Time Series Averaging Using Multi-Tasking Autoencoder

Tsegamlak Terefe, Maxime Devanne, Jonathan Weber, Dereje Hailemariam, Germain Forestier

Time Series Averaging Using Multi-Tasking Autoencoder

Tsegamlak Terefe, Maxime Devanne, Jonathan Weber, Dereje Hailemariam, Germain Forestier
Emails:{tsegamlak-terefe.debella,maxime.devanne,jonathan.weber,germain.forestier}@uha.fr
{tsegamlak.terefe,dereje.hailemariam}@aait.edu.et

***Abstract*—The estimation of an optimal time series average has been studied for over three decades. The process is mainly challenging due to temporal distortion. Previous approaches mostly addressed this challenge by using alignment algorithms such as Dynamic Time Warping (DTW). However, the quadratic computational complexity of DTW and its inability to align more than two time series simultaneously complicate the estimation. In this paper, we follow a different path and state the averaging problem as a generative problem. To this end, we propose a multi-tasking convolutional autoencoder architecture to extract latent features of similarly labeled time series under the influence of temporal distortion. We then take the arithmetic mean of latent features as an estimate of the latent mean. Moreover, we project these estimations and investigate their performance in the time domain. We evaluated the proposed approach through one nearest centroid classification using 85 data sets obtained from the UCR archive. Experimental results show that, in the latent space, the proposed multi-tasking autoencoder achieves competitive accuracies as compared to the state-of-the-art, thus demonstrating that the learned latent space is suitable to compute time series averages. In addition, the time domain projection of latent space means provides superior results as compared to time domain arithmetic means.**

***Index Terms*—Time Series, Averaging, Autoencoder, Multi-tasking**

I. Introduction

A time series can be seen as a set of ordered observations whose attributes can be a vector or a scalar. Although the ordering of these observation is usually made through time, particularly in terms of some equally spaced time intervals, it can also be taken through other dimensions, such as space [1]. Today, such data sets are available from different sources like acoustic signals, financial indices, internet traffic, satellite images, etc. [1]. A more formal or technical definition of these data sets considered a time series as a temporally ordered sequence $X : \{x_1, x_2, ..., x_N\}$, defined over an attribute set λ, $x_i \in \lambda$ [2]. With this definition, a time series is considered to be univariate if $\forall x_i \in \lambda : x_i \in \mathbb{R}$. On the contrary, it is called a multivariate time series if $\forall x_i \in \lambda : x_i \in \mathbb{R}^m$.

Like their non temporal counterparts, time series are also analyzed in a supervised, unsupervised and semi supervised manner [3]–[5]. In practice, data mining techniques falling in these broad categories have to be tuned to meet data specific demands [6]. When it comes to time series, most of the techniques have to be tuned for temporal misalignment (distortion). This challenge is illustrated in Figure 1. In the Figure, we have selected the BettleFly data set from the University of California univariate time series repository (UCR) [7]. We have selected data sets belonging to class one. From Figure 1, we can first observe how the data sets present a similar per class shapes (features). Additionally, we can also notice how temporal distortion creates misalignment between class members.

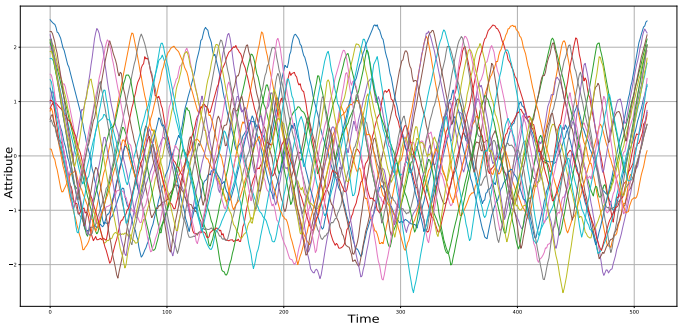


Fig. 1: Within class temporal distortion in BeetleFly data set obtained from the UCR.

Practically speaking, such distortions are apparent in temporal data sets for different reasons like difference in the sampling rate of measuring equipment or behaviour of measured entities [6], [8]. These distortion sources place a constraint on most temporal data mining techniques [6], [8]. For example, if we take the power consumption measurements of home appliances residing at different residents, we do not expect the measurements of similar appliances to be aligned perfectly. This is because, the utilization of the appliances are dependent on the schedule of the end user. Thus, if we are asked to identify appliances using templates (such as class averages), basic similarity measurement techniques, like Euclidean distance, are expected to perform poorly due to the misalignment [8]. In other problem domains, we might also be expected to generate representative averages for such measurements, e.g. K-means clustering, data indexing, etc [6], [8], [9]. In this regard, an Euclidean mean also performs poorly. This is better illustrated in Figure 2. In the figure, we have retaken the BeetleFly data set shown in Figure 1. A time series is defined for this data set by taking the boundaries of a binary (black & white) images of Beetles and Flies [8]. Even if, Beetles or Flies are expected to have a certain descriptive structures, the generated time series are not aligned perfectly due to a difference in the size of the insects, rotation of the images, etc. Thus, if we intend to define an average descriptor from several time series samples, their

 DOI: https://doi.org/10.1109/ICTAI50040.2020.00163

arithmetic mean estimation would look like the one shown in Figure 2. As highlighted by red boxes, we can observe that the arithmetic mean fails to preserve the overall shapes, resulting in a poor representative average.

