Estimating Time Series Averages from Latent Space of Multi-tasking Neural Networks

Tsegamlak Terefe[1,2] · Maxime Devanne[2] · Jonathan Weber[2] · Dereje Hailemariam [1] · Germain Forestier[2,3]

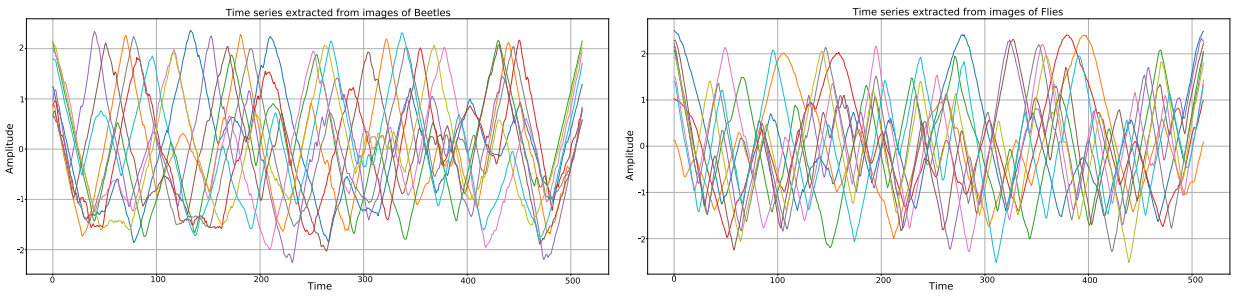
Abstract Time series averages are one key input to temporal data mining techniques such as classification, clustering, forecasting, etc. In practice, the optimality of estimated averages often impacts the performance of such temporal data mining techniques. Practically, an estimated average is presumed to be optimal if it minimizes the discrepancy between itself and members of an averaged set while preserving descriptive shapes. However, estimating an average under such constraints is often not trivial due to temporal shifts. To this end, all pioneering averaging techniques propose to align averaged series before estimating an average. Practically, the alignment gets performed to transform the averaged series, such that, after the transformation, they get registered to their arithmetic mean. However, in practice, most proposed alignment techniques often introduce additional challenges. For instance, Dynamic Time Warping (DTW) based alignment techniques make the average estimation process non-smooth, non-convex, and computationally demanding. With such observation in mind, we approach time series averaging as a generative problem. Thus, we propose to mimic the effects of temporal alignment in the latent space of multi-tasking neural networks. We also propose to estimate (augment) time domain averages from the latent space representations. With this approach, we provide state-of-the-art latent space registration. Moreover, we provide time domain estimations that are better than the estimates generated by some pioneering averaging techniques.

Keywords Latent Space · Multi-tasking · Time Series Averaging

1 Introduction

In today's data-driven world, time series are one of the dominant datasets that get intensively investigated. The possibility of defining the datasets from seemingly unrelated topics makes them even more interesting for further investigation (Lines 2015, Chen et al. 2015). For instance, in (Lines 2015), time series were extracted from segmented images of chickens, by recording the movement of earthworms, by taking the record of the power consumption of home appliances, from food spectrograph, etc. In reality, the temporal datasets covered broad application domains that extended from computer vision to behavioral genetics.

[1]School of Electrical and Computer Engineering, Addis Ababa Institute of Technology, Addis Ababa University, Addis Ababa, Ethiopia
[2]IRIMAS, Université de Haute-Alsace, Mulhouse, France
[3]DSAI, Monash University, Melbourne, Australia

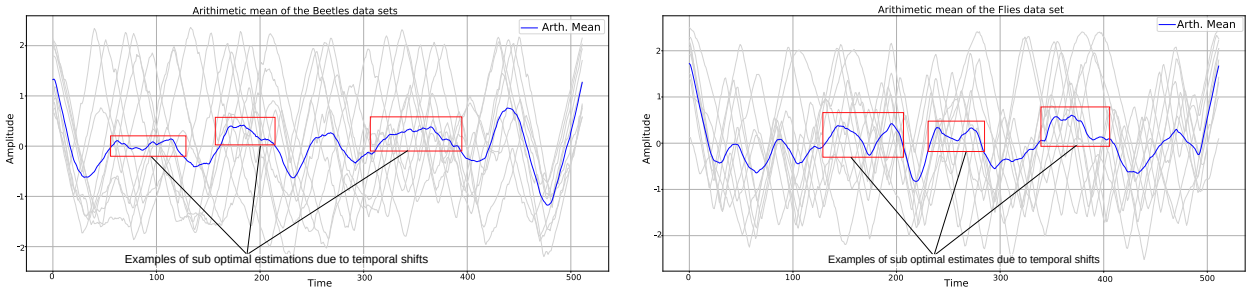


(a) Time series representing Beetles (b) Time series representing Flies

Fig. 1: Time series extracted from the images of Beetles and Flies

Mathematically, a time series gets defined by taking a set of ordered observations where the ordering can be in space, time, frequency, etc (Wei 2006). Moreover, each observation could be real numbers, symbols, boolean values, etc. However, in our context, we assume a univariate time series $X = \{x_1, x_2, x_3, ..., x_N\} \in \mathbb{R}^N$ is defined by taking individual observations ($x_i \in X$) from $\mathbb{R}$. However, if a time series is multivariate, each observation gets taken from $\mathbb{R}^K$. Overall, for both cases, the ordering of the individual observations get expected to form unique descriptive shapes (features) that are utilized by most supervised, unsupervised, and semi-supervised temporal data mining techniques (Bagnall et al. 2012, Lin & Li 2009, Lin et al. 2007, Fawaz et al. 2019). For instance, in time series classification, algorithms or optimization setups mainly aim to identify (learn) similar class-specific (per-class) features to identify class memberships (Bagnall et al. 2012, Fawaz et al. 2019). However, in practice, temporal data mining techniques do not always rely on features evident in individual time series. On the contrary, they sometimes could rely on series that summarize shapes or patterns observed in groups of temporal datasets. For example, in (Shawel et al. 2020), time series got defined by taking hourly traffic demands from 729 Base Transceiver Stations (BTS). The series later got grouped into clusters representing traffic patterns observed within different geographical areas. In this practical case, cluster centroids (averages) determined how the temporal datasets (base stations) get grouped. Moreover, the centroids determined the parameters of the forecasting models designed to capture future traffic demands of base stations belonging to a given cluster. Practically, the need for time series which summarize a group of temporal datasets is not limited to time series forecasting. On the contrary, we can also find such demands in time series clustering, classification, signal processing, etc (Aghabozorgi et al. 2015, Bagnall et al. 2012, Gupta et al. 1996). Overall, when such needs arise, a group of temporal datasets is often summarized by taking averages (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Petitjean & Gançarski 2012, Shapira Weber et al. 2019*a*, Bock 2008, Paparrizos & Gravano 2015, Schultz & Jain 2018).

Even though summarizing temporal datasets through averaging proved useful, in reality, such summarization is not trivial mainly due to temporal distortions (shifts along the time axis). In practice, temporal distortions could be evident for various reasons (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Petitjean & Gançarski 2012, Schultz & Jain 2018, Shapira Weber et al. 2019*a*, Terefe et al. 2020). To further elaborate on this, we can consider a group of time series representing Beetles and Flies as an example. In Figure 1, we have plotted samples from both categories taken from the University of California Univariate Time Series Repository (UCR) (Chen et al. 2015). As evident from the plots, the series representing Beetles or Flies are not well aligned, i.e., they are affected by temporal distortion. In this case, the temporal distortions originated from



(a) Beetles dataset arithmetic mean.

(b) Flies dataset arithmetic mean.

Fig. 2: Arithmetic mean of the UCR Beetles (a) and Flies (b) datasets.

the way the datasets got defined. In this regard, initially, colored images of Beetles and Flies were taken. Following this, the images got segmented into binary (black and white) images using pixel value thresholding. The segmentation introduced a demarcation (contour) between an image of a Beetle (Fly) and its surrounding background. With such contours at hand, it was possible to take euclidean distances between samples of contour points and a reference point within the boundaries of the image of Beetle (Fly). Consequently, the distance measurements defined the amplitude values of the temporal data sets, i.e., $x_i \in X$. Moreover, since the distance measurements were taken by taking samples at fixed angular steps, the sequence over which samples got taken defined the ordering of the amplitude values (Lines 2015). To this end, the time series extracted from the images of a Beetle (Fly) would get shifted along the time axis if an image gets rotated by a certain angle. Additionally, imperfections in images and small variations in the shapes and sizes of Beetles (Flies) would also contribute to the temporal distortions (Lines 2015). Overall, practically, we often find time series getting affected by time shifts for various reasons. For instance, differences in sampling rates, behaviors of observed entities, shapes, sizes, etc (Ye & Keogh 2009, Lines 2015, Shawel et al. 2020). In reality, such sources of temporal shifts often become a challenge for most temporal data mining techniques (Aghabozorgi et al. 2015, Xie et al. 2016, Bagnall et al. 2012, Fawaz et al. 2019). In this aspect, temporal distortions often make arithmetic mean, i.e., the most trivial average, to be a sub-optimal estimate in the context of preserving shapes (patterns) observed within an averaged set (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Petitjean & Gançarski 2012, Schultz & Jain 2018, Shapira Weber et al. 2019*a*). This is better demonstrated in Figure 2 where we have plotted the arithmetic means of the series shown in Figures 1a and 1b. In Figures 2a and 2b, we have marked some segments of the arithmetic means using red boxes in order to emphasize on segments with significant shape distortions. In these segments, temporal distortion misaligned peaks and troughs that give estimations close to zero. This simple example demonstrates how a temporal data mining technique relying on arithmetic mean could easily misinterpret underlying data due to temporal distortion.

In reality, temporal distortion affects the performances of a range of temporal data mining techniques (Bagnall et al. 2012, Lin & Li 2009, Lin et al. 2007, Aghabozorgi et al. 2015). To this end, in practice, different temporal data mining techniques often propose different heuristics to overcome the challenge. For instance, in distance-based time series classification and clustering tasks, earlier proposals suggest either domain transformation (Lin et al. 2007, Lin & Li 2009) or the utilization of elastic distance functions (Bagnall et al. 2012, Petitjean et al. 2011, Bagnall & Lines 2014). On the contrary, recent proposals suggest boosting classification accuracy by using neural networks that extract dominant descriptive features through complex nonlinear

transformations (Fawaz et al. 2019, Junyuan et al. 2016, Lafabregue et al. 2021*a*). In this aspect, pioneering and recent time series averaging heuristics often propose to transform members of an averaged set before estimating an average. Consequently, in most cases, they consider temporal datasets as vectors in $\mathbb{R}^N$ that gets transformed to vectors in $\mathbb{R}^\tau$, where $\tau \geq N$ (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Petitjean & Gançarski 2012, Schultz & Jain 2018, Paparrizos & Gravano 2015, Shapira Weber et al. 2019*a*). In practice, the transformation often gets performed by aligning members of an averaged set either in a pairwise manner (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009) or as compared to a template (landmark) that is iteratively updated (Petitjean et al. 2011, Shapira Weber et al. 2019*a*, Paparrizos & Gravano 2015). In practice, if the latter approach gets utilized, the template is often initialized either randomly or by using the values of one of the averaged series (Petitjean et al. 2011, Paparrizos & Gravano 2015). Moreover, as the averaging process progresses, the template often gets updated by taking the arithmetic mean of the aligned series. In other words, we can see such averaging techniques as performing registration of averaged series to their $\mathbb{R}^\tau$ space arithmetic mean. Generally, we can broadly group pioneering averaging techniques into two broad categories: i.e., those relying on elastic and non-elastic alignment functions (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Petitjean et al. 2011, Paparrizos & Gravano 2015) and those relying on different transformation techniques found in functional data analysis (Shapira Weber et al. 2019*a*, Chen & Srivastava 2021, Kowsar et al. 2022). Overall, to the best of our knowledge, most proposed averaging heuristics approach time series averaging as a registration (alignment) problem. To this end, the quality and the characteristics of the estimated means often get dependent on the type of underlying alignment technique. For instance, in both broad categories, the quality of the estimated means is often guaranteed in the transformed (registered) space. To this end, pioneering techniques often make assumptions about how the estimates get utilized. For instance, most distance-based classification and clustering tasks propose to utilize Dynamic Time Warping (DTW) distance while using averages estimated in DTW space (Bagnall et al. 2012, Aghabozorgi et al. 2015, Fawaz et al. 2019, Bagnall & Lines 2014, Petitjean et al. 2011). In reality, DTW distance transforms the estimated averages to their registered space where their optimality is relatively guaranteed. Similarly, in velocity field-based averaging heuristics, the heuristics rely on the capability of an alignment network to transform any unseen data into a space where the averages get estimated (Shapira Weber et al. 2019*a*). However, in some practical cases, such underlying assumptions might not always be feasible. For instance, in the case of (Shawel et al. 2020, Debella et al. 2022), we cannot directly utilize DTW distance or alignment networks while fitting forecasting models.

With these observations in mind, in this paper, we deviate from the standard approach and see time series averaging as an augmentation (generative) problem. This way, we further weaken the link between the average estimation process and how the estimates get utilized in practical situations. To meet this objective, we investigate the possibility of using latent features of neural networks for time series averaging. In our investigations, we emphasize two factors that could significantly impact the type and quality of latent space representations, i.e., objective functions and network architecture. In reality, by investigating these key parameters, we try to address two key questions. First, we ask ourselves, can we mimic multiple alignment (registration) through a carefully guided latent space embedding? If we can do so, we then ask, how can we estimate time domain averages from such latent space embedding? In general, by providing logical and sound answers to these two key questions, we make the following two contributions:

– We provide a systematic investigation of the possibility of utilizing the latent space of neural networks for time series averaging. Moreover, we provide a latent space registration that is

statistically better than the registrations obtained with all existing averaging heuristics.

– We provide a time domain estimate that is statistically better than a time domain arithmetic mean. Moreover, we also show that our best-performing proposal could outperform estimations obtained with some DTW-based averaging heuristics.

With this said, we organize the rest of this paper into five sections. In section 2, we present a brief review of previous proposals and concepts fundamental to time series averaging. In section 3, we present our proposals and the underlying argument behind them. We then give the experimental setup we used to evaluate our proposal in section 4. Finally, we present our findings and conclusive remarks in section 5 and 6.

2 Background and Previous Works

In practice, most pioneering time series averaging techniques highly rely on DTW to perform temporal alignment (Schultz & Jain 2018, Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Petitjean & Gançarski 2012). Consequently, all DTW based averaging technique's incorporate DTW distance to their objective function, i.e., the Fréchet function or (1) (Petitjean & Gançarski 2012, Schultz & Jain 2018):

$$F(\mu) = \min(\sum_{i=1}^{K} \delta_{p_i}(X_i, \mu)), \tag{1}$$

where, $K, \{X_i \in \mathbb{R}^M, \mu \in \mathbb{R}^N : M \leq N\}$, and δ_{p_i} are: the size of the averaged set, members of the averaged series, an estimated mean, and squared DTW distance under a warping path p_i. In reality, DTW distance is also dominantly used as a similarity measurement in most distance-based temporal data mining techniques (Bagnall et al. 2012, Aghabozorgi et al. 2015, Petitjean et al. 2011). Consequently, to better aid the reader, we will recap the algorithm before presenting averaging techniques relying on it.

2.1 Dynamic Time Warping

The Dynamic Time Warping (DTW) got introduced as an alignment technique for voice recognition systems (Sakoe & Chiba 1978, Itakura 1975). In practice, such systems get expected to identify commands spoken with different styles, speeds, emphasis, etc. Consequently, it is relatively challenging to correctly recognize spoken words using classical distance measurement techniques such as the Euclidean distance. With this in mind, DTW assumes two voice samples are of the form $X = \{x_1, x_2, ..., x_M\}$ and $Y = \{y_1, y_2, ..., y_N\}$. To measure the similarity between the two temporal datasets, it computes two $(M \times N)$ cost values, i.e., local and global cost values. In general, DTW computes local cost values by taking the distance $(d(x_i, y_j))$ between each and every coordinate value of X and Y. In most cases, the distance function $(d(x_i, y_j))$ is set to the squared difference of x_i and y_i. On the contrary, DTW computes global cost values using an $(M \times N)$ matrix which will end up containing a set of warping paths connecting (0,0) to (M, N). In DTW, a warping path gets defined using a group of allowable global cost matrix cells that minimize (2), where $GC_{i,j}$ and $LC_{i,j}$ are the global and local costs associated with cell (i,j). In practice, cells within a DTW warping path get known as DTW-associated coordinates. Thus, in the end, DTW-associated coordinates

formulate the coordinates (x_i's and y_i's) of the aligned series.

$$GC_{i,j} = \begin{cases} LC_{i,j} + \min\{GC_{i-1,j}, GC_{i,j-1}, GC_{i-1,j-1}\}, & \text{if } \{i,j\} \neq 0, \\ LC_{i,j} + \min\{GC_{i,j-1}\}, & \text{if } i=0, \\ LC_{i,j} + \min\{GC_{i-1,j}\}, & \text{if } j=0 \end{cases} \tag{2}$$

Even though DTW warping improved the performance of voice recognition systems, researchers quickly identified some problems associated with its definition (Sakoe & Chiba 1978, Salvador & Chan 2007, Cuturi & Blondel 2017). For instance, DTW gets expected to compute a ($M \times N$) global cost matrix. Thus, the computational complexity of aligning a pair of series is at least quadratic $\mathcal{O}(M \times N)$ (Salvador & Chan 2007, Petitjean & Gançarski 2012). In reality, the computational complexity could quickly become exponential as the number of warped series increases (Petitjean & Gançarski 2012). Furthermore, DTW gives rise to a nonmetric distance that fails to meet the properties of identity and triangular inequality (Ruiz et al. 1985). Moreover, DTW uses a min operation while computing global cost values. To this end, DTW is known to give rise to a nondifferentiable (nonsmooth) distance function (Cuturi & Blondel 2017). With these observations in mind, after DTW's introduction, a range of research got conducted to address a portion of the challenges. For instance, the authors in (Cuturi & Blondel 2017) proposed to use soft minimums to make DTW differentiable. Moreover, to make DTW run faster, some researchers have proposed to compute the global cost matrix either for pre-selected cells (Sakoe & Chiba 1978, Itakura 1975) or to systematically reduce the size of the global cost matrix (Salvador & Chan 2007). In general, since such proposals often focus on addressing a specific problem, algorithms relying on DTW often end up inheriting some of its undesired behaviors.

