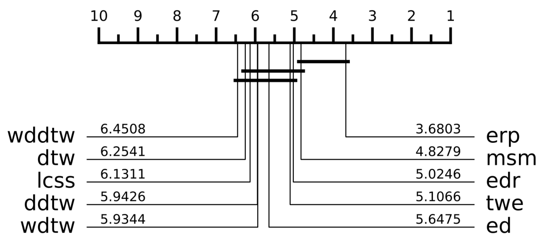
The results section is structured as follows. Firstly, the elastic distance measures will be compared using the classification statistics mentioned in the methodology. This also includes a comparison of the k-means and k-medoids. Lastly, the neural network alternatives will be compared within this category and thereafter with the best performing clustering methods with elastic distance functions, which will indicate which of the methods is preferred.

5.1 Elastic Distances and algorithms

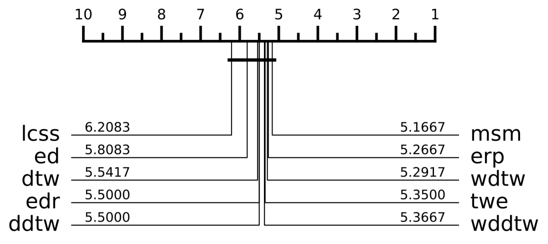
The evaluation of the elastic distances can be seen in Figure 2, where the k-means and k-medoids results are evaluated and critical differences are shown.



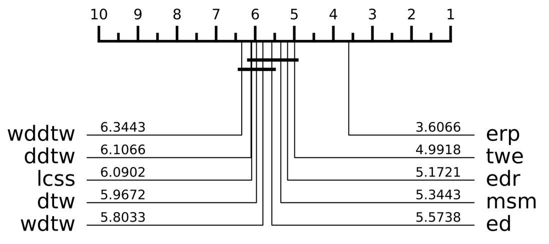
(a) Accuracy k-means



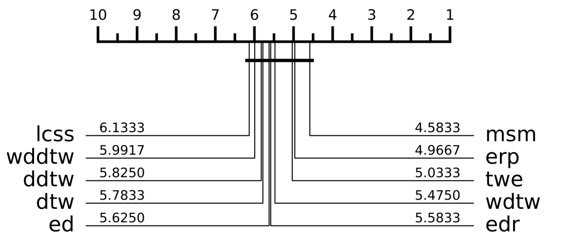
(b) Accuracy k-medoids



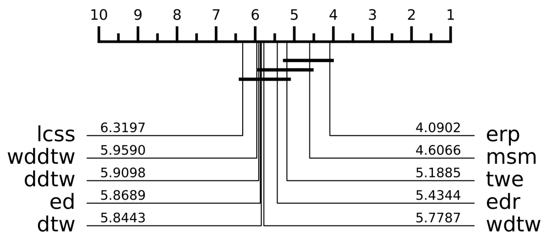
(c) ARI k-means



(d) ARI k-medoids



(e) NMI k-means



(f) NMI k-medoids

Figure 2: Critical difference diagrams for k-means and k-medoids clustering separately on 61 UCR datasets test data for three performance measures. Algorithms are listed on the sides with their corresponding average ranks, where a lower rank is better. The horizontal lines connect algorithms whose performance differences are not statistically significant. If two algorithms are not connected by a line, their performance difference is statistically significant.

These critical differences show that k-means and k-medoids clusterers perform relatively similar. Measures such as ERP, MSM, EDR and TWE perform perform generally better, although the difference between these measures seems smaller with k-means compared to k-medoids, where the latter shows a clearly better results for ERP. In Holder et al. (2024) MSM is the best performing measure, while it seems to be overshadowed here by the ERP distance. The worst performing experiments also coincide with the results of Holder et al. (2024), which include LCSS, ED and variants of DTW. They are on the other hand not significantly worse than all off the other measures.

A comparison of the runtimes is also given in Table 2.

Distance	k-Means	k-Medoids
MSM	55.73	43.53
TWE	**1329.88**	**293.31**
ERP	492.49	95.16
WDTW	44.57	54.61
DTW	138.02	53.33
ED	0.07	21.94
DDTW	395.44	96.77
WDDTW	123.44	97.42
LCSS	389.79	98.18
EDR	381.92	91.60

Table 2: Average runtime of experiment on 1 dataset in seconds of k-means and k-medoids over 61 datasets. In bold are the algorithms that took the longest.