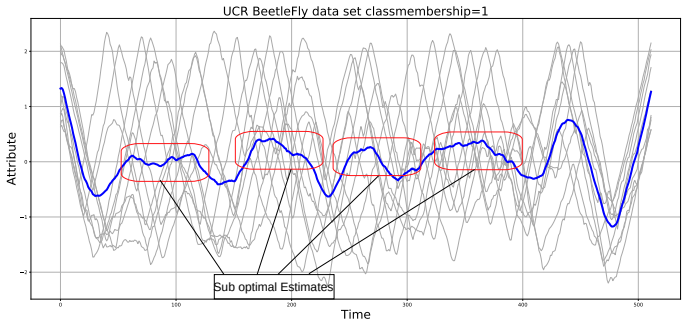


Fig. 2: Impact of temporal distortion on an arithmetic mean.

With this understanding, various temporal data mining techniques, including time series averaging, deploy alignment algorithms as a pre-processing step [10]–[13]. The algorithms minimize the phase discrepancy between pairs or groups of data sets. However, in most cases, the algorithms also present different challenges as seen from the task at hand [2], [9], [14]. For instance, the Dynamic Time Warping (DTW) is not suitable to align multiple time series [12]. Thus, it complicates the definition of an ensemble mean [15]. On top of that, it also provides a non smooth and non convex averaging cost function that is difficult to optimize [14]. With these observations, different data mining tasks propose mitigation techniques. For instance, domain transformation techniques have been utilized to speed up classification with sound accuracy [16], [17]. Furthermore, convolutional neural networks have also been intensively studied for time series classification [18]. Such CNN-base methods allowed to improve classification accuracy and overcome the high computational complexity associated with DTW based classification techniques. In the context of time series averaging, diffeomorphism have recently been proposed to mitigate the multiple alignment challenge in time series averaging [19]. However, in most proposed averaging techniques the possibility of utilizing neural networks have been overlooked due to complexities associated with DTW [10]–[12].

Practically speaking, we can state the averaging problem as: given a set of time series $X = \{X_1, X_2, ..., X_N\}$ where $X_i \in \mathbb{R}^m$, generate a series $z \in \mathbb{R}^n$ where $n \geq m$. We generate z in a way that minimizes:

$$D = \frac{1}{N} \sum_{i=1}^{N} d(z, X_i), \tag{1}$$

where, d is a distance function. In most cases, d it is taken as the DTW distance. To this end, currently available averaging heuristics transform the set X into an aligned representation. Then, they estimate an ensemble average by taking the arithmetic mean of the aligned ensemble. In this context, if DTW is utilized as an alignment procedure, (1) becomes a non smooth and non convex cost function [14]. Thus, it avoids the possibility of using DTW-based loss functions within neural networks for optimization.

In this paper, we follow a different path and investigate the latent space of an autoencoders to estimate (generate) z. To this end, we propose two sets of convolutional autoencoder architectures. An autoencoder performing reconstruction (encoding and decoding) and a multi-tasking convolutional autoencoder performing classification and reconstruction (decoding). In the latter proposal, classification is imposed as an additional task to force the encoder to learn per class separable and compact latent features. In our work, we place a higher emphasis on utilizing the latent features to estimate an optimal mean. We do this for one main reason:

- Given a set of time series (an ensemble) $X = \{X_1, X_2, X_3, \ldots, X_N\} | X_i \in \mathbb{R}^m$, a multiple alignment is expected to bring the members of an ensemble close to each other (obtains a compact transformation) in $\mathbb{R}^n$, where, $n \geq m$ [9], [19] . Thus, if we are able to extract compact and separable latent features using autoencoders, then we will be mimicking per class multiple alignment in $\mathbb{R}^l$ space, where $m > l$.

We intend to use this assumption and take the arithmetic average of the latent features as an optimal estimate of the mean. In addition, we also want to utilize the symmetric nature of an autoencoder to project the estimated latent mean to the time domain and observe its performance as compared to the state of the arts. To the best of our knowledge, generative models have never been investigated in such a way for time series averaging.

With this said, we have organized the rest of the paper as follows. In section II, we reviewed previous works addressing time series averaging. Following this, we described our approach in section III. The experimental results and their analysis is presented in section IV. We finalized the paper by drawing our conclusion and remaining future works in section V.

II. Related Work

As mentioned in the previous section, time series averaging techniques rely on some sort of alignment algorithms such as DTW or diffeomorphism [10]–[12], [19]. In reality, there are four well known DTW based averaging heuristics.

The first of these four proposals is the Non Linear Averaging and Alignment Filter (NLAAF) [10]. In NLAAF, members of an averaged ensemble are aligned in a pairwise manner prior to averaging. To this end, given N time series, NLAAF generate $N/2$ estimate in its first iteration. This is performed by taking the DTW aligned arithmetic mean of pairs [10]. This iterative process is continued until a single estimate is generated. However, the limitation with NLAAF is that, the algorithm assumes even number of ensemble members [11]. Furthermore, in NLAAF different estimations are generated for different pair selections. Moreover, the dimension of the estimated mean grows for each iteration due to averaging per DTW associated points [12].

