Towards deep learning for simultaneous time series classification and motif discovery

Name(s) omitted for blind review

***Abstract*—Time Series Classification (TSC) has attracted a growing interest over the past years. Distance-based approaches such as the 1-Nearest-Neighbor (1-NN) classifier with Dynamic Time Warping (DTW) or constructing Time Series Ensembles (TSE) merging several distance measures remain the leading methods for TSC. However, the computational complexity of TSE makes them intractable for larger real-life datasets. Up to now, machine learning approaches applied to TSC have been mostly unsuccessful because of their inability to handle the *curse of dimensionnality*. In this paper, we provide the first evidence that carefully trained models of *deep learning* can successfully be applied to univariate time series classification. By benchmarking an extensive set of simple models and analyzing several of their properties, we show that these methods can provide a new leap in both *accuracy, computational efficiency* and also *interpretability*. By comparing various parameter settings and network structures, we demonstrate that the architecture of these networks is the most critical component to their overall success. We show that these approaches are the first to *significantly* and *critically* outperform all previous state-of-art methods on both the complete UCR archive and an overall collection of 75 time series datasets. Furthermore, these enhanced accuracies are obtained for only a small fraction of the computational cost. We also show that deep networks implicitly perform a simultaneous assessment of time series motifs discovery, even across a large number of heterogeneous datasets. We hope that this paper open new avenues of research for deep learning applied to time series data mining.**

I. Introduction

Time Series Classification (TSC) is a research stream that has witnessed a flourishing interest in a wide variety of scientific topics such as speaker identification [12], optical character recognition [25] or cardiology [36]. Given sets of labeled time series belonging to different classes, the TSC task aims at training a classifier in order to label new time series of unknown class. An early approach to TSC was presented in [2] and since then, a variety of approaches have been applied to this topic ranging from 1-NN [35] to more complex statistical models such as Auto-Regressive Moving Average (ARMA) or Hidden Markov Models (HMM) [37]. In the context of TSC, it has been shown that the 1-NN classifier is extremely hard to overpower [35]. Several studies subsequently confirmed that the DTW-based 1-NN classification remained the best performing TSC method for almost a decade [9], [11], [26]. This comes from the fact that time series are inherently high-dimensional data. Therefore, it is very difficult to devise statistical methods that can avoid this *curse of dimensionality*. Recently, the idea of building *time series ensembles* (TSE) of classifiers based on multiple distance measures has been proposed [24], where the classification decision is based on a weighted sum of these. This method was shown to provide a significant improvement in accuracy over the 1-NN with DTW classifier.

Several drawbacks can be outlined from the literature in TSC research. First, despite recurrent efforts in the definition of novel distance measures, most seem to be converging towards pervasive performance ceilings which stands under satisfactory accuracies [9]. Hence, most distance measures introduced in the past years are still outperformed by the 30-years old DTW approach [5]. This can be related to the sub-optimality and

limitations of hand-designing distance measures. Indeed, manually crafted distances are inherently limited, if not only by the knowledge of their designer, but also in their restricted ability to generalize and adapt to different problems and datasets [14]. Furthermore, current classification systems are restricted to shallow 2-layers (compare patterns and match) architectures which are inherently limited. The idea of TSE provides a gain in accuracy as it adds a layer of processing (merging various distances) prior to class selection. This seems to confirm the general idea that depth of the classifying architecture might be crucial to its success. However, it requires the computation a set of (*quadratic* or even *cubic*) distance measures against the complete training dataset, which makes it impracticable for most real-life applications both in terms of computational complexity and memory requirements.

Recently, the field of *deep learning* [3] has witnessed a flourishing interest amongst the machine learning community. Its goal is to train connexionist architectures analoguous to the well-known neural networks. However, the typical feed-forward perceptrons are usually bound to small depth (number of layers) because of the *gradient diffusion* problem that lead to inefficient learning in deeper networks. The recent breakthrough of *greedy layer-wise pre-training* [4] allows to train each layer of the network independently and in an unsupervised manner. This leads to various types of architectures such as *deep belief networks* that can display an extensive number of layers [13], similar to those found in the brain for perceptual task such as visual or auditory processing. This approach appears as a valid candidate to learn AI models that could bypass the *curse of dimensionnality*, but also to learn automatically higher-level abstractions fit to various problems of time series data mining as exhibited in Figure 1.

In this paper, we provide the first evidence that ML approaches through deep learning can provide a new leap in time series classification and outperform existing approaches both in *accuracy*, *computational efficiency* and *interpretability*. Our goal was to rely solely on the simplest deep learning models

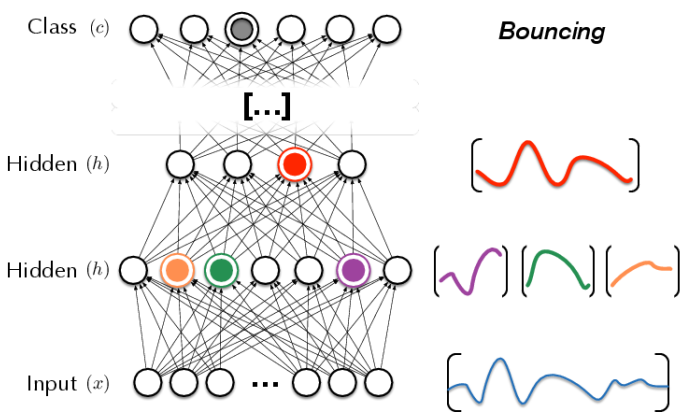

Fig. 1. The adequacy of *deep learning* models to perform simultaneous time series classification and motif discovery.