These results implicate that TWE is the the algorithm that has the longest runtime. This means that although TWE is a well-performing measure, the long runtime could be something to take into account when implementing in practice. Notably also, the k-medoids experiments have a considerably lower runtime. This can be understood by referring to the explanation of how k-medoids works, which is stated in Section 4.1. For k-medoids only the distances between observations are needed, compared to new average calculations in k-means. This severely lowers the runtime.

It is also important to consider the datasets Holder et al. (2024) use. Since this paper takes roughly half of the datasets that are used in Holder et al. (2024), the results can also be compared to these datasets specifically. Comparison of the accuracy can be seen in Table 3 for the same 61 datasets, while in Table 4 the accuracy of the full sample of Holder et al. (2024) is used. These results confirm that k-medoids algorithms have a higher accuracy overall, compared to k-means, as was also stated in Holder et al. (2024). It is interesting however to compare Table 3 and 4, as can be seen that the accuracy is exceptionally different between the "full" sample and the results that are found in this paper. These results indicate that datasets with a larger number of clusters and bigger size could lead to a lower average accuracy. This would match with the complexity of larger datasets and number of clusters. Especially the second criteria could lead to these results, since correctly clustering a time series to one of two clusters seems intuitively easier than to correctly cluster it when ten or more clusters are possible.

Then there exists a discrepancy in the results in Table 3. The results are similar, but not exactly equal. For the k-medoids algorithm, ERP is in both the best performing measure in accuracy. However, WDDTW performs the best in this paper's results, while MSM performs better in the results of Holder et al. (2024). This discrepancy might be explained by method of implementation. Newer versions of packages such as Tensorflow, aeon and scikit are used in experimentation, as well as a newer version of python.

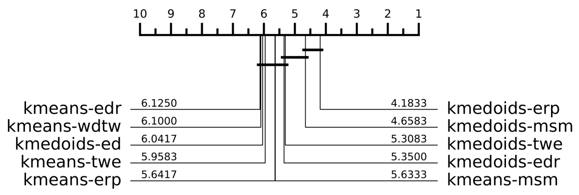
	Accuracy results 61 datasets			Accuracy results 61 datasets in Holder et al. (2024)		
Distance	k-Means (%)	k-Medoids (%)	Difference (%)	k-Means (%)	k-Medoids (%)	Average difference
MSM	60.87	63.20	2.33	**63.01**	64.67	1.81
TWE	60.30	62.19	1.89	61.95	63.64	1.55
ERP	61.00	**66.07**	5.08	64.14	**66.03**	1.55
WDTW	60.44	61.72	1.28	62.07	62.66	1.29
DTW	60.05	61.58	1.09	61.49	62.82	1.34
ED	59.18	60.00	0.82	60.86	61.31	1.50
DDTW	60.60	61.18	0.58	55.01	59.90	-3.44
WDDTW	**61.43**	58.37	-3.06	57.21	57.88	-2.34
LCSS	58.16	60.30	2.15	57.97	60.35	-0.07
EDR	59.65	62.41	2.76	59.61	59.67	-1.39

Table 3: Accuracy averaged over 61 problems for k-means and k-medoids. The average difference concerns the difference between the results of Holder et al. (2024) and results found in this paper of the same 61 datasets. The best performing algorithm is in bold

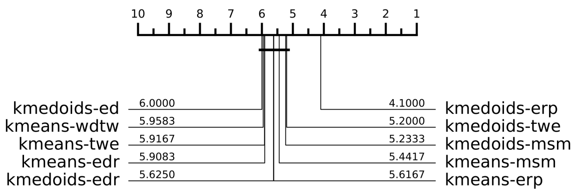
Distance	k-Means (%)	k-Medoids (%)	Average difference
MSM	**54.16**	**55.69**	7.11
TWE	52.85	55.63	7.00
ERP	50.89	54.83	10.68
WDTW	52.25	53.58	8.17
DTW	49.08	52.96	9.80
ED	51.78	51.50	8.00
DDTW	42.57	50.22	14.50
WDDTW	46.99	49.55	11.63
LCSS	45.76	49.88	11.41
EDR	45.20	49.70	13.58