To overcome these limitations, two averaging heuristics have been proposed: Prioritized Shape Averaging (PSA) and DTW Barycenter averaging (DBA) [11], [12]. PSA overcame the ordering dependency using hierarchical clustering and pairwise averaging [11]. However, PSA could not also overcome the limitation of growing mean dimension [12]. Hence, DBA proposed to align the members of the series to a template that is either selected from the ensemble or randomly initialized [12]. Additionally, DBA proposed to take the barycenter of DTW associated points [12]. This approach provided two advantages. First, DBA mimicked multiple alignment by aligning the series to a template. Hence, it provided improved results as compared to its predecessor. Second, the dimension of the estimated mean is fixed to the dimension of the template.Nevertheless, DBA is still expected to minimize a non smooth and non convex cost function [14].

To overcome this challenge, a differentiable version of DTW, called softDTW, have been proposed in [13]. The proposal smoothed the cost function, i.e., (1), hence improved the probability of DBA to identify global minimas, resulting in softDBA. In spite of this, DTW based averaging techniques still demanded re-execution when a single new data set becomes available [19].

To address this observation, continuous piecewise affine (CPA) velocity fields based on diffeomorphism was proposed in Diffeomorphic temporal alignment net (DTAN) [19]. DTAN estimated CPA velocity fields via a convolutional network from the input data sets. It then used the fields to conduct diffeomorphic transformation on the input that minimize the within group squared sum of similarly labeled time series. Thus, after transformation, similarly labeled time series became aligned and a per class arithmetic mean is presumed to be an optimal estimate [20]. In addition, beside addressing the memory less nature of DTW based approaches, DTAN also address the joint alignment problem by morphing groups of time series at a time. Thus, DTAN achieved superior results as compared to its predecessors [20].

III. Proposed Method

As discussed in previous section, most averaging techniques focused on addressing multiple alignment to obtain compact representations of original time series. On top of that, most of the techniques have not harnessed the potentials of neural networks or provided an end to end neural network solutions. To address the alignment challenge, we aim at obtaining compact representations of the original series in the latent space of a neural network learning the right task. For instance, if we consider a neural network built to conduct a classification task, we expect the network to extract class specific information (features) through its hidden layer [18], [21]. Thus, we expect the per class hidden layers (latent space) features to be compact and separable as compared to their original counterpart. In other words, we expect a neural network learning the right task to mimic temporal alignment in its latent space. Thus, in our work we try to identify a suitable architecture with a suitable learning task to transform the time series into a latent representation. Moreover, unlike most temporal transformation techniques [16], [17], we want our latent space transformation to have a meaningful interpretation in the time domain. This is because, we want to observe and evaluate the capability of the latent mean on preserving shapes and trends observed in the time domain. To this end, we have chosen to investigate autoencoders for two main reasons:

1) An autoencoder is by design used to obtain a compressed latent representation of a data set. Hence, it makes it a good candidate for feature learning.
2) With autoencoders, it is possible to project the learned latent features to their original representation.

With this understanding, we first investigate the feature extracting capability of a basic convolutional autoencoder, as shown at the top part of Figure 3. We then learn a per class discriminating features by making the architecture a multi-tasking autoencoder, as shown in Figure 3. For the multi-tasking, we select a classification task in addition to the reconstruction. This is expected to force the encoder learn a more compact per class latent representation. However, the latter architecture makes the feature learning process to be semi supervised, i.e., requiring class labels during training. Our approach accounts for this by being a trainable averaging technique. This is not true for most averaging heuristics, with the exception of DTAN [19].

A. Convolutional Autoencoders

*1) **Autoencoder models**:* An Autoencoder utilizes a symmetric architecture, i.e., encoder and decoder, to learn latent representations (features) of the input data sets and their re-projection respectively [22], [23]. To meet these objectives, a basic autoencoder aims at minimizing a reconstruction loss [23]. Given a time series $X = \{x_0, x_1, ..., x_N\}$ and their reconstructed version $R = \{r_0, r_1, ..., r_N\}$, a basic autoencoder aims at minimizing [22]:

$$L_{reconstruction} = \frac{1}{N}\sum_{i=0}^{N}(r_i - x_i)^2. \tag{2}$$

In practice, we have different versions of an autoencoder, each modifying the objective function to a specific task [23]. For instance, a variational autoencoder aims at learning latent features resembling a given probability distribution. Thus, it adds KL divergence loss in addition to the reconstruction loss [23]. In this paper, we are trying to investigate the capability of autoencoders on extracting compact features from the original distributions of the data sets. Thus, we have avoided utilizing advanced encoders and strictly focused on the basic cost function given in (2).