2.2 Dynamic Time Warping Based Time Series Averaging

The computation of an optimal time series average is studied for over three decades (Petitjean & Gançarski 2012, Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Schultz & Jain 2018, J. Jain et al. 2019, Paparrizos & Gravano 2015, Shapira Weber et al. 2019*a*). Moreover, most of the pioneering works propose to overcome the effects of temporal distortion using DTW (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Petitjean et al. 2011, Schultz & Jain 2018). However, in practice, integrating DTW into the averaging problem proved tricky due to its definition. This is because DTW gets mainly designed to align a pair of univariate temporal datasets. However, in time series averaging, we often desire to simultaneously warp members of an averaged set to obtain a relatively optimal estimate (Petitjean & Gançarski 2012, Shapira Weber et al. 2019*a*). However, upgrading DTW in such a manner is known to be computationally intractable (NP-hard) (Petitjean & Gançarski 2012, J. Jain et al. 2019). To this end, in practice, DTW-based techniques often propose heuristics to overcome this limitation. Overall, depending on the proposed heuristics, we can broadly categorize DTW-based averaging techniques into two, i.e., those relying on sequential and template-based alignments. With this understanding, we next present a summary of DTW-based averaging techniques falling within these two broad categories.

2.2.1 Sequential Approaches

After the introduction of DTW, a range of DTW-based time series averaging proposals approached the averaging problem through sequential warping of the averaged series (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Niennattrakul et al. 2012, Srisai & Ratanamahatana

2009, Ongwattanakul & Srisai 2009). Consequently, given a set of time series which might differ in length, i.e, $S = \{X_1,\ X_2,\ X_3, \ldots, X_N\}$, averaging techniques that are presumed to be sequential first subdivide the averaged set into groups of pair. Following this, the paired series get warped with DTW or its variant. The warping gets performed to generate intermediate estimates often obtained by taking the arithmetic mean of the warped series (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009). Finally, the intermediate estimates get paired, aligned, and averaged until a single estimate remains.

Practically, sequential approaches appear natural since DTW is defined to handle a pair of series at a time. However, in practice, such averaging techniques are often clouded by at least two challenges. First, the quality of the estimates often depends on how pairs get selected. In order to address this issue, some sequential approaches propose to cluster the averaged series in a hierarchical manner (Niennattrakul & Ratanamahatana 2009, Ongwattanakul & Srisai 2009). Thus, this way, the most similar series often get paired first. However, even if the hierarchical clustering minimized the effect of pair selection, the length of the estimates often significantly increased since sequential approaches take the arithmetic mean of DTW-associated coordinates. In reality, this problem gets magnified as the size of the averaged set grows (Niennattrakul & Ratanamahatana 2009, Petitjean & Gançarski 2012, Petitjean et al. 2011). In practice, this becomes a challenge at least in three aspects: interpretability, storage space, and computational time (Niennattrakul & Ratanamahatana 2009, Petitjean & Gançarski 2012, Ongwattanakul & Srisai 2009). With this understanding, alternative DTW-based averaging heuristics approach time series averaging as a registration problem.

2.2.2 Template Based Approaches

To overcome the problems associated with sequential approaches, alternative proposals associated time series averaging with the multiple alignment problem (Bulteau et al. n.d., Petitjean & Gançarski 2012). However, extending DTW in such a manner proved to be NP hard (Bulteau et al. n.d., Petitjean & Gançarski 2012, J. Jain et al. 2019). Consequently, nonsequential techniques often propose to register members of an averaged set to a pre-selected reference (template) that gets updated in an iterative manner (Petitjean et al. 2011, Petitjean & Gançarski 2012, Schultz & Jain 2018, J. Jain et al. 2019). In general, such techniques follow three basic steps. First, given a set of series in $\mathcal{X} = \{X_1, X_2, \ldots, X_N\} : X_i\ \in \mathbb{R}^M$, they often select a template in $\mathbb{R}^M$ that gets initialized: randomly, by taking one of the series (X_i), or by taking medoid of $\mathcal{X}$ (Schultz & Jain 2018, Petitjean & Gançarski 2012). Following this, members of the averaged series get warped to the selected template either one at a time (Schultz & Jain 2018) or in a batch (Petitjean et al. 2011). Finally, in recent proposals, each coordinate value of the template gets updated either by taking the barycenter of DTW associated coordinates (Petitjean et al. 2011, Petitjean & Gançarski 2012) or by using warping and variance matrices (Schultz & Jain 2018). In reality, the outcome of template-based approaches often depends on template initialization and the way the averaged series get warped, i.e., compared to a template. In the context of the latter case, batch-based approaches often get affected by the non-convexity of the averaging objective function, i.e., the Fréchet function (Petitjean et al. 2011, Schultz & Jain 2018). On the contrary, a sequential template-based approach avoids the non-convexity of the Fréchet function by optimizing for individual curvatures. In general, among the two proposals, the sequential template-based approach gets argued to provide better results, i.e., when the averaged set is composed of 50 or more samples (Schultz & Jain 2018). Moreover, since it individually aligned the averaged series to a template, its computational complexity is not related to the size of the averaged set. Consequently, it is comparatively faster than its batch-

based counterparts. However, overall, both template-based approaches performed better than their sequential counterparts (Petitjean et al. 2011, Petitjean & Gançarski 2012, Schultz & Jain 2018). Moreover, in practice, estimates generated by template-based approaches at times get utilized as an initialization step in alternative centroid estimation techniques. For instance, in (J. Jain & Schultz 2018), researchers proposed to extend the Learning Vector Quantization (LVQ) to DTW space, i.e., they proposed the Asymmetric Generalized Learning Vector Quantization (GALVQ). In the proposal, they utilized estimates generated via DBA while initializing the algorithm. The algorithm later updates the initial centroids using an update rule based on a Sigmoid function. Practically, through the updating rule, GALVQ further refines the estimates of DBA such that they become more suitable for one nearest centroid classification (1NCC). In reality, such use of averages estimated via template-based approaches further asserts their use in a more practical sense.

2.3 Alternative Alignment Techniques in Time Series Averaging

In practice, time series averages are often key inputs to various temporal data mining techniques (Bagnall et al. 2012, Bagnall & Lines 2014, Petitjean et al. 2011, Paparrizos & Gravano 2015, Shapira Weber et al. 2019*a*, J. Jain & Schultz 2018). Consequently, there are cases where data mining techniques could propose alignment (transformation) techniques that differ from DTW (Chen & Srivastava 2021, Shapira Weber et al. 2019*a*, Paparrizos & Gravano 2015, Kowsar et al. 2022). For instance, in (Paparrizos & Gravano 2015), the authors proposed to utilize correlation as a means of similarity measurement. They formalized this proposal by defining a correlation-based similarity measurement technique known as Shape Based Distance (SBD). Furthermore, the authors showed that the performance of SBD is statistically indifferent to DTW, i.e., while it gets used under a clustering setup named K-Shape. In this regard, the authors proposed to estimate cluster centroids by aligning cluster members to a reference template such that the alignment maximized their correlation (Paparrizos & Gravano 2015). Thus, in this case, cluster centroids (averages) were estimated using an alignment technique different from DTW. In practice, K-Shape is not the only technique that proposes estimating averages without using DTW. In practice, some alternatives propose alignment or transformation techniques found in functional data analysis (Srivastava & P. Klassen 2016, Kowsar et al. 2022, Chen & Srivastava 2021, Shapira Weber et al. 2019*a*). For instance, (Kowsar et al. 2022) proposed to treat temporal datasets as points on the surface of a sphere. Thus, this way, they defined the geometric shape of a time series which was later seen as a feature space for the augmentation of averages. On the contrary, in (Chen & Srivastava 2021, Shapira Weber et al. 2019*a*), authors proposed to morph temporal datasets using velocity fields. In comparison, (Chen & Srivastava 2021) proposed to utilize Square Root Velocity (SRV) velocity fields. On the contrary, (Shapira Weber et al. 2019*a*) proposed to utilize Continuous Piece-wise Affine (CPA) velocity fields (Detlefsen et al. 2018). However, despite this difference, such approaches can get summarized using three basic steps. First, they transform the time stamps of the averaged series under the guidance of a selected velocity field. Following this, they define the amplitude values for the newly generated time stamps via interpolation. Finally, if the transformation gets performed in the context of averaging, they take the arithmetic mean of the morphed series as an estimate. Consequently, velocity field based approaches end up registering the morphed series to their arithmetic mean. However, due to the interpolation step, there is an inherent assumption that the aligned datasets get defined from an underlying continuous function by taking samples (Srivastava & P. Klassen 2016). Despite this assumption, one interesting point associated with these approaches is that they often integrate their proposal into neural networks. For instance, (Shapira Weber et al. 2019*a*) proposed to intelligently learn the CPA velocity fields through a Temporal Transformer (TT)

layer defined under the framework of a Diffeomorphic Temporal Alignment Network (DTAN). In reality, a TT layer is composed of three components: a localization layer, a parametric grid generator, and a differentiable re-sampler (Shapira Weber et al. 2019*b*). The localization layer first takes a time series $U \in \mathbb{R}^N$. It then utilizes convolutional layers to extract parameter $\theta \in \mathbb{R}^N$ for the parametric velocity field $\nu^\theta(U)$. This parameters is then used to generate an evenly spaced grid in $\mathbb{R}^N$ that is within the ranges of $[-1, 1]$ or $G = (p_n)_{n=1}^N \subset [-1, 1]$. Finally, DTAN uses a differentiable re-sampler to generate the morphed series by re-defining the values of the newly generated time stamps (Shapira Weber et al. 2019*b*). However, in reality, parametric re-sampling by itself will not guarantee registration of the transformed series to their arithmetic means. To this end, DTAN proposed to fine-tune its parametric interpolation by recursively minimizing a per class WGSS loss, i.e., the Fréchet function (1) under euclidean distance.

Generally, the concept of utilizing neural networks for time series averaging is encouraged for at least two main reasons. First, by using neural networks, it is possible to conduct complex nonlinear transformations. These transformations, in turn, open the possibility of utilizing less difficult and off-the-shelf objective functions. Secondly, DTW-based averaging heuristics often do not have a memory of the averaging process (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Petitjean et al. 2011). In other words, the averaging process gets embedded into the estimates. Thus, if a new time series becomes available, the quality of the previously estimated averages is often guaranteed with a costly re-run (Petitjean et al. 2011, Petitjean & Gançarski 2012, Chen & Srivastava 2021). Contrary to this, in practice, neural networks can generalize over a range of unseen data sets. Hence, a neural network-based averaging heuristics could use transfer learning to update its estimate. However, despite this advantage, to the best of our knowledge, proposed neural network-based averaging techniques such as DTAN guarantee the quality of the generated estimates in the morphed space (Shapira Weber et al. 2019*a*,*b*). Consequently, they propose transforming unseen datasets into the morphed space before using the generated estimates. For instance, DTAN uses a trained TT layer to morph unseen datasets. To this end, it is unclear how such approaches perform in situations where the estimates are utilized in the time domain, for instance, as in the case of (Shawel et al. 2020).

2.4 Deep Learning in Time Series Data Mining

Contrary to time series averaging, for other temporal data mining tasks, a range of neural network-based solutions have been proposed (Fawaz et al. 2019, 2020, Iwana & Uchida 2021, Lafabregue et al. 2021*b*, Xie et al. 2016, Shawel et al. 2020). For instance, in (Fawaz et al. 2019), the authors intensively assessed the capability of various neural network architectures in extracting class-specific descriptive features. The investigation included neural networks with fully connected, convolutional, and recurrent layers. In the end, the authors have identified that the *Residual Network* (ResNet) gives better classification accuracies. The ResNet got initially proposed for image classification in (He et al. 2016). Motivated by this investigation, the authors in (Fawaz et al. 2020) have proposed to search for the *AlexNet* of time series classification (Christian et al. 2015). Like its ResNet counterpart, *AlexNet* is one of the most renown convolutional neural network architecture proposed for image classification (Krizhevsky et al. 2012). In this investigation, the authors surpassed the performance of an ensemble of distance-based classifiers proposed in (Bagnall et al. 2012). In another domain, the authors in (Lafabregue et al. 2021*b*) assessed the possibility of clustering time series with deep neural networks. In their investigation, the authors mainly focused on performing clustering using the embedding (latent space features) obtained from autoencoders (Dong et al.

2018) and Generative Adversarial Networks (GAN) (Goodfellow et al. 2014). In addition to investigating the different architectures, the authors also investigated the impact of objective functions (losses) on the quality of extracted embedding and the clusters. In general, the authors identified that the *ResNet* architecture obtained the highest statistical ranking in terms of cluster quality. However, the post-hypothesis test revealed no major statistical difference across different architectures and objective functions (Lafabregue et al. 2021*b*). In reality, these examples cover the broad category of supervised and unsupervised learning. However, in practice, the use of deep neural networks is not only limited to these temporal data mining tasks. For instance, deep neural networks got used in time series data augmentation (Iwana & Uchida 2021), regression (Shawel et al. 2020), etc.

With these observations in mind, in this paper, we propose to estimate time series averages using deep neural networks. In reality, we can think of the averaging problem as an optimization problem. Thus, we believe it would be counter-intuitive to neglect the potential of deep neural networks. Moreover, in practice, neural networks have been utilized in temporal data mining tasks that have a close tie with averaging, for instance, we can take the works given in (Junyuan et al. 2016, Xie et al. 2016) as an example. Motivated by this fact, we propose to augment time series averages from the latent space features (embedding) of neural networks. However, unlike DTAN, we avoid basing or network architecture on specialized layers that could significantly increase the deployment complexity. Consequently, in all of our proposals, we intend to customize and utilize off-the-shelf objective functions and neural network architectures.

3 Methodology

Practically, time series averaging is predominantly approached as an alignment problem (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Petitjean & Gançarski 2012, Petitjean et al. 2011, Schultz & Jain 2018, J. Jain et al. 2019, Shapira Weber et al. 2019*a*, Chen & Srivastava 2021, Paparrizos & Gravano 2015). However, in some cases, averaging proposals suggests generating estimates from a feature space (Kowsar et al. 2022, Terefe et al. 2020). However, such proposals are limited in number. Moreover, to the best of our knowledge, investigations that aimed to estimate averages from features provided by neural networks are scarce (Terefe et al. 2020). Contrary to this, in practice, neural network features space (latent space) got intensively used in various temporal data mining tasks. For instance, in augmentation (Iwana & Uchida 2021), clustering (Junyuan et al. 2016, Lafabregue et al. 2021*a*), classification (Gee et al. 2019), etc. With these in mind, we ask ourselves, what kind of network architectures and objective functions provide suitable latent space representations (embedding) for the augmentation of time series averages? To address this question, we first assess what pioneering time series averaging proposals perform while generating estimates.

If we carefully observe most pioneering averaging techniques, we find them performing three basic tasks. First, they transform (align) the averaged series using different warping techniques, i.e., DTW, diffeomorphism, correlation, etc. The transformation aims to obtain a $\mathbb{R}^N$ to $\mathbb{R}^\tau$ mapping, where $\tau \geq N$. Moreover, in $\mathbb{R}^\tau$, temporal distortion gets expected to be minimized. Thus, after transformation, averaging heuristics can either choose to update a reference template based on the values of the transformed series (Petitjean et al. 2011, Schultz & Jain 2018) or they could propose to take the arithmetic mean previously considered as sub-optimal (Shapira Weber et al. 2019*a*, Chen & Srivastava 2021, Paparrizos & Gravano 2015). Finally, some heuristics could propose a re-transformation technique while others propose to work in $\mathbb{R}^\tau$. For instance, DBA performs registration and estimation in DTW space. However, while

estimating averages, it takes the barycenter of DTW associated coordinates which re-transform the series from $\mathbb{R}^{\tau}$ to $\mathbb{R}^{N}$ (Petitjean et al. 2011). On the contrary, DTAN performs $\mathbb{R}^{N}$ to $\mathbb{R}^{N}$ transformation. Thus, it intends to register, estimate, and evaluate the quality of the averages in the transformed space (Shapira Weber et al. 2019*a*). With these understandings, we ask ourselves, can we mimic these transformations in the latent space of neural networks? Moreover, we also ask ourselves, can we augment time-domain estimates from the latent embeddings?