Table 4: Accuracy results averaged over 112 problems from Holder et al. (2024). The average difference concerns the difference between these results of Holder et al. (2024) and results found in this paper of 61 datasets. The best performing algorithm is in bold

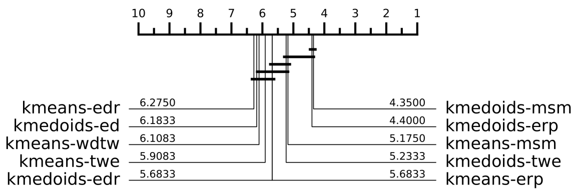
Next a comparison and critical difference diagram of the best performing models is given in Figure 3. It can be seen that the k-medoids methods perform generally better on all performance metrics. This concurs with the results in Table 3.



(a) Accuracy



(b) ARI

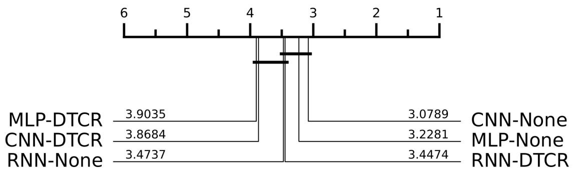


(c) NMI

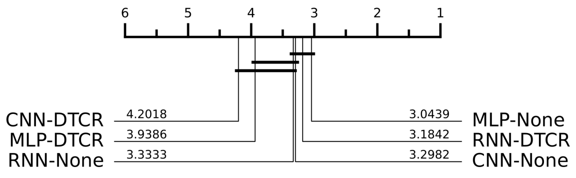
Figure 3: Critical difference diagrams for best performing distance measure analysis

5.2 Neural Network

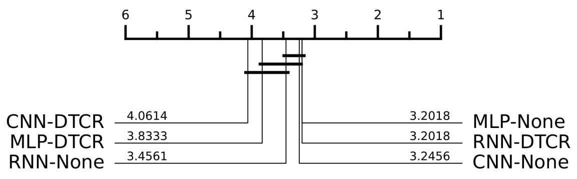
Firstly, a comparison is made between some of the possibilities within the neural network framework that Lafabregue et al. (2022) set up. This includes the architectures mentioned in the Methodology and the critical difference results, which can be seen in Figure 4.



(a) Accuracy



(b) ARI



(c) NMI

Figure 4: Critical difference diagrams for chosen neural network based algorithms on the clustering metrics

From Figure 4 it could be concluded that models without clustering loss training are preferred, since CNN and MLP architectures without clustering loss perform better than the counterparts with DTCR. The scatterplots of the results, along with the p-values of the Wilcoxon test are in Figures 5 and 6. It is noted that a small subset of clustering losses is experimented with on a small set of network architectures, so this does not dismiss the use in clustering loss at all. Also the RNN architecture performs better with the DTCR clustering loss. Ultimately, the models CNN and MLP without clustering loss perform the best of the UDNN models that are examined. However they are not significantly better on all performance measures. Furthermore, it can also not be said that CNN-None is the best model based on these results. However, this is the model that will be used in further comparison, since this is the model Lafabregue et al. (2022) stated as the best and it is still one of the better performing models in this experiment.