*2) **Convolutional Autoencoders**:* In addition to different loss functions, autoencoders also utilize different types of layers such as Convolution, LSTM and dense [18], [23]. A layer type influences the type of extracted latent features [21]. For this work, we have selected convolution layers for two main reasons:

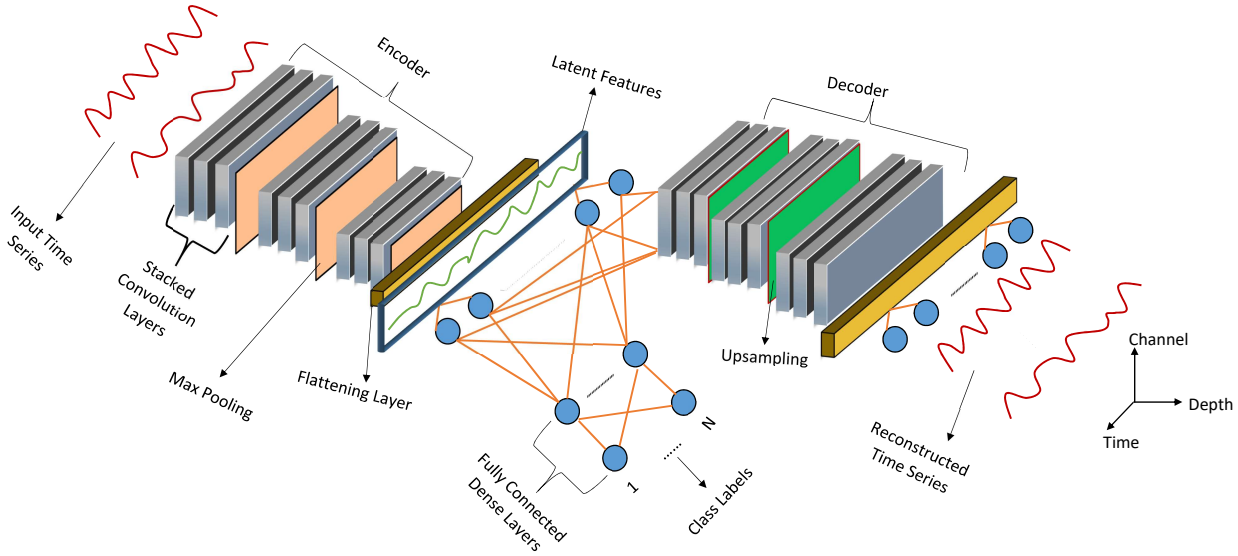


Fig. 3: Proposed multi-tasking autoencoder.

1) Convolution layers are computationally less demanding as compared to their dense and LSTM counterpart.
2) Convolution layers are able to learn local features and identify them globally.

A convolution layer extracts features by using the dot product of its kernel with the input. Hence, for a given input $X = \{x_0, x_1, x_2, ..., x_N\}$ and trainable kernel weights $W = \{w_0, w_1, ..., w_j\}$, the output entries of a convolutional layer are calculated as:

$$C(i) = \sum_{k=0}^{j} w_k x_{i+k} \tag{3}$$

In the convolution operation, we can assume the trainable weight are the impulse response of a linear filter. In linear systems, filters are designed to respond to a specific set of inputs. Similarly, a convolution layer kernel is responsive to a specific feature (shapes) [21]. Thus, we can consider each entries of the transformation $C(i)$ as a response of the kernel to the segmented input of the data sets. Practically, most convolution layers learn M such kernels in parallel. Thus, the output of a convolution layer with single stride and same padding is a $M \times N$ dictionary of responses, where N is the dimension of the original data set.

The complexity and descriptiveness of the learned features can be enhanced by sequentially stacking convolution layers [21], [24]. This will help to learn hierarchical shapes by systematically increasing the view of the filter [18], [21]. Moreover, beside the stacking of layers, most convolutional networks also refine the learned features with pooling. Pooling layers filter out noisy and redundant features by either selecting the most dominant (Max Pooling) or averaged (Average Pooling) features. Thus, a pooling operation is expected to reduce the dimension of the input data set by $\frac{1}{K}$, where K is the pooling kernel size. We can also further increase the complexity of the learned dictionary by integration of non linear activation functions [21], [24]. In this regard, most convolution layers utilize ReLU activation function [18], [21], [24]. The ReLU activation will truncate negative responses to zero [21], [24]. In reality, convolution layers were designed to process shapes and images. Thus, it would be counter intuitive to expect negative pixels and shapes. Furthermore, the ReLU activation is able to overcome the exploding and vanishing gradient problems while optimizing through back propagation [21]. Thus, most of the time, it insures convergence.

*3) **Proposed convolutional Autoencoder**:* With the mentioned technicalities, we can now consider the encoder of a convolutional autoencoder to be a non linear transformation function. The transformation learns the basis functions (basis features) of a given data set. Most of the time, basis functions have the capability of separating the phase information from the original data set. For instance, if we consider a time series $X = sin(t + 30), t = 1, \ldots, T$ as an example, it can be written as $sin(t)cos(30) + cos(t)sin(30)$. Hence, we can write X as $sin(t + 30) = [sin(t)\ cos(t)][cos(30)\ sin(30)]^T$. Furthermore, any variants of $sin(t)$ and $cos(t)$ can also be written in such manner, thus making $sin(t)$ and $cos(t)$ to be the basis dictionary. Generally, for an $M \times N$ basis dictionary D and a recombination factor $q \in \mathbb{R}^n$, a time series, $X \in \mathbb{R}^m$, spanned by the dictionary D is computed as,

$$X = Dq^T, \tag{4}$$

where the basis functions are arranged in column.