To answer the first question, we observe what happens to the averaged series after performing a transformation. In this regard, if we consider the averaged series as points in $\mathbb{R}^{N}$, after transformation (warping) we expect the points to be compact (Shapira Weber et al. 2019*a*). This is because, since all warping often gets conducted in reference to common landmark (template), the transformation (warping) is expected to minimize the discrepancy among the averaged series. Consequently, if we want to mimic this effect in the latent space of neural networks, we at least need to have an architecture that identifies features shared among inputs. However, in addition to compactness, we also desire the latent space to be interpretable so that it is possible to generate time domain estimates. With these sets of requirements in mind, one network architecture we identified to align with some of our objectives is an autoencoder (Dong et al. 2018). An autoencoder provides an embedding that is re-transformable to the time domain. This is in line with our objective of providing a time domain equivalent for the latent space estimations. However, we cannot solely expect the reconstruction loss of an autoencoder to give a dense latent embedding. In this regard, previous work has experimentally shown that a basic autoencoder could not achieve separable and dense per class features while used to estimate averages for multi-class datasets (Terefe et al. 2020). With this understanding, we focus on one additional off-the-shelf objective function that favors compact latent representations, i.e., multi-class classification. In practice, a neural network classifier gets expected to extract class-specific latent features that guarantee high classification accuracy (Fawaz et al. 2019, 2020). Thus, in such networks, it is intuitive to expect latent features belonging to a given class to have dense latent representation. This intuition is in line with our objective of registering the latent features to their arithmetic mean. Moreover, in practice, it is not the first time that class (cluster) label information gets used while estimating averages (Shawel et al. 2020, Shapira Weber et al. 2019*a*, Paparrizos & Gravano 2015). For instance, in (Shapira Weber et al. 2019*a*), DTAN used class labels while performing diffeomorphic transformation. Moreover, in (Shawel et al. 2020), averages got estimated on a cluster basis. This fact is also evident in (Paparrizos & Gravano 2015), where averages got estimated for individual clusters of time series. With these observations in mind, given an averaged set with M members, we first propose to utilize a multi-tasking autoencoder setup that minimizes (3), where X^i, $Y^i \in \mathbb{R}^N$ are an input time series and its reconstructed version. Moreover, h_{c_i} and p_{c_i} are an embedding used to uniquely identify a category and *Softmax* activation value associated with a category (c).

$$L_{multi}(X, Y, h, p) = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{N}\sum_{j=1}^{N}(y_j^i - x_j^i)^2 - \frac{1}{M}\sum_{i=1}^{M}\frac{1}{C}\sum_{c=1}^{C} h_{c_i}\log p_{c_i} \tag{3}$$

In general, we utilize the first portion of (3) to guarantee the interpretability of the latent embedding. Moreover, we use the classification portion of (3) to make the latent embedding dense. However, in reality, (3) by itself does not guarantee that estimates preserve shapes observed within an averaged set. Overall, we can consider the multi-tasking approach as an augmentation process. Consequently, we expect the quality of augmented averages to get impacted by at least the following key factors:

– **The architecture of the network:** In practice, we do not expect two different architectures optimizing similar objective functions to perform equally. Thus, different architectural

configurations get expected to have implications on the quality of latent representations.

– **The objective function:** The objective functions optimized by a neural network are a sub-factor that highly influences the type of extracted latent embedding. Even though we expect the multi-tasking approach to perform better than a plain autoencoder, it also has a limitation of its own. For instance, we aim to take the arithmetic mean of the latent space representations as a latent space estimate. However, in (3), there is no way of minimizing the within group square sum (WGSS) between the arithmetic mean and the per-class latent representations. Consequently, we solely rely on the classifier to ensure registration of the latent embeddings to their arithmetic means.

With these in mind, we next present the steps we followed in order to assess the implication of these two key factors.

3.1 Evaluating the Impacts of Network Architectures

We propose to base our investigation on three off-the-shelf neural network architectures, i.e., the VGG16, Residual Network (ResNet), and Inception version two. We propose to utilize these architectures by observing their ease of deployment and their success in other temporal data mining tasks such as classification, clustering, etc (Fawaz et al. 2019, 2020, Xie et al. 2016, He et al. 2016, Christian et al. 2015). With this understanding, we first propose to further investigate the performance of a multi-tasking autoencoder proposed in our previous work (Terefe et al. 2020). To this end, we initially adopt the architecture shown in Figure 3. Based on this architecture, we initially minimize the objective function given in (3). From this point onwards, we call this setup the reduced VGG16 architecture. Following this, we make a slight modification on the layers of the reduced VGG16 and evaluate it using the modified multi-tasking objective functions given in (8)-(6). Hereafter, we call this setup the modified reduced VGG16. In addition to these *VGG16* based architectures, we also propose two additional architectures shown in Figures 4 and Figure 5. The encoder and decoder portion of these additional architectures are also a reduced version of their original implementations, i.e, *Inception* version two (Christian et al. 2015) and *Residual Network (ResNet)* (He et al. 2016). Consequently, afterward, we call these architectures the reduced ResNet and the reduced Inception. In all network arrangements, we call our proposals reduced versions for two main reasons. First, the original implementations of the networks were designed to handle two-dimensional datasets (images). However, for our case, we customized the architectures to process vectors (time series). Secondly, for our proposals, we significantly reduced the number of trainable weights compared to their original implementations. In reality, we perform this parameter reduction for two main reasons. First, we do not wish our approach to rely on relatively huge networks to meet its desired objectives. Secondly, most evaluation datasets have a small number of training samples, i.e., as low as 2 samples per class. Thus, if we directly utilize the original implementations, we have a higher chance of overfitting. Consequently, we propose to keep the layer arrangements used in the original proposals and reduce the number of trainable weights. With these in mind, we present the layer arrangements used in the proposed network architectures.

3.1.1 Proposed Reduced VGG16 Architectures

The *VGG16* got initially proposed by the Visual Geometry Group (VGG) (Simonyan & Zisserman 2015). In its original form, the VGG16 is composed of trainable layers ranging from 11 to 19. However, in practice, the most utilized arrangement is constructed from 13 convolutional layers and

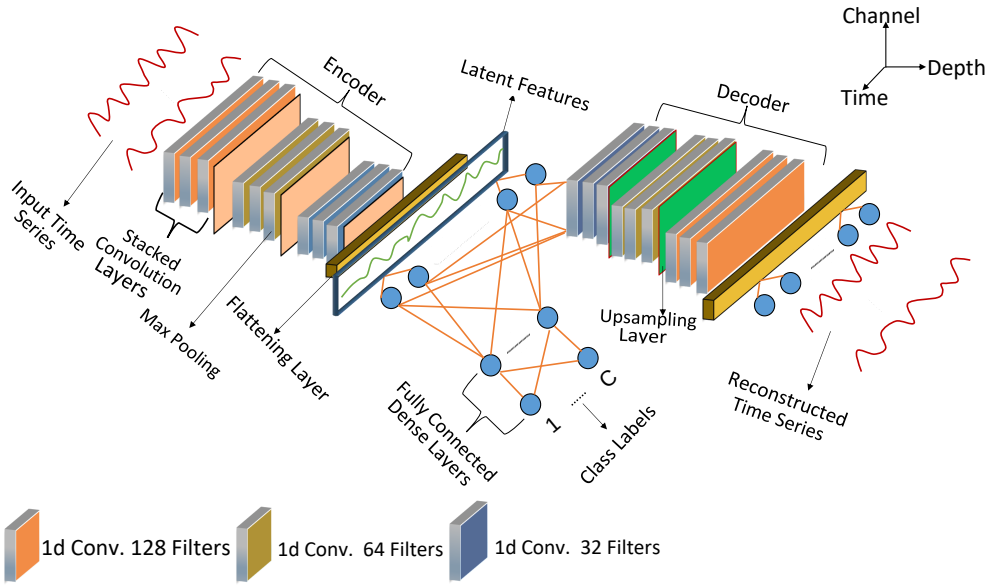


Fig. 3: The reduced *VGG16* multi-tasking autoencoder (Terefe et al. 2020)

3 dense layers. Contrary to this, our proposed reduced VGG16 architecture contains 6 convolutional blocks (stacks) that are equally divided among an encoder and a decoder. Consequently, an encoder or a decoder will have a total of 9 convolutional layers, i.e., as shown in Figure 3. Generally, at the encoder and decoder, a convolutional stack gets built from three convolutional layers. However, the way we configured the stacks differs. In this regard, at the encoder, we have set the first convolutional stack to output 128 channels with each convolutional layer. Following this stack, we have two stacks that output 64 and 32 channels of features, i.e., for each convolutional layer within a given stack. However, for all stacks, we have kept the kernel sizes of the convolutional layers to 3. In addition to these convolutional layers, we have three one-dimensional *MaxPooling* layers with a kernel size of 3. We placed the *MaxPooling* layers at the end of each encoder convolutional stack. Finally, we terminate the encoder by a *Flattening* and a *Dense* layer that has a total of $\frac{L}{4}$ fully connected neurons, where L is the length of the input series. On the contrary, at the decoder, the three convolutional stacks respectively output 32, 64, and 128 channels of features, i.e., per each convolutional layer found within the stacks. Similar to the arrangement at the encoder, we have kept the kernel size of the convolutional layers to 3. However, instead of *MaxPooling* layers, we have two *UpSampling* layers that have a kernel size of 3. We placed the *UpSampling* layers at the end of the first two convolutional stacks. Finally, following the same pattern, the decoder is also terminated by a *Flattening* and a dense layer. However, in this case, the dense layer has a total of L fully connected neurons. Overall, all network layers except the encoder's first convolutional layer, the classifier's last dense layer, and the decoder's last dense layer get configured with *ReLu* activation function. However, the encoder's and decoder's first and last layers use *Linear* activation. Moreover, the classifier's last dense layer uses *SoftMax* activation.

In contrary to the layer arrangement at the encoder and decoder, we have built the classifier using only three fully connected *Dense* layers. The layers respectively have $\frac{90L}{100}$, $\frac{80L}{100}$ *and* C fully connected neurons, where C is the number of category (classes) in the averaged set. In reality, we reused this classifier architecture in all of our proposals. In general, the reduced *VGG16* has 160,800 trainable convolutional layer weights. These weights are relatively small compared to the original

implementation of VGG16 which has millions of trainable weights (Simonyan & Zisserman 2015). However, we should also note that the fully connected dense layers at the encoder and decoder contribute to an additional $32 \times \frac{L^2}{108} + \frac{L}{4}$ and $1152 \times \frac{L^2}{4} \times +L$ trainable weights. Consequently, when the length of the averaged series (L) increases, the two dense layers significantly contribute to the number of trainable weights.

In reality, we have reused the reduced VGG16 architecture while assessing an alternative multi-tasking objective function. However, in this case, we slightly adjust the architecture in two aspects. First, due to the channel arrangement of the reduced VGG16, the trainable weights of the decoder are slightly larger than the encoder. This gives rise to an asymmetric encoder and decoder setup. However, in most practical cases, we expect an autoencoder to have a sense of symmetry (Dong et al. 2018). Thus, we first change the channel arrangement used at the decoder's stacks from (32, 64, 128) to (128, 64, 32). Furthermore, we also change the *stride* of the the last encoder's *MaxPooling* layer from 2 to 1. Thus, this way, datasets with smaller lengths can get processed without modifying the kernels of the encoder's *MaxPooling* layers. Finally, we also change the unintelligent *UpSampling* layers with transposed convolutional layers. We set the transposed convolutional layers with a kernel and stride of 3 and 2. These changes of parameters reduce the number of trainable weights at the encoder's and decoder's dense layers to $32 \times \frac{L^2}{36} + \frac{L}{4}$ and $288 \times \frac{L^2}{4} + L$.

3.1.2 Proposed Reduced Inception Version Two Architecture

In this setup, we maintain the classifier architecture used in the reduced VGG16 setup and modify the encoder and the decoder. In this regard, we first change the convolutional stacks with *Inception* modules (Christian et al. 2015). In general, an Inception module is composed of four parallel convolutional blocks. The four convolutional blocks are, in turn, built from convolutional layers that have kernel sizes of 1, 2, 3, and 5. Moreover, within a convolutional block, convolutional layers with kernel sizes different from 1 get their inputs through convolutional layers that have a kernel size of 1. On the contrary, a convolutional layer with a kernel size of 1 gets its input directly from either a predecessor Inception module or the averaged set. Consequently, within an Inception module, convolutional layers with a kernel size of 1 serve as a memory link. In the proposed reduced *Inception* setup, we limit the maximum convolutional kernel size to five for two main reasons. First, we want to make the *Inception* network light-weighted (reduced version). Secondly, in most temporal datasets, class-specific features are often short-lived, i.e., class-specific features are often steep edges, narrow peaks, and troughs (Fawaz et al. 2019). Thus, we strongly believe that large kernel size (receptive fields) will extract undesired common features into the latent representations. Hence, it will significantly impact the separability of the latent features. This, in turn, impacts the quality of latent means and their time domain re-projection. With this understanding in mind, we terminate an *Inception* module with a *Concatenation* layer. Additionally, we systematically concatenate the outputs of each convolutional block within an *Inception* module. This is because we want the reduced Inception setup to be weight-wise comparable to the reduced *VGG16* arrangement. In this aspect, for instance, we let each convolutional block of the first encoder's *Inception* module to output 32 channels of features. Thus, when they get concatenated, they output 128 channels. Using the same approach, we let the remaining two encoder's *Inception* modules output concatenated channels of 64 and 32. Similarly, the decoder is composed of *Inception* modules that output concatenated channels of 128, 64, and 32. However, at the decoder, we have transposed convolutional layers, i.e., between Inception modules. We set the channel size of these layers to be equal to the concatenated channel size of their preceding *Inception* module. Moreover, we set the kernel and stride size of these layers

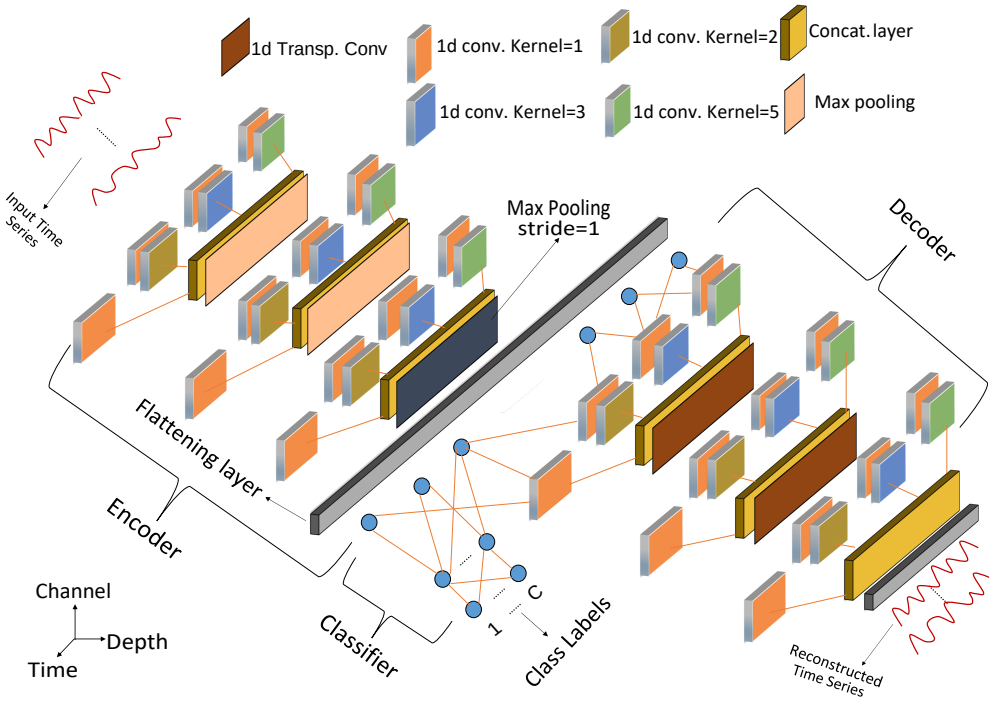


Fig. 4: Reduced *Inception* version two multi-tasking autoencoder.

to 3 and 2. However, we set the kernel and stride of the non-transposed convolutional layers to 3 and 1. Moreover, the kernel and stride size for the encoder's *MaxPooling* layers get set to 3 and 2, i.e., except the last *MaxPooling* layer. For this layer, we kept the kernel size to 3 while setting its stride to 1. Finally, we kept the activation function of the layers similar to the ones used in the reduced VGG16 setup. Overall, with these configurations, there are 53,776 trainable convolutional layer parameters. Moreover, the *Dense* layers in the encoder and the decoder account for $32 \times \frac{L^2}{36} + \frac{L}{4}$ and $288 \times \frac{L^2}{4} + L$ trainable weights. This is equivalent to the weights found at the dense layer of the modified reduced VGG16 architecture.