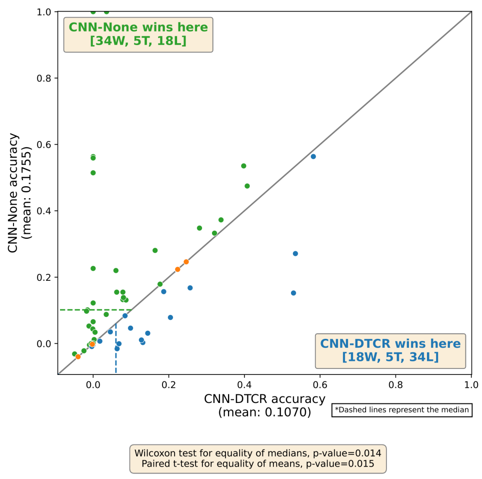


Figure 5: ARI scatterplot CNN with and without clustering loss

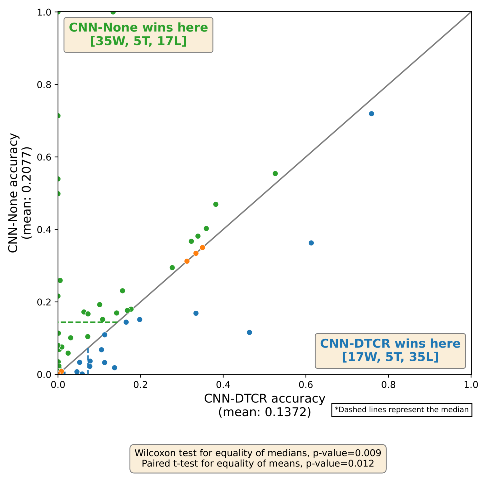
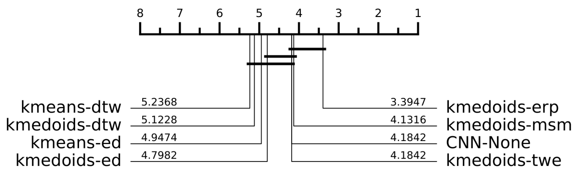
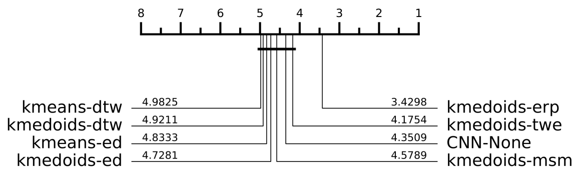


Figure 6: NMI scatterplot CNN with and without clustering loss

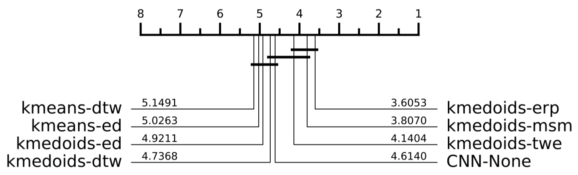
Now that the CNN model is confirmed to be one of the better models for TSCL, it can be compared to interesting distance based clustering results. Figure 7 picks some of the best performing models, which are also in Figure 3, the CNN model and DTW and ED measures. The last two are chosen, since this paper aims to validate the claim of Lafabregue et al. (2022), who conclude that CNN based models outperform DTW and ED based time series clustering.



(a) Accuracy



(b) ARI



(c) NMI

Figure 7: Critical difference diagrams for interesting combinations of experiments

It is apparent that CNN is not the best performing algorithm, also when looking at Figure 8, which shows the boxplot of the accuracy for these measures. The CNN algorithm looks quite different from all other measures and is more evenly spread towards its mean, while also obtaining some very low scores. It is important to note that the CNN algorithm does beat the DTW and ED distance measures, even if it is only marginally.

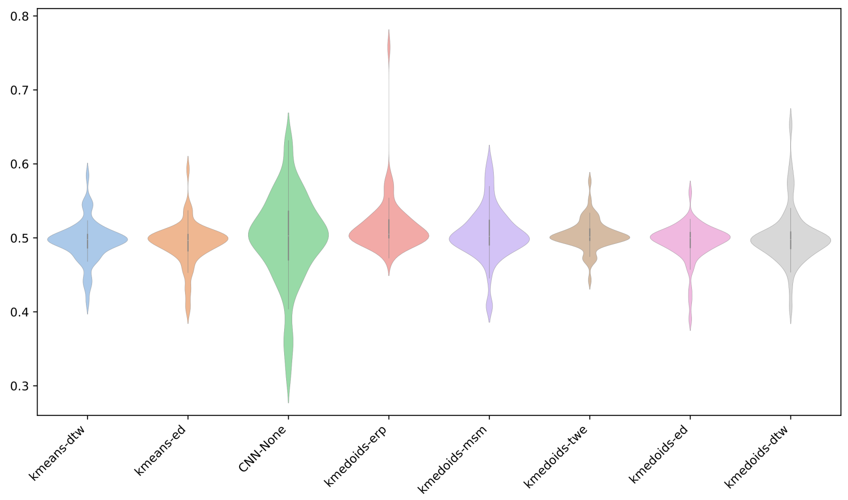


Figure 8: Boxplot of clustering accuracy (y-axis) of 8 different algorithms

This can also be seen in Table 5, where the average performance of the best performing algorithms for the three performance measures is stated. Interestingly, on average CNN performs similar to k-means ERP, but it can also be viewed that the other methods outperform CNN.