With this definition, we can now consider the outputs after the flattening layer of the encoder, shown in Figure 3, as a learned basis features (D). We can also assume the recombination factors (q) as the learned weights of the dense layer. At this point, it is feasible to think that if the inputs belong to similar classes or clusters, the convolution layers, the non linear activation and the pooling layers will pick up similar

features (response). Additionally, the combined transformations are expected to filter out minor phase distortion. If these conditions are met, one can expect the latent representations to be compact. Thus, in the latent space, we can assume an arithmetic mean to be an optimal estimation of the average.

Nevertheless, in most practical cases, multiple per class averages are needed. In such scenario, it would be unrealistic to expect a basic autoencoder to generate separable compact latent features. In previous proposals, heuristics utilize class information directly or indirectly. For instance, DBA will average each class separately, DTAN will use class labels to learn per class alignment [12], [19]. In our case, we have identified three possible paths:

1) For N classes, train N autoencoders
2) Force a single autoencoder to learn multiple per class dictionaries.
3) Integrate the class information with the help of multi-tasking.

We have avoided the first solution due to its computational cost. On the contrary, we have tested the second and third proposal. For the multi-tasking model, we propose an autoencoder performing classification and reconstruction with a shared encoder (hard multi-tasking), as shown in Figure 3.

B. Multi-tasking Autoencoder

The quality of the learned latent representations are dependent on the objective function. In basic autoencoders, the objective function will force the encoder to learn features minimizing reconstruction. Thus, in the case of multi class inputs, we have no control over the separability of the per class features. This will increase the chances of overlapping latent means which are poorly projected by the decoder. To overcome this limitation we can force the autoencoder to perform an additional learning task. In the context of obtaining a compact latent representation, a more suitable choice would be classification. This is true for two reasons:

1) The classifier will force the encoder to learn the most per class discriminating basis features [21].
2) The classifier will also force a compact recombination of the discriminating basis features generated after the flattening layer of the encoder. This is because, in the proposed architecture, the classifier is attached after the dense layer. Hence, in order to achieve higher accuracy it needs to recombine the learned basis features in a non overlapping (separable) manner.

In addition, multi-tasking is also expected to learn better features with limited training data sets. Generally, the multi-tasking autoencoder approach modifies objective function given in (2) and becomes [25]:

$$L_{multi}(x, r, h, P) = \frac{1}{N}\sum_{i=0}^{N}(r_i - x_i)^2 - \sum_{c=0}^{C} h_{o,c} \log p_{o,c}, \quad (5)$$

where r and x are the reconstructed and input time series, and h and p are the categorical representation and the softmax activation values for each C catagories, respectively.

IV. Experimental Evaluation

In this section, we have provided a description of the proposed autoencoder architectures and our training strategy.

A. Proposed Autoencoder Architectures

For the encoder and decoder portion of the network, we have followed an architecture resembling the one proposed by the visual geometry group (VGG), i.e., VGG16, as shown on the top part of Figure 3. Thus, for the encoder we have used nine convolutional layers. The nine layers are divided among three groups. Each group is assigned with a filter size of 128, 64 and 32. Furthermore, between each group we have used in total three max pooling layers with a kernel size of three. The same architecture is repeated for the decoder, where the max pooling is replaced with an upsampling layer having a kernel size of three. However, we have not used a third upsampling layer at the end of the decoder. Additionally, we have set the dimension D_l of the latent space to $D/4$, where, D is the dimension of a time series. Finally, for the multi-tasking autoencoder, the classifier is constructed from three fully connected dense layers with a node size of $D_l/2$, $D_l/4$ and the number of class labels N_c.

B. Evaluation Procedures and Data Sets

In practice, the quality of an estimated average is measured by within group squared sum (WGSS), i.e., a per class Fréchet sum given in (1) [12]. An alternative way of testing WGSS would be one nearest centroid classification. Indeed, if a mean achieves a high one nearest centroid classification accuracy, it is minimizing its WGSS with a high probability [20]. Moreover, one nearest centroid classification gives a practical implication in relation to application domains [9]. To this end, we have tested our proposal using one nearest centroid classification on 85 data sets obtained from the UCR archive [7].

We have conducted the classification task in both the latent space and time domain. For the latent space classification, we used the arithmetic mean of latent features extracted from the training data sets. We then projected the test data sets to the latent space and used Euclidean distance as a similarity measure. This is done to evaluate the separability and compactness of the latent features. On the contrary, since time domain is still impacted by distortion, the quality of the projected mean is evaluated with DTW distance. Finally, we have extracted the time domain results for the other averaging techniques from [20]. In [20], the classification results for DBA, softDBA were computed using DTW distance for 84 data sets. To conduct the evaluation for DTAN [20], authors first trained their diffeomorphic network using the train data sets. Then, they used the trained network to project test data sets so that a multiple alignment of the data sets is achieved. Thus, in the aligned space, the one nearest centroid classification was conducted by taking an arithmetic mean that is estimated from the train set and euclidean distance. However, the paper avoided estimating the mean for the data "StarlightCurves" for unmentioned reasons. Nevertheless, we have run our experiments for 85 data sets for future

comparison. Moreover, [20] also computed the classification accuracy for the arithmetic mean using euclidean distance, which is not fair. This is because, an arithmetic mean is highly impacted by temporal distortion. Moreover, the results for DBA, softDBA are evaluated using DTW distance which is expected to give superior results [6], [8]. Thus, we have recomputed the results for the arithmetic mean using DTW distance. In our implementations, we have used Tslearn's implementations of DTW, DBA and SDBA [26].