3.1.3 Proposed Reduced ResNet Architecture

The final architecture we propose to evaluate is the reduced *ResNet* multi-tasking autoencoder shown in Figure 5. For this network, we re-utilize the convolutional stacks used in the modified reduced *VGG16* setup. However, in the *ResNet*, we have skip connections that mix the output of a predecessor convolutional stack to the output of its immediate successor. These skip connections serve as memory links as the network's depth increases (He et al. 2016). Moreover, the ResNet combines the outputs of skip connections with the outputs of convolutional stacks using a one-dimensional *Addition* layer. To this end, unlike the reduced VGG16 setup, we built the convolutional stacks of the reduced ResNet from four convolutional layers. We added the extra convolutional layer to account for the dimensional mismatch between skip connections and the outputs of the convolutional stacks. For instance, in the reduced ResNet, a skip connection originating from the encoder's first convolutional stack will have 128 channels. However, its successor convolutional stack only has 64 channels. To account for such mismatches, within each convolutional stack, we add a fourth convolution layer that output 32 channels of features. However, other than this, we maintained most configurations similar to the modified reduced *VGG16*. Consequently, we

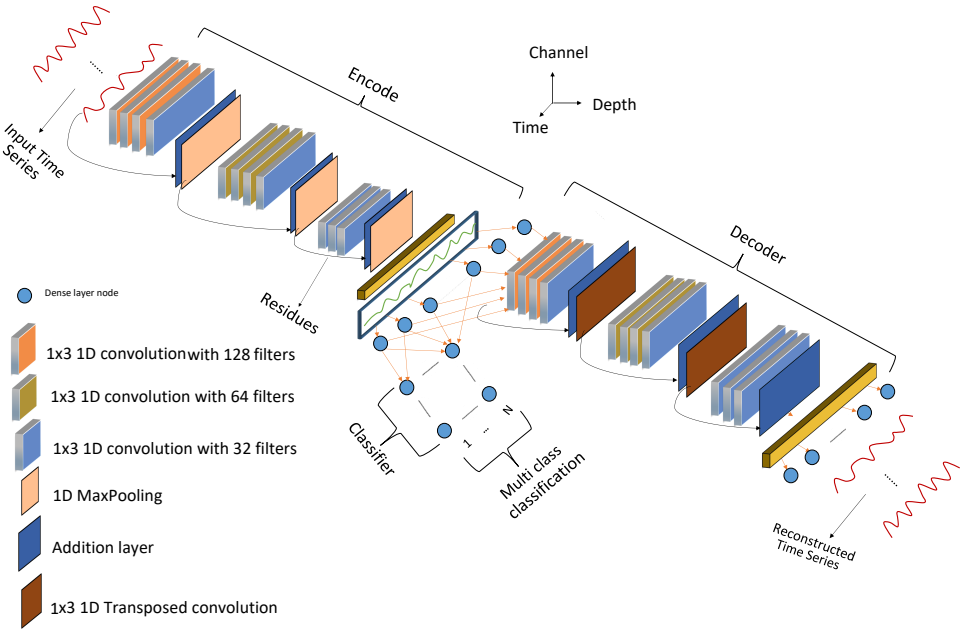


Fig. 5: Reduced *ResNet* multi-tasking autoencoder

set the channels of the remaining three convolutional layers to 128, 64, or 32. Moreover, we kept the kernel sizes of the *MaxPooling*, convolutional, and transposed convolutional layers to 3. Additionally, we also kept the *stride* of the *MaxPooling* and transposed convolutional layers to 2. However, the encoder's last *MaxPooling* layer used a stride of 1. Finally, we have used *ReLu*, *Linear*, and *Softmax* activation functions in a manner similar to the once used in the reduced VGG16 setup. In general, the reduced *ResNet* architecture has 157,792 trainable convolutional layer weights. Moreover, we kept the number of trainable weights for the encoders and decoders *Dense* to be similar to the ones found in the *Inception* and modified *VGG16* setups.

In conclusion, in all of our proposed network setups, we have used *Linear* and *ReLu* activation functions. In practice, these activation functions and neural networks, in general, are known to be sensitive to weight initialization. In order to account for this factor, we respectively used *He normal* and *Glorot normal* initialization for the *Relu* and *Linear* activation functions (Glorot & Bengio 2010, Kaiming et al. 2015).

3.2 Improving the Multi-tasking Objective Function

Beside network architecture, another factor expected to impact the quality of latent features is the objective (cost) function. In this regard, we have observed minor limitations on the generic multi-tasking objective function given in (3). In reality, in the latent space, we have proposed to take the arithmetic mean of the latent features as an estimate. We make this proposal with one statistical implication in mind. In reality, an arithmetic mean is one optimal distribution parameter in maximum likelihood estimation, i.e., given the distribution under consideration is multivariate Gaussian. In other words, the parameters of a multivariate Gaussian distribution ($G(\mu_{est},\ Cov_{est})$) that maximizes the likelihood of observing K multivariate random variables that are in $\mathbb{R}^{\tau}$ gets

computed as:

$$\begin{cases} \mu_{est} = \frac{1}{K}\sum_{i=1}^{K} Z_i \\ Cov_{est} = \frac{1}{K}\sum_{i=1}^{K}(Z_i - \mu_{est})(Z_i - \mu_{est})^T \end{cases} \tag{4}$$

where $Z_i \in \mathbb{R}^{\tau}$ is a latent space representation of an input dataset. Thus, statistically, taking an arithmetic mean corresponds to estimating the mean of a Gaussian distribution that maximizes the likelihood of observing the latent embeddings. In practice, such Gaussian distribution assumptions get commonly found in generative models such as VAE, GAN, Gaussian mixture models, etc (Diederik P. & Max 2014, Goodfellow et al. 2014). However, even though latent arithmetic means have statistical implications, (3) does not have a way to guarantee latent embeddings are registered to their arithmetic means. On the contrary, we mainly relied on the classifier to register the latent embeddings to their respective means. This is because we expect the classifier to induce the extraction of class-specific features expected to be dense. However, in this case, we have no control over whether embeddings get registered to their latent means or not. To address this issue, we propose to force the encoder to also optimize for a per-class Within Group Suqred Sum (WGSS) loss, i.e., the third entry in (8). The WGSS loss gets expected to increase the density of class-specific latent features. In addition to this improvement, we also observed that (3) tries to minimize a reconstruction loss using Mean Squared Error (MSE). However, in practice, MSE is known to be prone to outliers. This is because the reconstruction errors for outliers significantly get magnified through the squaring operation. Consequently, such outliers could significantly push gradient vectors in an undesired direction. In practice, one possible way of mitigating this problem is using Mean Absolute Error (MAE). However, as compared to MSE, the absolute function provides a relatively flatter objective function which, in turn, could affect the speed of network convergence. Moreover, MAE has a very narrow optimal point which could lead to an oscillation while computing gradients. In addition to these limitations, MAE and MSE errors confine reconstruction to a median value. This is because, at the decoder, MAE and MSE penalize over and under reconstructions equally. To this end, under such objective functions, there is a higher probability that a decoder will re-project latent arithmetic mean into a time domain arithmetic mean. Thus, we propose to change the reconstruction loss given in (3) with a quantile regression loss given in (5), where $\lambda \in [0, 1]$ and $\{X, Y\} \in \mathbb{R}^{\mathbb{N}}$ are an input time series and its reconstruction.

$$L_{reg}(\lambda,\ X,\ Y) = \frac{1}{N}\sum_{j=1}^{N} \max\{\lambda(y_j - x_j),\ (\lambda - 1)(y_j - x_j)\} \tag{5}$$

In quantile regression, we say over and underestimations have occurred when $y_i - x_i > 0$ and $y_i - x_i < 0$. In reality, in quantile regression, we can have three kinds of estimations: over estimation ($\lambda < 0.5$), under estimation ($\lambda > 0.5$), or median estimation (MAE) for $\lambda = 0.5$. Consequently, we could systematically use λ values in order to control the behavior of a decoder. In reality, we are mainly using the decoders of the proposed architectures to re-project latent space estimates that have no time domain ground truth. Thus, it is intuitive to have a relaxed reconstruction constraint that allows different time domain re-projection possibilities. In this aspect, quantile regression allows us to control the location of a median reconstruction line with over, under, or median regressions. However, in order to utilize this possibility, we propose to compute quantile regression for a pair of λ values. The rationale behind this proposal is discussed in the training setups section.

In reality, we also propose to utilize quantile regression in the latent space of the proposed multi-tasking setups. In this regard, we propose to compute the quantile difference between

the latent embedding of the input series and their regressed form, i.e., the second term of (7). Practically, we propose to make this loss visible to only the decoder portion of the multi-tasking setups. This is because we believe the introduction of this loss could further enhance the decoder's capability of interpreting latent space embeddings. In this aspect, in quantile regression, the decoder tries to estimate the values of the input series as closely as possible. Consequently, if we select a λ value that does not introduce major shape distortion, the decoder outputs regressed series that do not exactly map to the input series. Thus, if we project these series back to the latent space, we expect to get latent mappings that are near neighborhoods of the latent embeddings of the input series. Hence, by computing latent space quantile loss, we expect to gain at least two advantages. First, the decoder will now get feedback from the encoder on how well it is regressing the input series. This in turn creates better harmony in the overall encoder-decoder arrangement. In addition to this, the decoder will now be forced to learn regressions that take neighborhood embeddings into account. This in turn will contribute positively to the re-projection of latent means which are expected to be within the neighborhood of the averaged series latent embeddings.

With these technicalities in mind, given a group of temporal dataset $S_t = \{X^1,\ X^2,\ \ldots,\ X^M\} : X^i\ \in\ \mathbb{R}^N$ that have C categories, their latent space representation $S_l = \{Z^1,\ Z^2,\ \ldots,\ Z^M\}\ : Z^i\ \in\ \mathbb{R}^\tau$, and their regressed output $R_t = \{Y^1,\ Y^2, \ldots,\ Y^M\} : Y^i\ \epsilon\ \mathbb{R}^N$; the objective function optimized by the encoder, decoder, and classifier becomes:

$$L_{classifier} = -\frac{1}{M}\sum_{i=1}^{M}\frac{1}{C}\sum_{c=1}^{C} h_c^i \log p_c^i \tag{6}$$

$$L_{decoder} = \frac{1}{M}\sum_{i=1}^{M} L_{reg}(\lambda,\ X_{pred}^i,\ X_{true}^i)\ +\ \frac{1}{M}\sum_{i=1}^{M} L_{reg}(\lambda,\ Z_{pred}^i,\ Z_{true}^i) \tag{7}$$

$$L_{encoder} = L_{classifier} + L_{decoder} +\ \frac{1}{C}\sum_{c=1}^{C}\ (\frac{1}{K_c}\sum_{l=1}^{K_c}\frac{1}{\tau}\sum_{j=1}^{\tau}(z_j^l - \mu_j^c)^2\ ) \tag{8}$$

In equations (6)-(8), $\mu_i\ \in\ \mathbb{R}^\tau$, M and K_c correspond to: the latent mean of the c^{th} class (category), the total number of averaged series, and the number of series in the c^{th} class. Moreover, Z_{pred}^i and Z_{true}^i are the latent space embedding of a regressed and an input time series. With these technicalities in mind, we proceed and present the experimental setups.

4 Experimental Setups

In practice, while evaluating a neural network-based optimization setup, we at least need to consider two factors. First, it is often advised to evaluate proposed approaches using a range of datasets. In practice, such evaluations will minimize the probability of reporting outcomes that could be data biased. Moreover, with such rigorous tests, we avoid outcomes that could be random (mere chance). Besides such rigorous evaluations, it is also advised to access the impact of hyper-parameters on experimental outcomes. In this subsection, we present how we have addressed these suggestions.

4.1 Summary of Data Sets

In all experimental evaluations, we have utilized datasets obtained from the UCR archive. The archive contains 128 multi-class datasets with train and test splits (Chen et al. 2015). However,

Table 1: The 114 UCR datasets categorized based on their source.

No.	Data source	Total data sets	dimension ranges	class ranges
1	Device measurements	8	90-2000	2-10
2	ECG measurements	6	96-750	2-42
3	EOG measurements	2	1250	12
4	EPG measurements	2	601	3
5	Hemodynamics measurements	3	2000	52
6	HRM-PCR measurements	1	201	18
7	Images	32	46-2709	2-60
8	Motions	17	150-1882	2-12
9	Power consumption	1	144	2
10	Sensor measurements	20	24-1639	2-39
11	Synthetic (simullated)	8	60-1024	2-8
12	spectrographs or chemical analysis	8	235-1751	2-5
13	SEMG measurements	4	1500-2844	2-6
14	Pedestrian Traffic count	2	24	2-10

11 datasets contain samples that are variable in length. The datasets found in this category are: *AllGestureWiimote* {*X, Y, Z*}, *GestureMidAir* {*D1, D2, D3*}, *GesturePebble* {*Z1, Z2*}, *PickupGestureWiimoteZ*, *PLAID*, and *ShakeGestureWiimoteZ*. In reality, most of these datasets correspond to records of different gestures. In practice, different gestures take different duration to be completed, i.e., depending on the type and the individuals committing them. To this end, the extracted temporal datasets have different lengths. In addition to these 11 variable-length datasets, 3 datasets contain missing values. The datasets falling in this category are: *DodgerLoopDay*, *DodgerLoopGame*, *DodgerLoopWeekend*. These datasets correspond to the count of the number of vehicles on a freeway located in *Los Angeles*. Overall we have avoided experimenting on these 14 data sets since they require further processing. With these in mind, in Table 1, we have summarized the remaining 114 datasets into 14 different categories.

4.2 Training Setups

We have implemented our proposed architectures using Python's Tensorflow version 2.4. Generally, we have assessed our proposals using two training setups. First, we train the modified reduced VGG16, reduced Inception, and reduced ResNet architectures for 1500 epochs, i.e., using 4 different λ pair values, and once on each 84 UCR archive datasets. We use this training setup to identify the better-performing architecture. Following this, we train the better performing architecture using: 25 repeated trials, 4 λ pair values, and 114 UCR archive datasets. In this training setup, the selected network gets trained for 100 repeated trials on each UCR archive dataset. We use this training setup to thoroughly assess the performance of our proposed approach as compared to currently available averaging techniques. In addition to this intensive network training, we also evaluate the implication of encouraging over and underestimations in quantile regression. To make this assessment, we have trained the better-performing architecture using 6 λ pair values that favors the encouragement of over (under) estimations. However, for the evaluation, we have only used 84 UCR archive datasets. Moreover, we have trained the network once on each UCR archive dataset.

In all evaluations, we consider λ values as hyper-parameters that get selected randomly. In general, we propose to use two sets of λ pair values, i.e., $\lambda_{conf1} = \{(0.15, 0.85), (0.25, 0.75), (0.35, 0.65), (0.5, 0.5)\}$ and $\lambda_{conf2} = \{(0.85, 0.85), (0.75, 0.75), (0.65, 0.65), (0.15, 0.15), (0.25, 0.25), (0.35, 0.35)\}$. In reality, we propose λ to be a pair of values to create scenarios where the quantile regression loss either encourages or discouraged over (under) estimations. In this aspect, we consider λ_{conf1} to be a configuration that discourages over (under) estimations equally. On the contrary, we consider λ_{conf2} to be a configuration that encourages over (under) estimations. For instance, lets consider the λ pair values $(0.25, 0.75)$, i.e., from λ_{conf1} , as an exmple. In this case, when we put $\lambda = 0.25$ in (5), overestimation is penalized by 25% and underestimation is penalized by 75%. On the contrary, when we put $\lambda = 0.75$ overestimation is penalized by 75% and underestimation by 25%. However, if we take the maximum of the two computations, we end up penalizing over (under) estimation by 75%. However, if we take similar computational steps and take the λ pair values $(0.85,\ 0.85)$, i.e., from λ_{conf2}, as an example, we will end up encouraging underestimations since they only get penalized by 15%. However, had we selected a λ pair with values lower than 0.5, we would end up encouraging overestimation. In order to practically realize these scenarios, we compute the quantile regression losses given in (7) and (8) using (9). In (9), λ_1 and λ_2 are the individual λ pair values within one of the λ configuration (λ_{conf1} and λ_{conf2}). Moreover, U could be either a latent representation or an input series. On the contrary, V could either be the latent embedding of a regressed series or a regressed input series.

$$L_{quantile} = \frac{1}{M} \max \Big(\sum_{i=1}^{M} L_{reg}(\lambda_1, U^i,\ V^i),\ \sum_{i=1}^{M} L_{reg}(\lambda_2, U^i,\ V^i) \Big) \tag{9}$$

With these technicalities in mind, We have trained all of our proposed networks using the training datasets given in the UCR archive. Moreover, when the size of the training split is sufficient enough, we used a mini-batch size of $\frac{M}{4}$, where M is the size of a training set. We then used the trained network to project the training set into the latent space. Following this, we estimated latent means by taking the arithmetic mean of latent embeddings of each class. Moreover, we used the decoder portion of the multi-tasking set up to project the latent means into the time domain. We also used the trained network to project the UCR archive's test datasets into the latent space. Finally, we evaluate the representativeness of the estimated latent and time domain means using the evaluation technique discussed in the following subsection. [1]

4.3 Evaluation Technique

Evaluating the quality of time series averages is quite tricky. This is because, the different averaging techniques estimate the means in different spaces. For instance, DTW based averaging techniques estimate means in DTW space (Gupta et al. 1996, Niennattrakul & Ratanamahatana 2009, Petitjean et al. 2011, Schultz & Jain 2018). On the contrary, DTAN estimates the mean after morphing the averaged series with an affine transformation (Detlefsen et al. 2018, Shapira Weber et al. 2019*a*). In practice, some pioneering proposals used DTW space WGSS (1) as an evaluation metric (Petitjean et al. 2011, Schultz & Jain 2018). However, in such proposals, compared averaging techniques are mostly DTW-based. To this end, they ended up comparing the performances of averaging techniques that optimized the same objective function. Moreover, the WGSS got compared in DTW

[1] The Python implementation of the proposed architectures, the training setup, and 1NCC evaluation can be found at the following github repository: https://github.com/tsegaterefe/Estimating-Time-Series-Averages-from-Latent-Space-of-Multi-tasking-Neural-Networks/tree/main

space which is a space where the estimated means got registered. Consequently, we can safely such comparisons to be fair. However, in practice, averages do not always get estimated either in DTW space or using DTW (Shapira Weber et al. 2019*a*, Paparrizos & Gravano 2015, Kowsar et al. 2022). When this is the case, researchers often propose to assess the representativeness of estimated means using one nearest centroid (1NCC) classification (Shapira Weber et al. 2019*a*, Paparrizos & Gravano 2015, Kowsar et al. 2022). However, in such cases, the distance function utilized for the 1NCC determines the neutrality of the comparisons. For instance, if DTW distance gets utilized, the evaluation ends up favoring DTW-based estimates.