	CNN	KMeans-erp	Kmeans-msm	Kmedoids-erp	Kmedoids-msm
Accuracy	0.61	0.61	0.61	0.66	0.63
ARI	0.18	0.18	0.19	0.25	0.22
NMI	0.21	0.21	0.23	0.27	0.26

Table 5: Average performance measures for the best performing algorithms over 61 datasets

The question remains why a deep neural network could not outperform all elastic distance based approaches. This could have multiple explanations. Firstly, there is the nature of the unsupervised context. CNNs typically excel in supervised learning context and even with an autoencoder set-up, the lack of a specific model trained for TSCL could have an impact on the results. Secondly, there is the set-up of this experiment, specifically the datasets that are chosen. These include the smallest datasets from the UCR archive with the smallest number of training observations, compared to the larger datasets which also exist in the archive. The amount of training is crucial for a neural network. The lack of extensive training possibilities through smaller datasets could be a reason why CNN does not perform better.

On the other hand, methods such as MSM and ERP are specifically designed for time series data and excel at capturing temporal relationships and distortions. These methods do not require training and are less likely to fail due to the data's specific characteristics.

6 Conclusions

This study explored two primary approaches: elastic distance-based methods and unsupervised deep neural networks, aiming to optimise clustering performance. Specifically, k-means and k-medoids with nine elastic distances and a convolutional neural network were tested across 61 datasets from the UCR archive.

First, in addressing whether the results from the replication match those of Holder et al. (2024), the findings confirmed several key insights from their study. Notably, k-medoids models exhibited better performance than k-means and MSM and ERP measures consistently outperformed DTW and other examined measures. By taking a subset of the full database Holder et al. (2024) use, it is interesting to see that the ERP measure performs better in general on smaller datasets. The results are not exactly the same however, which might be explained by version of software or other randomisation.

Secondly, all results are compared through three performance measures: clustering accuracy, adjusted Rand index, and normalised mutual information. This selection is also used in Holder et al. (2024) and is also obtained for the neural networks.

The following sub-question regards the identification of the "best" deep neural network for time series clustering. The CNN-None model proves to be a fitting model compared to other options. On the other hand, it cannot be said that it is significantly better than the MLP-None and RNN-DTCR models.

The computation time comparisons revealed that the distance measure TWE has the longest runtime. Another result showed that k-means has significantly longer runtimes than the k-medoids algorithm.

The comparisons between the neural network and the best-performing elastic distance-based algorithms showed that the CNN model cannot outperform the MSM and ERP distance measures. While it improves upon measures such as DTW and ED, it does not conclusively demonstrate superior performance.

In summary, while UDNN models for TSCL are promising, especially with larger and more diverse datasets, traditional elastic distance measures like MSM and ERP remain highly effective and reliable. For practical applications where robustness is critical, distance-based methods should continue to be favoured. Future research could explore hybrid approaches such as implementing k-medoids instead of k-means initialisation in the neural networks. In the literature, there exists no combination of these two branches of TSCL so this could be interesting to combine in a hybrid solution if possible. Furthermore, one could delve deeper into other medoids type algorithms, since this algorithm performed best in this study. This could involve partitioning around medoids, described in Leonard Kaufman (1990). As it stands now in this paper however, unsupervised neural networks cannot outperform distance based algorithms in the field of time series clustering.

References

Abanda, A., Mori, U. & Lozano, J. A. (2019). A review on distance based time series classification. *Data Mining and Knowledge Discovery*, *33*(2), 378–412.

Al Hasan, M., Chaoji, V., Salem, S. & Zaki, M. J. (2009). Robust partitional clustering by outlier and density insensitive seeding. *Pattern Recognition Letters*, *30*(11), 994–1002.

Alqahtani, A., Ali, M., Xie, X. & Jones, M. W. (2021). Deep time-series clustering: A review. *Electronics*, *10*(23), 3001.

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*, 606–660.

Bandara, K., Bergmeir, C. & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert systems with applications*, *140*, 112896.

Benavoli, A., Corani, G. & Mangili, F. (2016). Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, *17*(1), 152–161.

Berndt, D. J. & Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining* (pp. 359–370).

Chen, L. & Ng, R. (2004). On the marriage of lp-norms and edit distance. In *Proceedings of the thirtieth international conference on very large data bases-volume 30* (pp. 792–803).