We have conducted more than 580 experiments. The experimentation code was implemented in Keras with Tensorflow backend. Our implementation of the autoencoder and the full experimental results are available online[1]. For the experiments, we have used 80% of the training data for training and 20% for validation. Moreover, we have trained the plain autoencoder for 2,500 epochs with no L1 or L2 regularization. On the contrary, we have trained the multi-tasking autoencoder with five different L2 regularization setups as shown in Table I. For the first regularization setup, we have trained the network with zero L2 regularization and 600 epochs. We have done this to reduce computational time and also avoid over-fitting. For the remaining four setups, we have trained the network for 2,500 epochs so that the network converges. We then have used the best results for analysis. Finally, for all training, we have set the learning rate to 10^{-4}.

TABLE I: L2 Regularization schemes used while training the multi tasking autoencoder.

No.	Layer	L2 Regularizations
1.	Encoder	$0, 10^{-4}, 10^{-3}, 10^{-3}, 10^{-2}$
2.	Decoder	$0, 10^{-4}, 10^{-3}, 10^{-3}, 10^{-2}$
3.	Classifier	$0, 10^{-3}, 10^{-3}, 10^{-2}, 10^{-2}$

C. Experimental Results

*1) **Numerical and Graphical Analysis**:* For the multi-tasking autoencoder, we have selected outcomes with higher latent space classification accuracy. Such accuracy indicates separable and compact latent features. As we can see in Table II, in the latent space, the multi-tasking autoencoder achieved wins without ties (clean wins) on 32.14% of the data sets (MT.Enc.Lat). On the other hand, the time domain projected means were able to achieve clean wins on 2.38% of the data sets (MT.Enc.Time). On the contrary, the basic autoencoder achieved a clean win on 2.38% of the latent space classifications (Enc.Lat). The time domain projection of the means won on 1.19% of the data sets (Enc.Time) and behaved as an arithmetic mean, as shown in Figure 7. Relatively, DTAN, DBA, SDBA, Arithmetic mean are able to achieve clean wins on 35.71%, 2.38%, 19.05% and 1.19% of the data sets in the time domain, respectively. In general, in the latent space, the classification results depicts the multi-tasking autoencoder is able to extract compact representations and achieve comparable results to the state of the art (DTAN). The

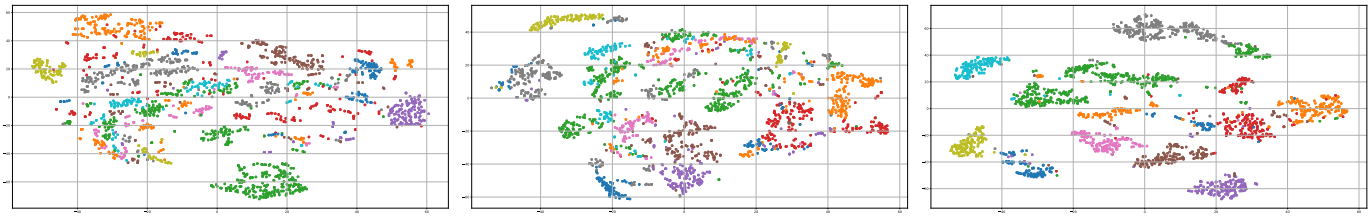
[1]https://github.com/tsegaterefe/Time-Series-Averaging-Using-Multi-Tasking-Autoencoder

TABLE II: Summary of one nearest centroid classification wins and ties.

No.	Avg. Method	Wins	Ties
1.	MT.Enc.Lat	27	1
2.	MT.Enc.Time	2	1
3.	Enc.Lat	2	0
4.	Enc. Time	1	0
5.	DTAN	30	4
6.	DBA	2	1
7.	SDBA	16	1
8.	Arth.Mean	1	0