With these pros and cons in mind, we propose to evaluate the quality of our estimates using 1NCC. In reality, we selected 1NCC for two main reasons. First, the multi-tasking setup is not directly minimizing a WGSS loss. Second, we found 1NCC to be meaningful in the context of assessing the representativeness of a mean. In practice, we often estimate time series averages with an underlying application in mind. For instance, classification (Bagnall et al. 2012), clustering (Paparrizos & Gravano 2015, Petitjean et al. 2011), forecasting (Shawel et al. 2020), etc. In this aspect, 1NCC gives more sense than WGSS since it assesses the quality of a mean, not only, compared to the group it gets estimated from but also compared to other groups within an averaged set. With this understanding in mind, for our proposals, we performed two kinds of 1NCC, i.e., latent space and time domain. For the latent space 1NCC, we used: latent means estimated from training sets, the latent space representation of test datasets, and euclidean distance. On the contrary, for the time domain 1NCC, we used: the UCR archive's test datasets, time domain means estimated from a training set and DTW distance. To this end, in the time domain, estimates generated with our proposals are disadvantaged compared to estimates generated with DTW. In general, for the first training setup, we identified the best 1NCC accuracy obtained in the latent spaces of the different architectures. We then identify its corresponding time domain accuracy. Finally, we compare these accuracies with accuracies obtained with alternative techniques. However, after identifying the best performing architecture and performing the 25 repeated training trials, we define four types of latent space and time domain 1NCC accuracies that are obtained on the test split: maximum, minimum, median, and mean. Thus, in the extended evaluation, we compare these accuracies to the maximum 1NCC accuracies obtained with the alternative averaging techniques. In addition to these evaluations, we have also reassessed the multi-tasking setup that is minimizing (3) using: the reduced *VGG16* architecture, 114 UCR datasets, and the training setup given in (Terefe et al. 2020).

4.4 Evaluated Alternative Averaging Techniques

We have compared the performance of our estimates with estimates generated with seven different average estimation techniques, i.e., Arithmetic, DBA, SDBA, SSG, KShape, DTAN, and AGLVQ (Petitjean et al. 2011, Cuturi & Blondel 2017, Schultz & Jain 2018, Paparrizos & Gravano 2015, Shapira Weber et al. 2019*a*, J. Jain & Schultz 2018). To generate the estimates of these alternative averaging techniques, we have used *Tslearn* (Tavenard et al. 2020) implementation of DBA, SDBA, SSG, KShape, and DTW. Moreover, we have computed SDBA using five different γ values, i.e., for $\gamma = 10e-3, 10e-2, 10e-1, 1, 10$. However, for AGLVQ, we have adopted its Java implementation given in (J. Jain & Schultz 2018) without modifying most of the hyper-parameters given in the implementation. In this regard, the only hyper-parameter we changed is the number of estimated mean corresponding to a cluster (class), i.e., we set this parameter to one. Overall, while evaluating alternative techniques, we selected and reported outcomes that gave the best 1NCC

result on the test split. However, since there is no standardized implementation of DTAN, we have used the outcomes reported in (Shapira Weber et al. 2019*b*). According to (Shapira Weber et al. 2019*a*,*b*), DTAN was trained on 84 UCR archive datasets for 2500 epochs. Moreover, to conduct the training, DTAN utilized two regularization and smoothing setups. After training the transformer network, the authors morphed the training datasets of the UCR archive and generated estimates in the morphed space. Moreover, they also used the trained transformer network to morph test datasets of the UCR archive. Following this, they conducted 1NCC accuracy using the estimates generated from the training sets and the morphed representation of the test datasets. Finally, they reported the best 1NCC accuracies obtained on the test split. To this end, in this paper, we present two comparisons. First, we compare the best outcomes associated with our approaches, arithmetic mean, DTAN, DBA, SDBA, KShape, GALVQ, and SSG. Following this, we exclude DTAN and compute averages using arithmetic, DBA, SDBA, KShpae, and SSG on 30 additional UCR archive datasets. We then compare the outcomes of our proposals and the six mentioned averaging techniques on 114 datasets. Overall, for all the alternative proposals, we estimated averages using the training datasets. Moreover, since the UCR archive is a multi-class, we estimate the means using the training samples of each class. Additionally, since KShape got proposed for clustering, we have configured it to cluster each class individually, i.e., a cluster size of 1. Thus, in the end, we have taken the cluster centroids as the estimates for each class. Finally, while conducting the 1NCC for DBA, SDBA, SSG, and KShape, we utilized 25 repeated trials and DTW distance. However, the 1NCC for DTAN got conducted using euclidean distance (Shapira Weber et al. 2019*b*). At this point, we like to mention that KShape proposed to use SBD as a distance metric while identifying cluster members and updating cluster centroids (Paparrizos & Gravano 2015). However, since we cannot find a separate standardized implementation of SBD, i.e., in Python, we used DTW as a distance metric while conducting the 1NCC. However, in (Paparrizos & Gravano 2015), SBD was found to be statistically indifferent to constrained DTW (Sakoe & Chiba 1978). Consequently, we expect that DTW will not significantly affect the assessment of KShape.

5 Results

In this section, we evaluate the representativeness of averages generated with different averaging techniques using 1NCC. In general, we have divided our assessments into two major sections. In the first section, we make a preliminary assessment to identify which of our proposals is performing better. For the evaluation, we have trained our proposed architectures using λ pair values from λ_{conf1} and 84 UCR archive datasets. We then selected the best 1NCC accuracy despite the λ pair values given in λ_{conf1}. Following this assessment, we thoroughly evaluate the better-performing architecture using 114 UCR archive datasets, 25 repeated training trials, and maximum, minimum, mean, and median 1NCC accuracies.

5.1 Preliminary Assessment of Proposed Approaches

We will start our assessment of the averaging techniques by observing wins, ties, and losses obtained on 84 UCR archive datasets. In this analysis, we assume an averaging technique is winning if it has a 1NCC accuracy that is better than its competitors. However, we consider a tie has occurred if at least two averaging heuristics have the same classification accuracy. Finally, an averaging technique is assumed to be losing if there is at least one technique performing better than it. With this in mind, in Table 2, we have summarized the wins, ties, and losses associated with the different

Table 2: Summary of wins, loss and ties using 1NCC accuraciesobtained on 84 UCR archive datasets. The 1NCC was performed using averages estimated form the UCR archive's training sets.

Averaging techniques	Wins	losses	ties
AGLVQ	4	75	5
Arithmetic	0	84	0
DBA	0	82	2
DTAN	7	75	2
SDBA	4	80	0
SSG	5	78	1
KShape	2	81	1
Inc_Quant_Lat	**4**	**76**	**4**
Inc_Quant_Time	0	83	1
MT_ENC_LAT	4	79	1
MT_ENC_Time	1	82	1
Res_Quant_Lat	**12**	**69**	**3**
Res_Qunat_Time	0	84	0
VGG_OU_Qunat_Lat	**16**	**62**	**6**
VGG_OU_Qunat_Time	2	80	2
VGG_Quant_Lat	**12**	**67**	**5**
VGG_Quant_Time	0	82	2

averaging techniques. In Table 2, *MT_ENC_LAT* (*Time*) represents the latent space (time domain) 1NCC outcomes obtained with the estimates of the reduced VGG16 architecture that minimized 3). On the contrary, *Res_Quant_Lat* (*Time*), *Inc_Quant_Lat* (*Time*), and *VGG_Quant_Lat* (*Time*) represent the latent space (Time domain) 1NCC accuracies obtained with the estimates of the reduced multi-tasking: ResNet, Inception, and VGG16 architectures minimizing (8) to (6). Moreover, when there is no *OU* within the abbreviations, the architectures have used λ pair values from λ_{conf1}. In other words, the architectures have discouraged over (under) estimations equally.

According to Table 2, in the latent space, the modified reduced *VGG* and *ResNet*(*VGG_Quant_Lat* & *Res_Quant_Lat*) architecture appears to be winning on more datasets, i.e., when we consider setups that discourage over (under) estimation equally. However, in terms of losses, the reduced *VGG*(*VGG_Quant_Lat*) performed better than its *Resnet* counterpart. However, overall, the modified reduced VGG16 performs better than all of its counterparts, i.e., while it encourages over (under)estimations (*VGG_OU_Quantile_Lat*). In other words, when the reduced modified VGG16 gets trained using λ pair values from λ_{conf2}. However, in reality, Table 2 might be misleading in a statistical sense. For instance, compared to *VGG*16_*Quant_Lat*, the *Res_Quant_Lat* might be losing on two additional datasets with a small accuracy margin. Moreover, it might obtain more wins with comparatively smaller average and median accuracy. With these in mind, we further assess the 1NCC accuracies using two additional statistical evaluation techniques, i.e., a box whiskers plot and a hypothesis test. In this regard, Figure 6 shows the box whisker plot associated with the 1NCC accuracies. Moreover, Table 3 summarizes the statistics of the plot. In Figure 6, on one hand, the lower and upper whiskers indicate the lowest and highest classification accuracies. On the other hand, the start and end of a box demarcate the 25% and 50% quantiles of the 1NCC accuracies obtained over 84 UCR datasets. With these technicalities in mind, Table 3 shows that the modified reduced *VGG16* (*VGG_Quant_Lat*) is better than its *ResNet* counterpart (*Res_Quant_Lat*). For

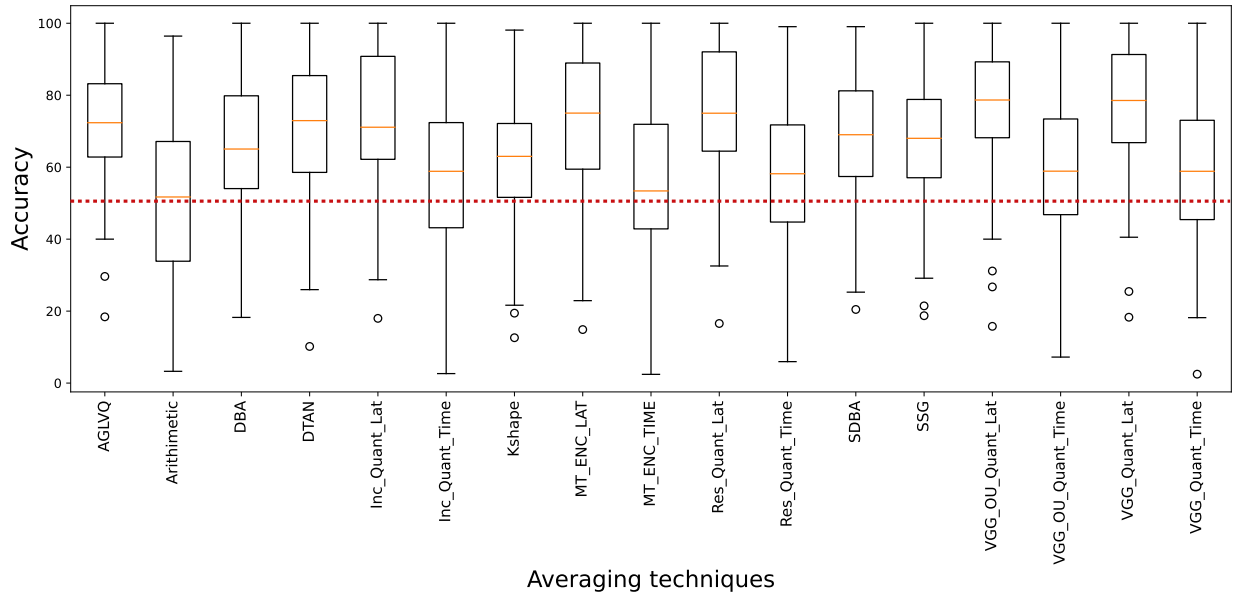


Fig. 6: Box whisker plot of the one nearest centroid classification accuracies of different averaging technique on 84 datasets.

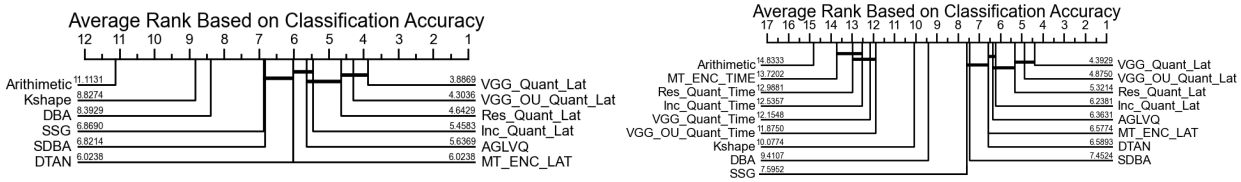
instance, *VGG_Quant_Lat* obtained a median accuracy of 78.54%. On the contrary, over the 84 datasets, the *Res_Quant_Lat* obtained a 75% median 1NCC accuracy. Moreover, *ResNet*'s lower quartile starts at 64.45%. In this regard, the modified reduced *VGG16*'s lower quantile starts at 66.82%. Consequently, we have selected the modified reduced *VGG16* architecture to evaluate the impact of encouraging over and underestimation. The outcomes of this experiment is denoted as *VGG_OU_Quant_Lat* (*Time*) in Tables 2 and 3. In reality, the two tables show that encouraging over (under) estimation gives better 1NCC results. In this aspect, the outcomes of the *VGG_OU_Quant_Lat* that could get considered as *"bad"* (the first 25% quartile) are between $40 - 68.18\%$. Furthermore, 50% of the classification accuracies are within $68.18 - 89.27\%$ and have a 78.67% median accuracy. This performance gets closely followed by the latent space classification results of *VGG_Quant_Lat*. In this case, the first 25% quartile ranges between 40.53% and 66.82%. Moreover, 50% of its classification accuracies are within $66.82 - 91.31\%$ and have a 78.54% median accuracy. These results indicate that the modified reduced version *VGG*16 architecture is a natural choice for further investigation. However, for the *VGG16*, the difference between the statistics of encouraging (discourage) over (under) estimations appear to be relatively close. Consequently, it is quite unclear which of the two approaches we should pursue. In order to answer this question, in the next subsection, we conduct hypothesis tests (Demšar 2006).

5.1.1 Hypothesis Test

The box whisker plot gives a broad statistical inference into the 1NCC accuracies. However, it does not evaluate the degree of closeness among individual classification accuracies. Thus, we cannot conclude if one averaging heuristic is either statistically different or similar using only a box whisker plot. Consequently, we first propose to conduct a Friedman average rank test on the 1NCC accuracies that are obtained with the different averaging techniques. Moreover, as a post hypothesis test, we propose the Wilcoxon rank sum test. This test is used to evaluate if a pair of averaging techniques are statistically different or not. A detailed explanation of these statistical hypothesis

Table 3: Statistical summary for the Box-whisker plot given in Figure 6. L_Q, U_Q, L_W and U_W are respectively the lower quartile, upper quartile, lower, and upper whiskers.