Chen, L., Özsu, M. T. & Oria, V. (2005). Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 acm sigmod international conference on management of data* (pp. 491–502).

Dau, H. A., Keogh, E., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., ... Hexagon-ML (2018, October). *The ucr time series classification archive.* (`https://www.cs.ucr.edu/ eamonn/time`$_series_data_2$`018/`)

Ester, M., Kriegel, H.-P., Sander, J., Xu, X. et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, pp. 226–231).

Górecki, T. & Łuczak, M. (2013). Using derivatives in time series classification. *Data Mining and Knowledge Discovery*, *26*, 310–331.

Holder, C., Middlehurst, M. & Bagnall, A. (2024). A review and evaluation of elastic distance functions for time series clustering. *Knowledge and Information Systems*, *66*(2), 765–809.

Hu, B., Chen, Y. & Keogh, E. (2016). Classification of streaming time series under more realistic assumptions. *Data mining and knowledge discovery*, *30*(2), 403–437.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L. & Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data mining and knowledge discovery*, *33*(4), 917–963.

Jeong, Y.-S., Jeong, M. K. & Omitaomu, O. A. (2011). Weighted dynamic time warping for time series classification. *Pattern recognition*, *44*(9), 2231–2240.

Keogh, E. & Kasetty, S. (2002). On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proceedings of the eighth acm sigkdd international conference on knowledge discovery and data mining* (pp. 102–111).

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, *2*(1-2), 83–97.

Lafabregue, B., Weber, J., Gançarski, P. & Forestier, G. (2022). End-to-end deep representation learning for time series clustering: a comparative study. *Data mining and knowledge discovery*, *36*(1), 29–81.

Leonard Kaufman, P. (1990). *Partitioning around medoids (program pam), chapter 2.* Wiley.

Linde, Y., Buzo, A. & Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on communications*, *28*(1), 84–95.

Ma, Q., Zheng, J., Li, S. & Cottrell, G. W. (2019). Learning representations for time series clustering. *Advances in neural information processing systems*, *32*.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297).

Marteau, P.-F. (2008). Time warp edit distance with stiffness adjustment for time series matching. *IEEE transactions on pattern analysis and machine intelligence*, *31*(2), 306–318.

Paterson, M. & Dančík, V. (1994). Longest common subsequences. In *International symposium on mathematical foundations of computer science* (pp. 127–142).

Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., ... Keogh, E. (2013). Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *7*(3), 1–31.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, *66*(336), 846–850.

Räsänen, T. & Kolehmainen, M. (2009). Feature-based clustering for electricity use time series data. In *Adaptive and natural computing algorithms: 9th international conference, icannga 2009, kuopio, finland, april 23-25, 2009, revised selected papers 9* (pp. 401–412).

Sakoe, H. & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, *26*(1), 43–49.

Stefan, A., Athitsos, V. & Das, G. (2012). The move-split-merge metric for time series. *IEEE transactions on Knowledge and Data Engineering*, *25*(6), 1425–1438.

Tavakoli, N., Siami-Namini, S., Adl Khanghah, M., Mirza Soltani, F. & Siami Namin, A. (2020). An autoencoder-based deep learning approach for clustering time series data. *SN Applied Sciences*, *2*, 1–25.

Vinh, N. X., Epps, J. & Bailey, J. (2009). Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual international conference on machine learning* (pp. 1073–1080).

Wagner, S. & Wagner, D. (2007). Comparing clusterings: an overview.

Wang, Z., Yan, W. & Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *2017 international joint conference on neural networks (ijcnn)* (pp. 1578–1585).

Xie, J., Girshick, R. & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis.

In *International conference on machine learning* (pp. 478–487).

Zhang, T., Ramakrishnan, R. & Livny, M. (1996). Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, *25*(2), 103–114.

Zhao, B., Lu, H., Chen, S., Liu, J. & Wu, D. (2017). Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, *28*(1), 162–169.