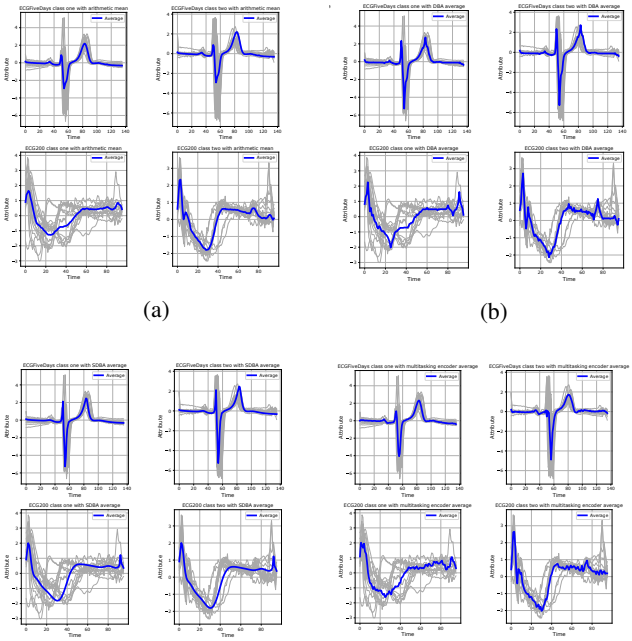
results also show that the basic autoencoder fails to capture a separable per class features. This is better demonstrated in Figure 4. In the figure, we have made the t-SNE projection of the FaceUCR data sets having 14 class labels. Figure 4a shows the projection of the time domain test data set, whereas Figure 4b and Figure 4c show the latent space projection of the test data set using the basic and multi-tasking autoencoders, respectively. We have only used class labels for the sake of depiction. The projections show that the multi-tasking autoencoder is learning a separable and compact per class latent features. Hence, with these features, we are able to compute non overlapping latent arithmetic means with superior performance in both the latent space and time domain. Furthermore, to give a feel of what the time domain projections look like, we have presented examples of estimated means using an arithmetic mean, DBA, SDBA and our multi-tasking autoencoder in Figure 5. However, we have not made the graphical comparison with DTAN since we were unable to run their implementation due to code dependencies. We have selected the ECG200 (top) and ECGFiveDays (bottom) data sets from the UCR repository. For the multi-tasking-based estimation, we have used a trained network to project the train data set into the latent space. We then have taken the arithmetic mean of the latent features as an estimate of the mean. Following this, we have projected the estimation to the time domain using the decoder. On the contrary, the arithmetic mean of the train time domain data set is taken for the arithmetic mean plots. Figure 5d shows the efficiency of the multi-tasking autoencoder at capturing peak values, trends and shapes observed in the original data set as compared to a time domain arithmetic mean shown in Figure 5a. To compute the DBA and SDBA means we have executed their respective algorithm for 100 epochs. Figure 5b and Figure 5c shows how DBA and SDBA better capture peak values and overall shape trends than the multi-tasking autoencoder on time domain.

*2) **Hypothesis Test**:* In addition to latent space graphical representations and win and tie analysis, we have conducted Wilcoxon signed rank hypothesises test [27]. The test is summarized in the critical diagram shown in Figure 7. The diagram shows that the latent space classification using the multi-tasking autoencoder (MT.Enc.Lat) is a similar hypothesis as DTAN. Thus, in the latent space we are indeed mimicking alignment and an arithmetic mean is a good representative



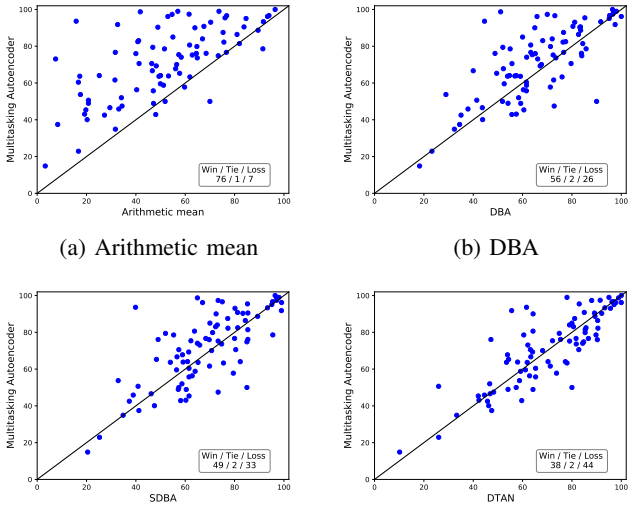
(a) Time series features (b) Autoencoder latent features (c) Multi-tasking autoencoder latent features

Fig. 4: t-SNE projections of the FacesUCR data set. Input features are time series (a), encoded time series using a simple autoencoder (b) and encoded time series using our multi-tasking autoencoder. (c)



(a) (b) (c) (d)

Fig. 5: Comparison of four estimated means using an arithmetic mean (a), DBA (b), SDBA (c) and our multi-tasking autoencoder (d).



(a) Arithmetic mean (b) DBA

(c) SDBA (d) DTAN

Fig. 6: Accuracy comparison between our multi-tasking autoencoder and an arithmetic mean (a), DBA (b), SDBA (c) and DTAN (d).

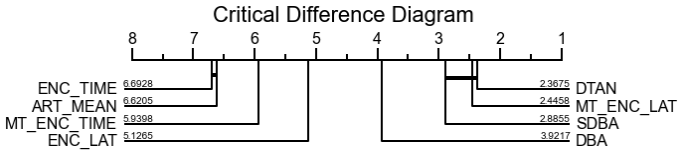


Fig. 7: Comparison of different averaging methods.

of the features. Moreover, it also shows that the projected latent mean (MT.Enc.Time) of the multi-tasking autoencoder outperforms an arithmetic mean. However, since the basic autoencoder (Enc.Lat) is incapable of providing a separable and compact latent representation, as shown in Figure 4, its performance is inferior as compared to the other averaging heuristics. Nevertheless, it still outperforms an arithmetic mean in the latent space. However, its time domain projection (Enc.TIME) behaves as an arithmetic mean.

*3) **Impact of regularization**:* We have also analyzed the impact of the regularization on winning a classification task. To this end, we have observed which regularization setup wins a classification task for a given data set. In this regard,in the latent space, the multi-tasking autoencoder with zero L2 regularization running for 600 epoch wins on 27 (32.53%) data sets as compared to the other regularization setups. The 2^{nd} to 5^{th} regularization setups given in Table I wins on 16 (19.28%), 19 (22.89%), 15 (18.07%) and 6 (7.23%) data sets respectively. In general, if the intention is to win latent space classification, we advise to initially test zero L2 regularization and move to the third regularization setup. However, in terms of achieving better classification accuracy on the overall data set, the second L2 regularization setup given in Table I performs better.