Averaging Technique	L_Q (25%)	U_Q (75%)	L_W	U_W	$Median$
AGLVQ	62.82	82.13	40.00	100	72.36
Arithmetic	33.87	67.14	3.27	96.43	51.72
DBA	54.05	79.84	18.25	100	65.04
DTAN	58.55	85.45	25.97	100	72.94
SDBA	57.41	81.22	25.27	99.05	69.02
SSG	57.07	78.83	29.14	100	68.02
KShape	51.61	72.14	21.64	98.09	63.01
Inc_Quant_Lat	62.18	90.80	28.73	100	71.10
Inc_Quant_Time	43.17	72.39	2.64	100	58.86
MT_ENC_LAT	59.44	88.95	22.91	100	75.03
MT_ENC_TIME	42.85	71.92	2.43	100	53.40
Res_Quant_Lat	**64.45**	**92.05**	**32.54**	**100**	**75.00**
Res_Quant_Time	44.75	71.28	5.96	99.05	58.17
VGG_OU_Quant_Lat	**68.18**	**89.27**	**40.00**	**100**	**78.67**
VGG_OU_Quant_Time	46.80	73.40	7.23	100	58.86
VGG_Quant_Lat	**66.82**	**91.31**	**40.53**	**100**	**78.54**
VGG_Quant_Time	45.43	73.03	18.18	100	58.86



(a) Hypothesis tests using registered space 1NCC accuracies

(b) Hypothesis test including time domain 1NCC accuracies

Fig. 7: Hypothesis test of various averaging techniques using 84 UCR datasets.

tests can be found in (Demšar 2006). Furthermore, the Python implementation of these tests was adopted from (Fawaz et al. 2019). In overall, the Python implementation incorporates methods that are useful for the visualization of the hypotheses tests. In practice, the outcomes of such hypothesis evaluations are often shown using a critical difference (CD) diagram (Demšar 2006).

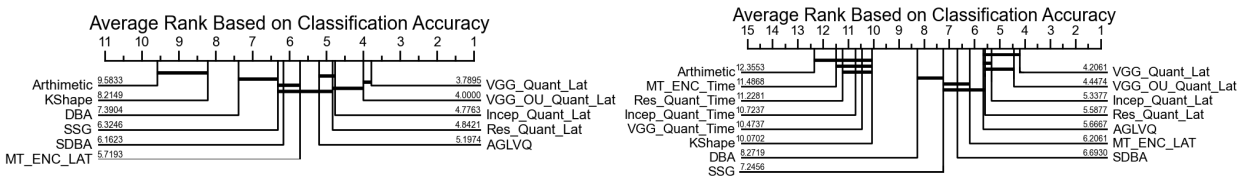
In CD diagrams, we indicate the average ranks of the Friedman test using 90° bent lines that are extended from a horizontal scaled line. The scaled horizontal line indicates the Friedman average ranks of the compared technique. On the contrary, the outcomes of post-hypothesis tests are indicated using bold horizontal lines that connect two Friedman average rank lines. In general, two Friedman average rank lines are connected if they are found to be statistically indifferent (Demšar 2006). With these understandings, for better clarity, we divided the hypothesis evaluations into two categories. In the first category, i.e., Figure 7a, we compared 1NCC accuracies which we consider as accuracies obtained in the registered space of the evaluated averaging techniques. In this aspect, we consider DBA, DTAN, SDBA, SSG, KShape, and latent space of our proposals as registered space outcomes. In general, for DBA, SDBA, SSG, and KShape the means are estimated either in

DTW space or using a metric that is equivalent to DTW. Moreover, the 1NCC is conducted using DTW distance. Consequently, we consider the 1NCC as a task performed in the registered space of the averaging techniques. Similarly, for DTAN, the authors transformed the test sets before the 1NCC (Shapira Weber et al. 2019*a*). Consequently, we also take this 1NCC as a task performed in a registered space. Finally, for our proposals, we consider the latent space to be the registered space since we are mimicking multiple alignments in this space. On the contrary, in the second category, we included arithmetic mean and the time domain estimates of our proposals into the hypothesis tests, i.e., as sown in Figure 7b. With these said, Figures 7a and 7b shows that in the latent space the modified reduced *VGG16* architecture outperforms most of our proposals. However, the post-hypothesis test indicates that there is no statistical difference between VGG_Quant_Lat and Res_Quant_Lat. This in turn indicates that, over the 84 datasets, it is better to discourage over (under) estimations equally. However, despite the post hypothesis equivalency, the modified reduced VGG16 architecture obtained a better Friedman average rank. Consequently, we selected the modified reduced VGG16 architecture which equally penalized over (under) estimations for further investigation. In addition to these observations, the hypothesis test shows that most of our proposals achieved better latent space registration, i.e., compared to the state-of-the-art (DTAN). With this in mind, we place our focus on the performances of the time domain estimates.

According to Figure 7b, the time domain 1NCC accuracies obtained by our proposals are below the outcomes of the DTW-based averaging techniques. However, we find this to be no surprise for two reasons. First, DTW averaging techniques initially register their time domain estimate in DTW space. Thus, it is logical to expect the DTW-based averaging techniques to perform better in DTW space. Second, the assessment given in Figure 7b is based on a one-shot experiment. Consequently, there is a higher likelihood that we are capturing the median or mean performances of our proposed approaches. Contrary to this, the outcomes of the alternative proposals correspond to the maximum 1NCC accuracies selected from repeated trials. However, even under this difference, we obtained better estimations, i.e., compared to the arithmetic mean. In this regard, $VGG\_OU\_Quant\_Time$ obtains time domain estimates that performed comparatively close to DBA. This is in line with our argument of minimizing the constraint on the time domain re-projection. However, it should also be noted that, according to the post hypothesis test, Res_Quant_Time and $VGG\_OU\_Quant\_Time$ are considered statistically similar. Such similarity is also evident between Inc_Quant_Time and MT_ENC_TIME. With these observations in mind, before proceeding to the further evaluation of the modified reduced $VGG16$, we first assess if the introduction of additional one-shot experiments could reveal changes in the hypothesis evaluations.

5.1.2 Hypothesis Test with Additional Data sets

In neural network-based approaches, it is advised to test proposals using a range of datasets. In practice, such tests get expected to reduce the chances of making biased conclusive remarks. With this in mind, in this subsection, we combine the experimental results of the 84 data sets with 1NCC accuracies associated with 30 additional datasets. Thus, we re-evaluate the null hypothesis of *"every averaging technique is equal"* on 114 data sets. However, in this evaluation, we exclude $DTAN$ since we do not have its standardized implementation and results corresponding to the additional 30 datasets not reported in (Shapira Weber et al. 2019*b*). Nevertheless, for the rest of the averaging techniques we follow the same approach and first compared the classification accuracy in the registered spaces of the averaging techniques, i.e., Figure 8a. We then included the outcomes of the time domain re-projection as shown in Figure 8b respectively.



(a) Hypothesis test using registered space 1NCC accuracies

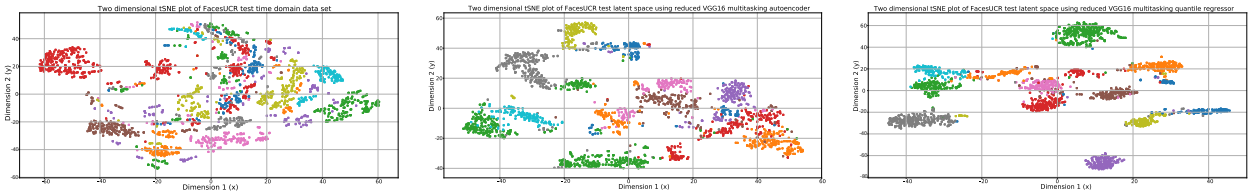
(b) Hypothesis test including time domain 1NCC accuracies

Fig. 8: Hypothesis test of averaging techniques using 114 UCR datasets.

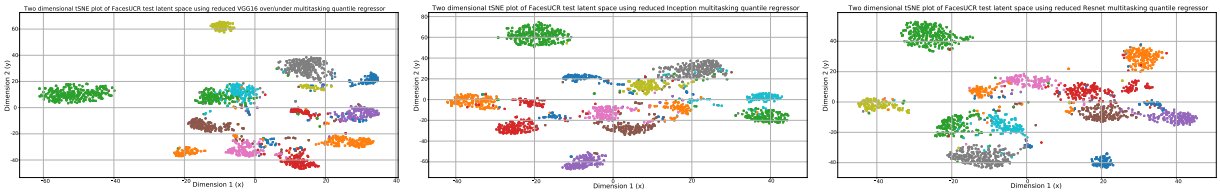
According to Figure 8a, the latent space of the modified reduced *VGG* architectures still outperform all of the alternative averaging techniques in a statistically different manner. However, in this case, there is no statistical difference between the performances of estimates generated while the architecture encouraged or discouraged over (under) estimations. Furthermore, the *Inception* setup obtained a better Friedman rank as compared to its *ResNet* counterpart. Additionally, in the time domain, the new evaluation reveals that the reduced *ResNet* performs similarly to KShape. We find this to be encouraging since the 1NCC accuracies associated with KShape correspond to maximum 1NCC accuracies obtained from 25 repeated trials.

In general, the primary assessments reveal the performance of the modified reduced *VGG* architecture optimizing for quantile regression and multi-class classification is our best performing setup. However, in practice, the *Inception* and *ResNet* architectures are expected to give better results as compared to the *VGG* architecture (Fawaz et al. 2019, 2020). In this regard, we identified three reasons that could be contributing to the difference. First, we have not completely adopted the original architectures of the *Inception* and *ResNet* networks. This customization, at times, has helped us to avoid overfitting that could have clouded our judgment. For instance, in Figure 8a and Figure 8b the *Inception* managed to improve its Friedman's rank due to its smaller number of trainable weight. Another contributing reason for the deviation from the expectation could be the objective function that gets optimized. In practice, most works utilized the *Inception* and *ResNet* for time series classification in contrary to the multi-tasking setup we proposed (Fawaz et al. 2019, 2020, Lafabregue et al. 2021*b*). The final reason could be that, in the latent space, we desired to avoid the impact of temporal distortion. However, the *Inception* and *ResNet* setups have skip connections that leak the effects of temporal distortions. To this end, in most cases, the latent representations of the *Inception* and *ResNet* appear to be less dense. For instance, we can consider Figure 9 which shows the t-distributed Stochastic Neighbor Embedding (tSNE) (Der Maaten & Hinton 2008) projection of the *FacesUCR* dataset (Chen et al. 2015) as an example. Even though tSNE loosely defines cluster density, we can use it as a visual aid if the projection is over the same dataset and if similar tSNE hyper-parameters are used (Der Maaten & Hinton 2008). In this aspect, for the plots, we have used the test split of *FacesUCR* dataset and the trained networks of our proposals. Moreover, we have configured t-SNE's *perplexity*, learning rate, and the maximum iterations to 44, 10, and 5000. We selected a perplexity size of 44 since the minimum class size has 44 members. Moreover, this value is within the ranges of 5 to 50, i.e., as suggested in (Der Maaten & Hinton 2008).

In general, in Figure 9, all the latent space projections of multi-tasking approaches have obtained separable and compact latent embeddings. However, if we further zoom in and look at the per class latent embeddings, we can see that the *VGG16*'s latent embeddings are relatively

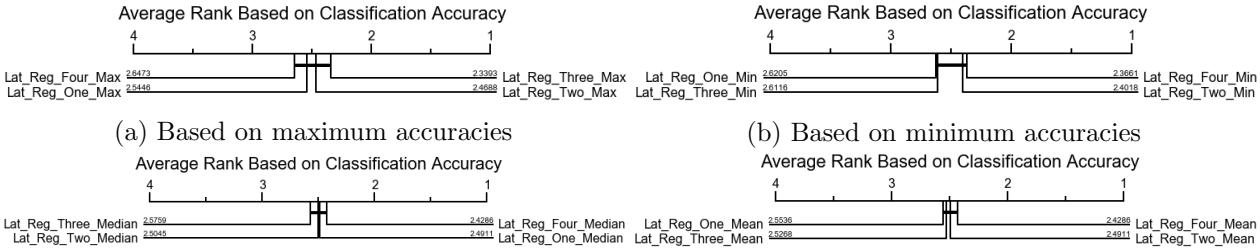


(a) Time domain. (b) *MT_ENC_LAT*. (c) *VGG_Quant_LAT*



(d) *VGG_Quant_OU_LAT* (e) *Inc_Quant_LAT* (f) *Res_Quant_LAT*

Fig. 9: t-SNE projection of the UCR *FacesUCR* dataset. Time domain (9a) and latent spaces of our proposals (9c- 9f)



(a) Based on maximum accuracies (b) Based on minimum accuracies

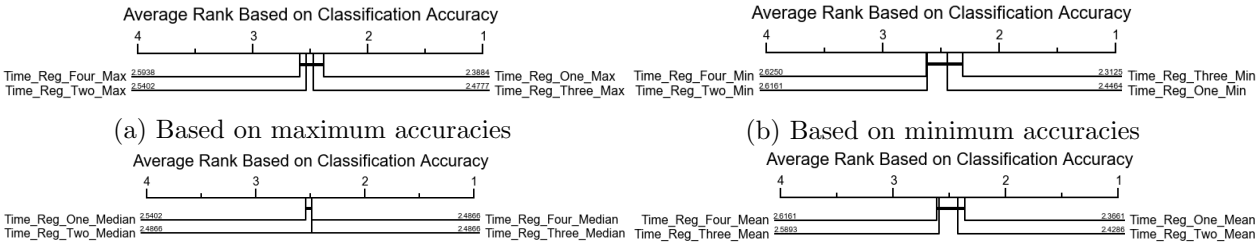
(c) Based on median accuracies (d) Based on mean accuracies

Fig. 10: Performance evaluation of λ pair values based on latent space 1NCC accuracies.

dense. We find this to be in line with our initial argument that the skip connections leak temporal distortion into the latent space.

5.2 Extended Evaluation of The Modified Reduced VGG16 architecture

In neural networks, layer weight initialization is one major source of randomness. Moreover, neural networks have hyper-parameters that contribute to variations in reported outcomes. For instance, for the proposed quantile regression-based autoencoders, the λ pair values are one key hyperparameter. However, evaluating all possible combinations of such hyperparameters for all of the proposed architectural setups is computationally demanding. To this end, in this subsection, we have selected the modified reduced *VGG* architecture and trained the network for 25 repeated trials in order to identify: which of the λ pair values gives better performance, to test the stability of our proposals, and to better capture maximum obtainable performances. To make these assessments, we have trained the selected architecture using the λ pair values given in λ_{conf1}. With this at hand, we first assess which of λ pair values give a better performance. We then assess the stability of the proposal by observing the standard deviation (σ) across the 1NCC accuracies obtained with the 25 repeated trials. Finally, we observe if the repeated trials have better captured the maximum obtainable accuracies. In all of these evaluations, we use the term



(a) Based on maximum accuracies

(b) Based on minimum accuracies

(c) Based on median accuracies

(d) Based on mean accuracies

Fig. 11: Performance evaluation of λ pair values based on time domain 1NCC accuracies.

According to Figure 10a, in the latent space, the third λ pair values give the highest of the maximum obtainable accuracies. However, on the post-hypothesis test, it is statistically equivalent to the fourth λ pair values. Moreover, if we consider minimum, median, and average latent space classification accuracies, the fourth λ pair values give better performance. However, if we focus on the minimum and mean latent space accuracies, we can see that it is statistically indifferent to the performances of the first λ pair values. On the contrary, on the median latent space classification accuracies, the post hypothesis equivalency is between the fourth and third λ pair values.

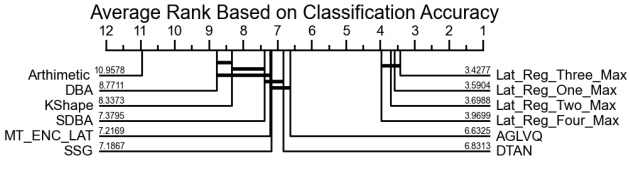
In addition to these observations, Figure 11 shows the performance evaluation of the λ pair values based on time domain classification accuracies. Contrary to the latent space, the first λ pair values obtained the highest possible maximum 1NCC accuracies. However, according to the post-hypothesis test, it is statistically indifferent to the performances of the fourth λ pair values. Furthermore, if we focus on the median and mean classification accuracies, the fourth and first λ pair values performed better. However, in both cases, they are found to be statistically equivalent in the post-hypothesis test. Finally, when we focus on the minimum time domain classification accuracies, the third λ pair values obtained the best worst-case results. However, according to the post-hypothesis test, it is statistically equivalent to the fourth λ pair values. Overall, we found the first λ pair values give better results.

Before making any conclusive remarks about the λ pair values, we will continue with our assessment and evaluate the stability of the network. For this assessment, we can safely consider a small σ as an indication for a narrow distribution curve over the 1NCC accuracies obtained with the 25 repated trials. This, in turn, implies a higher likelihood of reproducible outcomes. With these understandings in mind, in Table 4, we have summarized the average standard deviation (σ) associated with 1NCC accuracies obtained with the different λ pair values. In general, on average, the latent space and time domain accuracies have a standard deviation that is below 5%. In other words, on each URC archive dataset, the average variation of 1NCC accuracies within one standard deviation is below $\pm$5%. This, in turn, indicates that, on each repeated trial, our proposal generates comparatively similar estimates. Thus, we can safely assume that the modified reduced $VGG16$ is stable for practical considerations.