Appendix

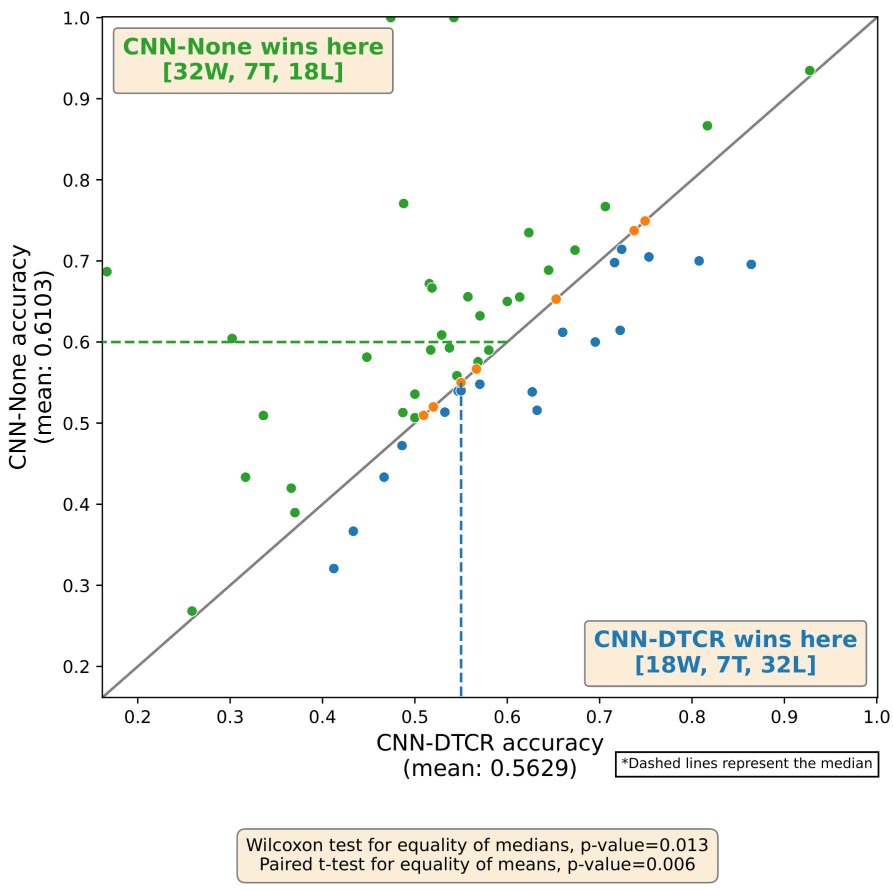


Figure 9: Scatterplot of accuracy CNN with and without clustering loss

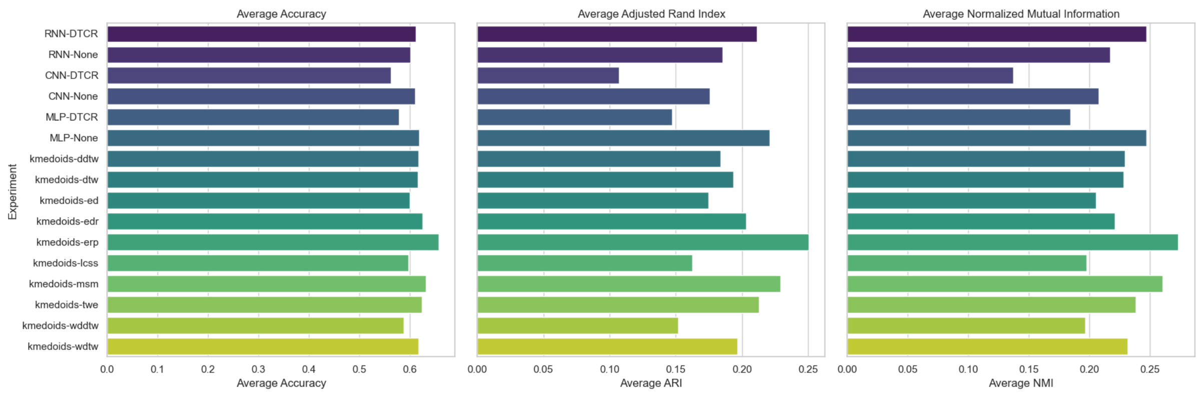


Figure 10: Visualisation of performance measures on neural network verus *k*-medoids

Parameter	**Value**
Batch Size	10
Number of Filters	40
Compared Length	null
Depth	10
Number of Steps	10
Number of Steps for Pretraining	10
Kernel Size	3
Penalty	null
Learning Rate	0.001
Number of Random Samples	1
Negative Penalty	1
Latent Dimension	320
Reduced Size	160

Table 6: List of default hyperparameters used in deep learning experiments