In general, the latent space performance of the multi-tasking autoencoder achieved its desired objective of mimicking multiple alignment. However, the performance of the projected means are inferior as compared to the state of the art. This can be explained by the non continuous nature of the latent space.

This is because, for an autoencoder performing reconstruction, the encoder is expected to map each input time series to a specific latent space representation. To this end, the latent space becomes discrete. Thus, if we are intending to use the basic autoencoder as a generative model, then the decoder is forced to reconstruct an unseen latent features as in the case of the arithmetic means. Hence, the resulting projections will not be optimal in most cases. Moreover, the classifier and the autoencoder are competing for the encoder's attention, hence there is a high probability that a feature learned for one task to be a non natural response for the other. Hence, further investigation is needed on the matter.

V. Conclusion

In this paper, we have investigated the use of a generative neural model for time series averaging. To meet this objective, we proposed to mimic multiple alignment in the latent space of a basic and multi-tasking convolutional autoencoders. Our latent space outcomes show that an autoencoder optimizing the right cost function, i.e., the multi-tasking autoencoder, can indeed achieve the desired objective. Moreover, the projected latent means give superior results as compared to a time domain arithmetic mean. However, their performance is inferior to the latent space due to the fact that the decoder is only able to observe limited examples of the latent space. To overcome this limitation, we intend to investigate continuous latent space by leveraging variational autoencoders with Gaussian mixture models. Additionally, we also intend to give a more extensive investigation on the impact of filter and pooling kernel sizes, classifier architecture and network size on the quality of the projected means.

References

[1] W. Wei, *Time Series Analysis: Univariate and Multivariate Methods*. Pearson Addison Wesley, 2006.

[2] B. J. Jain and D. Schultz, "On the existence of a sample mean in dynamic time warping spaces," 2016.

[3] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clustering–a decade review," *Information Systems*, vol. 53, pp. 16–38, 2015.

[4] A. Bagnall and J. Lines, "An experimental evaluation of nearest neighbour time series classification," *arXiv preprint arXiv:1406.4757*, 2014.

[5] Z. You, *Weak-supervison Time-series Analysis*. PhD thesis, Sch. of Elec. and Comp. Eng., Oregon State University, Mar. 2019.

[6] A. Bagnall, L. Davis, J. Hills, and J. Lines, "Transformation based ensembles for time series classification," in *Proceedings of the 2012 SIAM international conference on data mining*, pp. 307–318, SIAM, 2012.

[7] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

[10] L. Gupta, D. L. Molfese, R. Tammana, and P. G. Simos, "Nonlinear alignment and averaging for estimating the evoked potential," *IEEE transactions on biomedical engineering*, vol. 43, no. 4, pp. 348–356, 1996.

[8] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, 2017.

[9] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[11] V. Niennattrakul and C. A. Ratanamahatana, "Shape averaging under time warping," in *2009 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, vol. 2, pp. 626–629, IEEE, 2009.

[12] F. Petitjean and P. Gançarski, "Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment," *Theoretical Computer Science*, vol. 414, no. 1, pp. 76–91, 2012.

[13] M. Cuturi and M. Blondel, "Soft-dtw: a differentiable loss function for time-series," in *Proceedings of the 34th International Conference on Machine Learning*, (Sydney, NSW, Australia), pp. 894–903, Aug. 2017.

[14] D. Schultz and B. Jain, "Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces," *Pattern Recognition*, vol. 74, pp. 340–358, 2018.

[15] M. Brill, T. Fluschnik, V. Froese, B. Jain, R. Niedermeier, and D. Schultz, "Exact mean computation in dynamic time warping spaces," *Data Mining and Knowledge Discovery*, vol. 33, no. 1, pp. 252–291, 2019.

[16] J. Lin and Y. Li, "Finding structural similarity in time series data using bag-of-patterns representation," in *International conference on scientific and statistical database management*, pp. 461–477, Springer, 2009.

[17] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[18] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[19] R. A. Shapira Weber, M. Eyal, N. Skafte, O. Shriki, and O. Freifeld, "Diffeomorphic temporal alignment nets," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 6574–6585, Curran Associates, Inc., 2019.

[20] R. A. Shapira Weber, M. Eyal, N. Skafte, O. Shriki, and O. Freifeld, "Diffeomorphic temporal alignment nets: Supplemantary material," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 6574–6585, Curran Associates, Inc., 2019.

[21] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, and V. Zocca, *Python Deep Learning: Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, 2nd Edition*. Packt Publishing, 2019.

[22] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49, 2012.

[23] G. Dong, G. Liao, H. Liu, and G. Kuang, "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*, vol. 6, no. 3, pp. 44–68, 2018.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[25] Y. Zhang and Q. Yang, "An overview of multi-task learning," *National Science Review*, vol. 5, no. 1, pp. 30–43, 2018.

[26] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, "Tslearn, a machine learning toolkit for time series data," *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020.

[27] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.