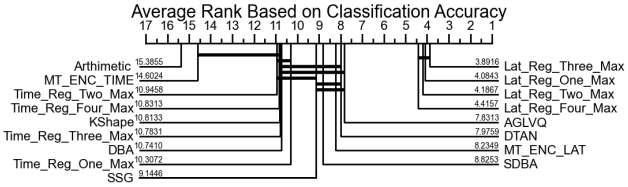
With these in mind, we will finalize our re-evaluation with the hypothesis tests shown in Figures 7 and 8. In this regard, we will first present the re-evaluation of the hypothesis tests on 84 datasets, i.e., including the reported outcomes of DTAN. Following this, we include the 1NCC accuracies of 30 additional datasets and re-evaluate the averaging techniques without DTAN. With this said, Figure 12 shows the hypothesis re-evaluation using 84 datasets. Conforming to our

Table 4: Average standard deviation of accuracies for different λ pairs.

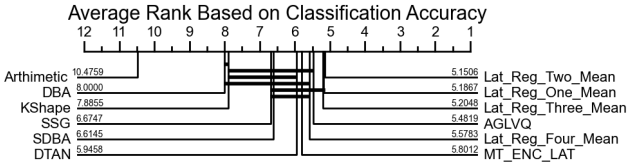
λ pairs	Latent Space $\pm\sigma$ in %	Time Domain $\pm\sigma$ in %
(0.15, 0.85)	2.757	3.903
(0.25, 0.75)	3.246	4.257
(0.35, 0.65)	2.946	4.234
(0.5, 0.5)	3.220	4.673



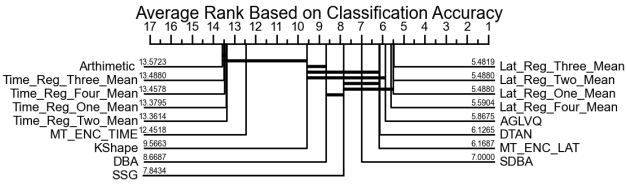
(a) Based on Max. latent space accuracies



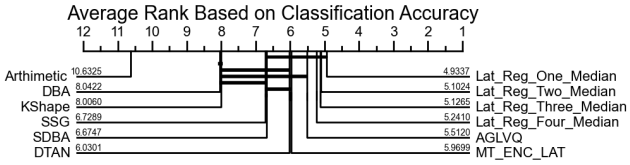
(b) Based on max. time domain accuracies



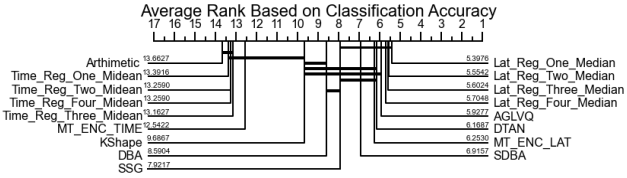
(c) Based on mean latent space accuracies.



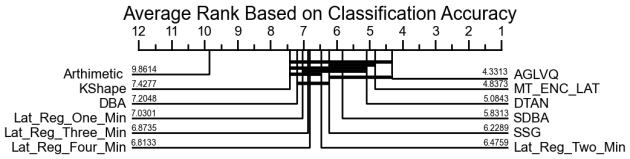
(d) Based on mean time domain accuracies.



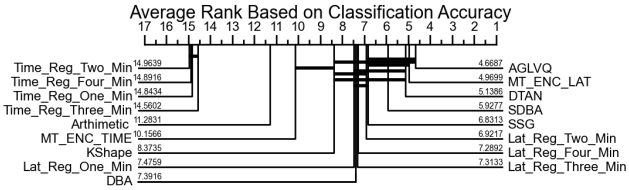
(e) Based on median latent space accuracies.



(f) Based on median time domain accuracies.



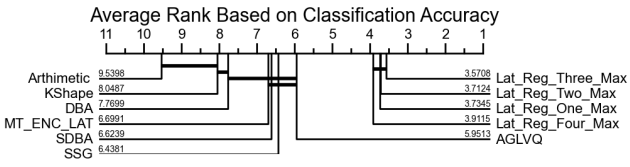
(g) Based on min. latent space accuracies.



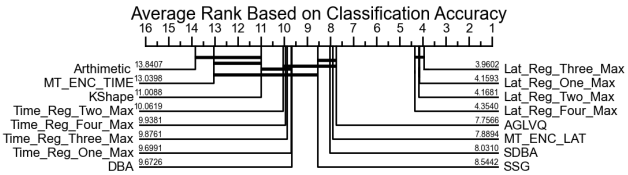
(h) Based on min. time domain accuracies.

Fig. 12: Hypothesis tests for the modified reduced $VGG16$ with 84 datasets, maximum, minimum, median, and average 1NCC accuracies.

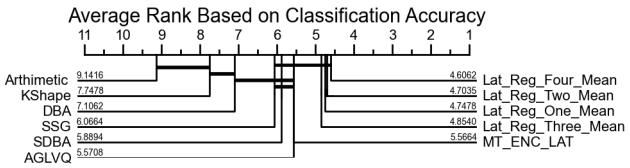
previous argument, repeated training has better captured the maximum obtainable accuracies. To this end, in Figure 12b, we can show that, in the time domain, our estimates could outperform the estimates of DBA and KShape. This is very encouraging since our time domain re-projections have no prior knowledge of the DTW space. Additionally, the repeated trials further validate the better registration of the latent space embeddings to their arithmetic means. In overall, in Figures 12a and 12b we have used maximum classification accuracies. However, in practice, we often obtain such maximum accuracies after some repeated trials. Thus, such assessments could not give us a better picture of the common performance of the proposed technique. To this end, in Figures 12 we also have evaluated the performance of our selected proposal using minimum, median, and mean accuracies. However, it should also get noted that for the alternative proposals



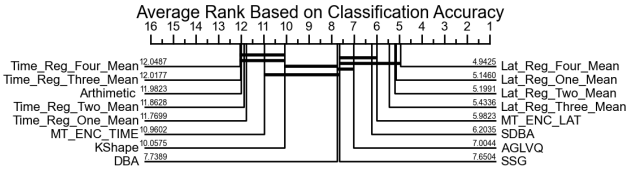
(a) Based on max. latent space accuracies.



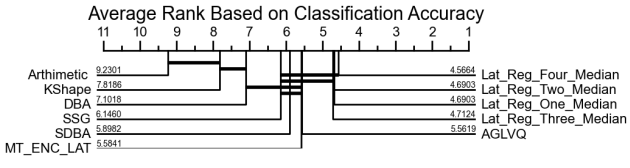
(b) Based on max. time domain accuracies.



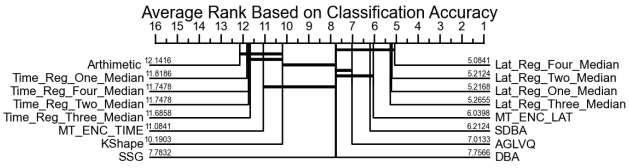
(c) Based on mean latent space accuracies.



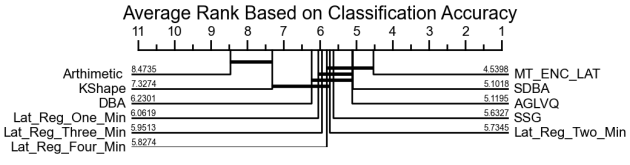
(d) Based on mean time domain accuracies.



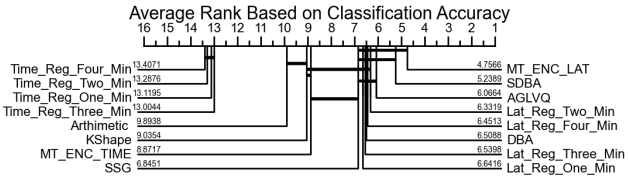
(e) Based on median latent space accuracies.



(f) Based on median time domain accuracies.



(g) Based on min. latent space accuracies.

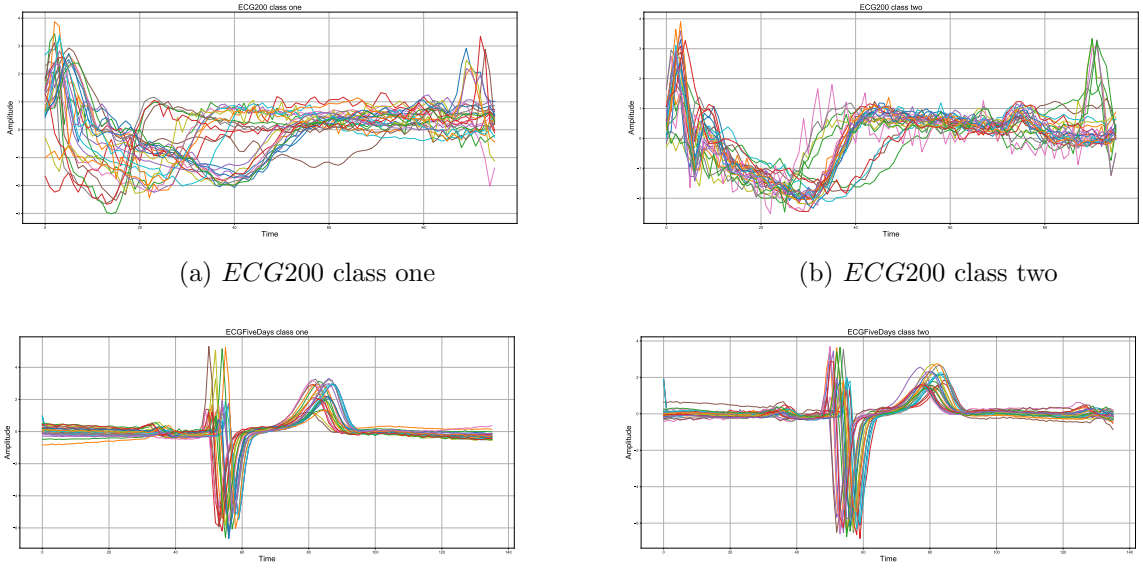


(h) Based on min. time domain accuracies.

Fig. 13: Hypothesis tests for the modified reduced $VGG16$ with 114 datasets, maximum, minimum, median, and average 1NCC accuracies.

we are still using maximum 1NCC accuracies. In this regard, Figure 12c and Figure 12e shows that we are able to provide a registration space that is better than the one provided by DTAN. This is encouraging since we compare maximum (outlier) accuracies with median and mean accuracies. However, in the time domain, the mean and median accuracies performed lower than DBA but better than a time domain arithmetic mean. However, our estimates are still performing at least as best as the estimates of KShape. Finally, in the worst case scenario, some of our latent space registrations are statistically equivalent to DTAN, i.e., as shown in Figure 12g and Figure 12h. However, in this case, the time domain re-projections performed poorly compared to the arithmetic mean.

We will conclude the hypothesis re-evaluation with the assessment based on 114 datasets. According to Figure 13a, in the latent space, our proposed approach outperforms the best of the DTW based approaches (AGLVQ). Moreover, in the time domain, our estimates still outperforms estimates generated with KShape and as best as AGLVQ, i.e., as shown in Figure 13b. Furthermore, in the latent space, we are able to obtain performances that are better than all of the alternatives with median and mean classification accuracies, i.e., as shown in Figures 13c and 13e. However, in these cases, the time domain re-projections are lower than DBA. However, time domain re-projections are still performing as well as KShape. Finally, in the worst case scenario or Figures 13g and 13h, some of the λ setups achieved latent space accuracies that are at least statistically equivalent to SDBA. However, overall, the time domain re-projections performances are lower than the time domain arithmetic mean.



(a) *ECG*200 class one (b) *ECG*200 class two

(c) *ECGFiveDays* class one (d) *ECGFiveDays* class two

Fig. 14: The UCR *ECG*200 and *ECGFiveDays* datasets.

In general, in terms of λ setups, $\lambda = (0.15, 0.85)$ often provided better performances. However, we suggest the deployment of $\lambda = (0.35, 0.65)$, i.e., if estimates that give balanced performances in either the time domain or latent space get desired. This is because according to (8), this λ setup gives more room for the classifier. Consequently, the classifier has more say on what kind of latent space embedding is selected. In this aspect, it utilizes a (0.65) 65% penalty factor as compared to (0.85) 85%. Hence, when undesired features get encountered, the classifier and the WGSS loss will have more say in (8). In general, since we are aiming to mimic registration in latent space and since the classifier one contributor, $\lambda = (0.35, 0.65)$ gives better latent space results. However, in the time domain, it is relatively relaxed in the context of the decoder's loss function (7). Consequently, even though it provides better time domain estimates, i.e., as compared to most λ pair values, it often does not outperform $\lambda = (0.15, 0.85)$. In this aspect, the (0.85) 85% penalty provides a less constrained re-projection space. Thus, it often can avoid significant shape distortion in regressed estimates while leaving a small room for over (under) estimations compared to an MSE loss.

5.3 Visual Demonstration of Estimated Averages

We will conclude the results section by presenting a visual demonstration of the time domain re-projections. For this demonstration, we will use time domain estimations associated with the *ECG200* and *ECGFiveDays* datasets. These datasets often get selected as demonstrative examples in the reports of previous proposals (Petitjean & Gançarski 2012, Shapira Weber et al. 2019*a*). With these in mind, in Figure 14, we have shown samples taken from the test split of these data sets (Chen et al. 2015). In the figure, it is evident that the data sets present simple and short-lived descriptive shapes. Moreover, there is a visible temporal distortion that misaligns members of the averaged sets.

In general, Figure 15 depict how different averaging techniques estimated averages for the

Table 5: One nearest centroid classification accuracy for the *ECG*200 and *ECGFiveDays* datasets.

Averaging Techniques	*ECG*200 accuracy in %	*ECGFiveDays* accuracy in %
Arithmetic	67	52.96
AGLVQ	41.5	79.91
DBA	72	65.85
SDBA	73	67.02
SSG	77	69.80
KShape	72	67.18
MT_ENC_TIME	72	58.65
VGG _ Quant_Time	**73**	59.69
VGG_ OU _ Quant_Time	70	**76.66**
Res_Quant_Time	70	68.06
Inc_Quant_Time	68	64.58

two classes of the *ECG*200 and *ECGFiveDays* datasets. In reality, the averages got estimated from the training split of the two datasets. Overall, if we compare the shapes of the estimated averages, a time domain arithmetic mean shows a significant distortion compared to the shapes observed in the original datasets. On the contrary, the two DTW-based averaging heuristics, i.e., DBA and SDBA, and KShape appear to be preserving shapes observed in the original datasets. However, Figure 15b and Figure 15d shows minor shape distortion. One contributing reason behind the distortion is that DTW often associates a single coordinate of a series with multiple coordinates of its counterpart. In practice, this is commonly called *pathological associations*. When we focus on the estimates of our proposals, the multi-tasking autoencoder appears to give relatively distorted estimations. However, we find it capable of generating estimates shape-wise better than a time domain arithmetic mean. On the contrary, the modified reduced *VGG16* appears to provide an estimate that resembles the estimates of SDBA. This visual equivalency is valid when it encourages over and under estimations (*VGG_OU_Quant_Time*). This gets expected since our hypothesis evaluations show the setup has a better average rank in the time domain. To make the visual assessment full, we conclude this section by presenting the classification accuracies associated with each estimation shown in Figure 15. According to Table 5, discouraging over and underestimations with the VGG16 architectures (*VGG_Quant_Time*) appears to give better results on the short-lived features of the *ECGFiveDays*. On the contrary, encouraging over or underestimation worked better for the slowly transitioning *ECG*200 datasets, i.e., with *VGG_OU_Quant_Time*.

6 Conclusion

In this paper, we proposed to approach time series averaging as a generative problem. To meet this objective, we proposed to pair off-the-shelf objective functions with the appropriate neural network architectures. In this regard, we proposed to utilize multi-tasking autoencoder setups that minimized quantile regression, WGSS, and classification losses. With these proposals, we showed that it is possible to learn latent embeddings which are registered to their arithmetic mean. Moreover, based on these latent embeddings, we are able time domain estimations that are better than the estimates generated by a previously proposed multi-tasking autoencoder minimizing reconstruction and multi-class classification losses. With this result, we are able to empirically support our initial argument that relaxing the reconstruction constraint gives better time domain

re-projection. In addition to this encouraging latent space performances, we are able to show that our time domain estimates could outperform estimates generated with some of the alternative averaging techniques. However, we found the evaluation of the time domain estimates to be relatively challenging. The challenge mainly arises from the unavailability of a *"neutral (unbiased)"* distance metric. For instance, in this paper, we paired the re-projected estimates with DTW distance to conduct 1NCC. However, in reality, the utilization of DTW distance favors estimates generated with DTW since they get estimated in DTW space. Overall, in the time domain, the only comparison we consider fair is the comparisons made between the time domain arithmetic mean and our re-projected estimates. This is because both estimation techniques do not have prior information about DTW space. Overall, comparatively, our proposed approach outperforms time domain arithmetic mean in a statistically significant manner, i.e., on both 84 and 114 univariate time series. Moreover, some of our time domain estimates outperformed averages generated with DBA and KShape, i.e., while we compare maximum 1NCC accuracies. We find this encouraging since the two averaging techniques get favored by the underlying distance metrics.

In overall, in this paper, we have systematically assessed the possibility of utilizing latent space representations for the estimation of time series averages. However, we also note that there is room for further improvements. For instance, to make our assessments feasible, i.e., in the context of computational resources, we have emphasized on using a supervised arrangement which we found to be straightforward in the sense of mimicking multiple alignments. However, our proposed approaches could easily be made unsupervised by combining them with concepts in deep clustering. However, further investigations are needed to address unforeseen challenges. Additionally, in recent times, task-specific neural network architectures are being proposed, for instance, the *InceptionTime*. We strongly believe that utilizing such architectures could significantly improve the quality of latent embedding that, in turn, improve the quality of time domain estimates. Finally, we would like to place a great emphasis on defining (agreeing) on a space where the estimated averages could get compared without bias. We also aim to direct our future effort in this regard.

Acknowledgements The authors would like to thank the creators and providers of the UCR datasets: Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Eamonn Keogh and Mustafa Baydogan. Moreover, the authors would also like to thank the university of Strasbourg for allowing us to use its HPC clusters (*Mesocentrer*). Last but not least, we would also like to thank Ouloufa Dorani and Sophia Nicée, and Dr. Esayas Gebreyouhannes for their roles in the continuation of the Ethio-France Ph.D. program under challenging circumstances. This work got conducted under the support of the French embassy for the African Union and Ethiopia and the former Ethiopian Ministry of Science and Higher Education (MOSHE).

References

Aghabozorgi, S., Shirkhorshidi, A. S. & Wah, T. Y. (2015), 'Time-series clustering a decade review', *Information Systems* **53**, 16–38.

Bagnall, A., Davis, L., Hills, J. & Lines, J. (2012), Transformation based ensembles for time series classification, *in* 'Proceedings of the 2012 SIAM international conference on data mining', Society for Industrial and Applied Mathematics, Anaheim,CA, USA, pp. 307–318.

Bagnall, A. & Lines, J. (2014), An experimental evaluation of nearest neighbour time series classification, Technical report, University of East Angelina, http://uk.arxiv.org/abs/1406.4757.

Bock, H.-H. (2008), 'Origins and extensions of the -means algorithm in cluster analysis.', *Journal lectronique d'Histoire des Probabilits et de la Statistique [electronic only]* **4**, 1–18.

Bulteau, L., Froese, V. & Niedermeier, R. (n.d.), 'Tight hardness results for consensus problems on circular strings and time series', *SIAM Journal on Discrete Mathematics* **34**(3), 1854–1883.

Chen, C. & Srivastava, A. (2021), Srvfregnet: Elastic function registration using deep neural networks, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops', IEEE Computer Society, New Orleans, Louisiana, USA, pp. 4462–4471.

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A. & Batista, G. (2015), 'The ucr time series classification archive'. www.cs.ucr.edu/~eamonn/time_series_data/.

Christian, S., Liu, W., Jia, Y., Pierre, S., Scott, R., Dragomir, A., Dumitru, E., Vincent, V. & Andrew, R. (2015), Going deeper with convolutions, *in* '2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE Computer Society, Boston, MA, USA, pp. 1–9.

Cuturi, M. & Blondel, M. (2017), Soft-dtw: a differentiable loss function for time-series, *in* 'Proceedings of the 34th International Conference on Machine Learning', JMLR.org, Sydney, NSW, Australia, pp. 894–903.

Debella, T. T., Shawel, B. S., Devanne, M., Weber, J., Woldegebreal, D. H., Pollin, S. & Forestier, G. (2022), Deep representation learning for cluster-level time series forecasting, *in* '8th International conference on Time Series and Forecasting (ITISE)', MDPI, Gran Canaria, Spain, pp. 1–11.

Demšar, J. (2006), 'Statistical comparisons of classifiers over multiple data sets', *Journal of Machine learning research* **7**, 1–30.

Der Maaten, L. v. & Hinton, G. (2008), 'Visualizing data using t-sne', *Journal of Machine Learning Research* **9**, 2579–2605.

Detlefsen, N. S., Freifeld, O. & Hauberg, S. (2018), Deep diffeomorphic transformer networks, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE Computer Society, Lake City, UT,USA, pp. 4403–4412.

Diederik P., K. & Max, W. (2014), Auto-encoding variational bayes, *in* '2nd International Conference on Learning Representations (ICLR 2014)', ICLR, Banff, AB, Canada, pp. 1–14.

Dong, G., Liao, G., Liu, H. & Kuang, G. (2018), 'A review of the autoencoder and its variants: A comparative perspective from target recognition in synthetic-aperture radar images', *IEEE Geoscience and Remote Sensing Magazine* **6**(3), 44–68.

Fawaz, H. I., Benjamin, L., Forestier, G., Charlotte, P., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P. A. & Petitjean, F. (2020), 'Inceptiontime: Finding alexnet for time series classification', *Data Mining and Knowledge Discovery* **34**, 19361962.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L. & Alain-Muller, P. (2019), 'Deep learning for time series classification: a review', *Data Mining and Knowledge Discovery* **33**(4), 917–963.

Gee, A. H., Garcia-Olano, D., Ghosh, J. & Paydarfar, D. (2019), 'Explaining deep classification of time-series data with learned prototypes', *CEUR workshop proceedings* **2429**, 15–22.

Glorot, X. & Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, *in* 'Proceedings of the 13th Conference on Artificial Intelligence and Statistics', PMLR, Chia Laguna Resort, Sardinia, Italy, pp. 249–256.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014), 'Generative adversarial networks', pp. 1–9.

Gupta, L., Molfese, D., Tammana, R. & Simos, P. (1996), 'Nonlinear alignment and averaging for estimating the evoked potential', *IEEE transactions on biomedical engineering* **43**(4), 348–356.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE Computer Society, Las Vegas, NV, USA, pp. 770–778.

Itakura, F. (1975), 'Minimum prediction residual principle applied to speech recognition', *IEEE Transactions on Acoustics, Speech, and Signal Processing* **23**(1), 67–72.

Iwana, B. K. & Uchida, S. (2021), 'An empirical survey of data augmentation for time series classification with neural networks', *PLOS ONE* **16**(7), 1–32.

J. Jain, B., Froese, V. & Schultz, D. (2019), 'An average-compress algorithm for the sample mean problem under dynamic time warping', *CoRR* **abs/1909.13541**, 1–15.
URL: *http://arxiv.org/abs/1909.13541*

J. Jain, B. & Schultz, D. (2018), 'Asymmetric learning vector quantization for efficient nearest neighbor classification in dynamic time warping spaces', *Pattern Recognition* **76**, 349–366.

Junyuan, X., Ross, G. & Ali, F. (2016), Unsupervised deep embedding for clustering analysis, *in* 'Proceedings of the 33rd International Conference on International Conference on Machine Learning', Vol. 48, p. 478487.

Kaiming, H., Xiangyu, Z., Shaoqing, R. & Jian, S. (2015), Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *in* '2015 IEEE International Conference on Computer Vision (ICCV)', IEEE Computer Society, Santiago, Chile, pp. 1026–1034.

Kowsar, Y., Moshtaghi, M., Velloso, E., Bezdek, J. C., Kulik, L. & Leckie, C. (2022), 'Shape-sphere: A metric space for analysing time series by their shape', *Information Sciences* **582**, 198–214.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in Neural Information Processing Systems', Neural Information Processing Systems Foundation, Inc. (NeurIPS), Lake Tahoe, Nevada, USA, pp. 1106–1114.

Lafabregue, B., Weber, J., Ganarski, P. & Forestier, G. (2021*a*), 'End-to-end deep representation learning for time series clustering: a comparative study', *Data Mining and Knowledge Discovery* pp. 1–53.

Lafabregue, B., Weber, J., Ganarski, P. & Forestier, G. (2021*b*), 'End-to-end deep representation learning for time series clustering: a comparative study', *Data Mining and Knowledge Discovery* **36**, 29–81.

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

Lin, J. & Li, Y. (2009), Finding structural similarity in time series data using bag-of-patterns representation, *in* 'International conference on scientific and statistical database management', Springer, New Orleans, LA, USA, pp. 461–477.

Lines, J. (2015), Time Series classification through transformation and ensembles, PhD thesis, School of Electrical and Computer Engineering, University of East Anglia.

Niennattrakul, V. & Ratanamahatana, C. A. (2009), Shape averaging under time warping, *in* '2009 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology', IEEE, Chonburi, Thailand, pp. 626–629.

Niennattrakul, V., Srisai, D. & Ratanamahatana, C. A. (2012), 'Shape-based template matching for time series data', *Knowledge-Based Systems* **26**, 1–8.

Ongwattanakul, S. & Srisai, D. (2009), Contrast enhanced dynamic time warping distance for time series shape averaging classification, *in* 'Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human', Association for Computing Machinery, p. 976981.

Paparrizos, J. & Gravano, L. (2015), K-shape: Efficient and accurate clustering of time series, *in* 'Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data', Association for Computing Machinery, Melbourne, Victoria, Australia, p. 18551870.

Petitjean, F. & Gançarski, P. (2012), 'Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment', *Theoretical Computer Science* **414**(1), 76–91.

Petitjean, F., Ketterlin, A. & Gançarski, P. (2011), 'A global averaging method for dynamic time warping, with applications to clustering', *Pattern Recognition* **44**(3), 678–693.

Ruiz, E. V., Casacuberta Nolla, F. & Segovia, H. R. (1985), 'Is the dtw distance really a metric? an algorithm reducing the number of dtw comparisons in isolated word recognition', *Speech Communication* **4**(4), 333–344.

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.

Salvador, S. & Chan, P. (2007), 'Toward accurate dynamic time warping in linear time and space.', *Intelligent Data Analysis* **11**(5), 561–580.

Schultz, D. & Jain, B. (2018), 'Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces', *Pattern Recognition* **74**, 340–358.

Shapira Weber, R. A., Eyal, M., Skafte, N., Shriki, O. & Freifeld, O. (2019*a*), Diffeomorphic temporal alignment nets, *in* 'Advances in Neural Information Processing Systems 32 (NeurIPS 2019)', Neural Information Processing Systems Foundation, Inc. (NeurIPS), Vancouver,Canada, pp. 6574–6585. **URL:** *http://papers.nips.cc/paper/8884-diffeomorphic-temporal-alignment-nets.pdf*

Shapira Weber, R. A., Eyal, M., Skafte, N., Shriki, O. & Freifeld, O. (2019*b*), Diffeomorphic temporal alignment nets: Supplementary material, *in* 'Advances in Neural Information Processing Systems 32 (NeurIPS 2019)', Neural Information Processing Systems Foundation, Inc. (NeurIPS), Vancouver,Canada, pp. 6574–6585.

Shawel, B. S., Debella, T. T., Tesfaye, G., Tefera, Y. Y. & Woldegebreal, D. H. (2020), Hybrid prediction model for mobile data traffic: A cluster-level approach, *in* '2020 International Joint Conference on Neural Networks (IJCNN)', IEEE, Glasgow,UK, pp. 1–8.

Simonyan, K. & Zisserman, A. (2015), Very deep convolutional networks for large-scale image recognition, *in* '3rd International Conference on Learning Representations (ICLR)', ICLR, San Diego, CA, USA, pp. 1–14.

Srisai, D. & Ratanamahatana, C. A. (2009), Efficient time series classification under template matching using time warping alignment, *in* '2009 Fourth International Conference on Computer Sciences and Convergence Information Technology', IEEE, Seoul, Korea, pp. 685–690.

Srivastava, A. & P. Klassen, E. (2016), *Functional and Shape Data Analysis*, Vol. 1, Springer, NY.

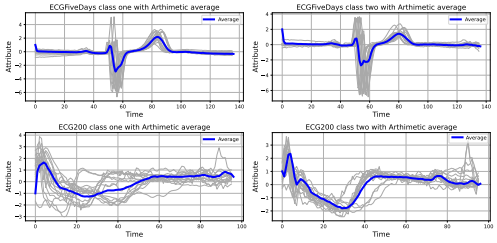
Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K. & Woods, E. (2020), 'Tslearn, a machine learning toolkit for time series data', *Journal of Machine Learning Research* **21**(118), 1–6.

Terefe, T., Devanne, M., Weber, J., Hailemariam, D. & Forestier, G. (2020), Time series averaging using multi-tasking autoencoder, *in* '2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)', IEEE Computer Society, Baltimore, MD, USA, pp. 1065–1072.

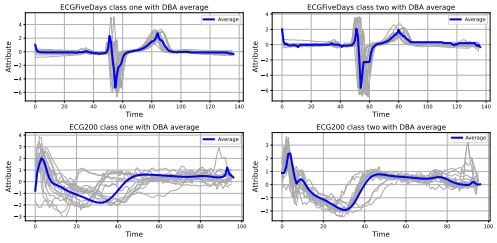
Wei, W. (2006), *Time Series Analysis: Univariate and Multivariate Methods*, 2nd edn, Pearson Addison Wesley, NY,USA.

Xie, J., Girshick, R. & Farhadi, A. (2016), Unsupervised deep embedding for clustering analysis, *in* 'Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML'16)', ICML, NY, USA, p. 478487.

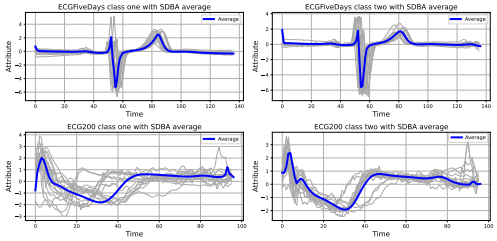
Ye, L. & Keogh, E. (2009), Time series shapelets: A new primitive for data mining, *in* 'Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', Association for Computing Machinery, Paris,France, p. 947956.



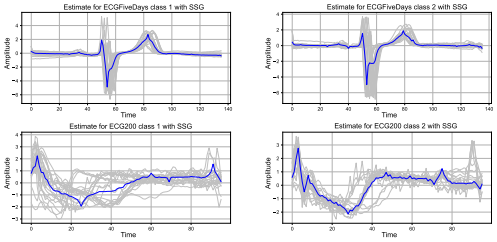
(a) Arithmetic



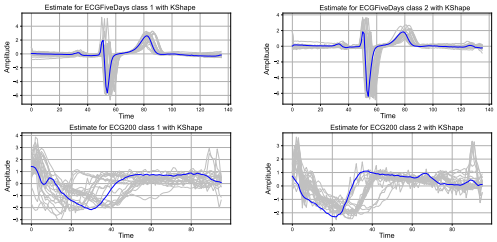
(b) DBA's estimates



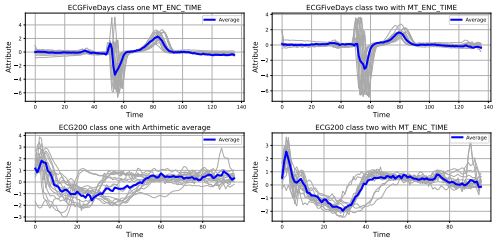
(c) SDBA's estimates



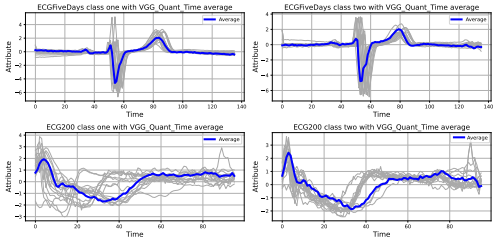
(d) SSG's estimates



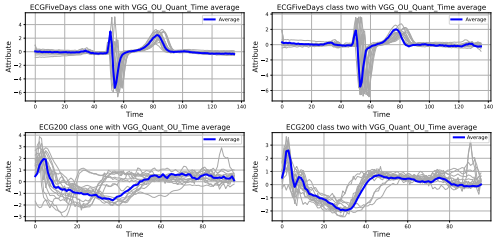
(e) Kshape's estimates



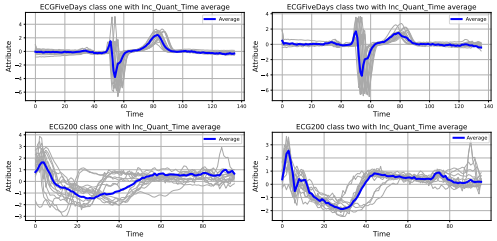
(f) MT_ENC_TIME's estimates



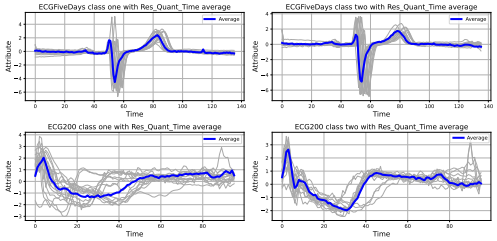
(g) VGG_Quant_Time's estimates



(h) $VGG\_OU\_Quant\_Time$'s estimates



(g) Inc_Quant_Time's estimates



(h) Res_Quant_Time's estimates

Fig. 15: Visual comparisons of estimated averages for the UCR archive's *ECG200* and *ECGFiveDays* datasets.