and show that by carefully investigating and adapting their architecture, properties and parameters to the intrinsic characteristics of time series, we could obtain extremely powerful time series data mining methods. We investigated the major deep learning algorithms [3] and compared the probabilistic and computational view over these connexionnist architectures. For all of the benchmarked models, we analyzed different variants, types of computation units and influence of various learning parameters on the success of these approaches. Finally, we extensively studied the impact of various architecture shapes (connection patterns) and depth of the networks. In this, we show that the architecture and its shape are of paramount importance to the success of learning.

(?? MAYBE TRASH IF CLUSTER TIME UNSUFFICIENT ??)

We also introduce two mecanisms specifically targeted at time series processing. First, the *manifold densification* aims at balancing the number of examples from each dataset in the pre-training step while accounting for the notion of temporal warping. The idea is that this densification avoids the algorithm to focus on only the most represented sets. **+ Warp in DAE**

(?? MAYBE TRASH IF CLUSTER TIME UNSUFFICIENT ??)

We show that carefully selected deep learning approaches provide a strongly better accuracy and only for a small

fraction of the computational complexity at testing time. We further show that these classifiers outperform all competing approaches and state-of-art results *significantly* and *critically* on the complete 46 UCR time series datasets [20] and the additional 29 datasets used for evaluating time series ensembles [24]. Furthermore, relying on deep networks provides several paramount advantages over existing approaches. First, regarding computational complexity, the computations required for classification consist in a single forward pass through the network. Therefore, it is independent of the size of the training set and can also easily be applied in real-time or streaming situations. Finally, the computational units in deep learning provide an extremely more significant interpretability by providing a direct access to automatically learned time series motifs. Hence, we provide evidence that this approach can simultaneously lead to powerful time series motif discovery mechanisms without any additionnal computations. By feeding various unifromly warped versions of the series, we can simultaneously discover (through learning) and detect (through activations) of the motifs with obtained correlations implied by each computation unit.

(??? HERE DEPENDS ON CLUSTER RESULTS ???)

* We show that (? archi ¿ parameters ?), random (? Maybe not ?) and DAE vs. RBM

(??? HERE DEPENDS ON CLUSTER RESULTS ???)

The remainder of this paper is organized as follows. We start by outlining previous work and necessary background in time series classification and deep learning (Section II). Then, we detail our evaluation methodology and corresponding measures of statistical significance (Section III). We provide an in-depth evaluation of our results over all datasets (Section IV) and separate the analysis of the results by first comparing various models, architectures and parameters (Section IV-A), then comparing our results to the state-of-art methods (Section IV-B) and also discussing their differences in complexity and interpretability. Then, we provide our results on time series motif mining based on the same approach (Section IV-C). Finally, we provide our conclusions and directions of future work (Section VI).

II. State of the art

A time series T is a collection of values obtained from sequential measurements over time. It can be defined as an ordered sequence of n real-valued variables

$$T = (t_1, \ldots, t_n), t_i \in \mathbb{R}$$

A time series is often related to a process observed at uniformly spaced *time instants* at a given *sampling rate*. The critical aspect for TSC lies in defining an adequate *dissimilarity measure* $\mathcal{D}(S, Q)$ between series.

Definition 1. A *dissimilarity measure* $\mathcal{D}(S, Q)$ between series S and Q is a function taking two series as inputs and returning a non-negative real value. If the measure also provides the properties of *symmetry* ($\mathcal{D}(S, Q) = \mathcal{D}(S, Q)$) and *subadditivity* ($\mathcal{D}(Q, T) \leq \mathcal{D}(Q, S) + \mathcal{D}(S, T)$), it is said to be a *metric*.

An ideal time series distance measure should be robust to distortions such as amplitude, scaling, warping, noise and outliers. Hence, a similarity measure should be consistent with human intuition and provide recognition of perceptually similar objects, even though they are not mathematically identical. Many authors have reported various transformation invariances required for similarity measures. We recently proposed four properties expressing robustness for *scaling* (amplitude modifications), *warping* (temporal modifications), *noise* and *outliers*. We direct interested readers to the formal definition of these robustness properties [10]. The Euclidean distance is unable to reach such a level of abstraction and numerous authors have pointed out the pitfalls of using ℓ^p distances [9], [17]. Hence a wide variety of distance measures have been developed deriving from Dynamic Time Warping (DTW) [5] such as Weighted DTW (WDTW) [16], Optimal Subsequence Bijection (OSB) [21] or from the edit (Leven-

shtein) distance such as the Longest Common SubSequence (LCSS) [7], Edit with Real Penalty (ERP) [6]. Other types of time series distances such as a Fast Fourier Transform (FFT), AutoCorrelation Function (ACF) [1] and Compression-based Dissimilarity Measure (CDM) [18] have also been proposed for TSC. As a complete review of time series distance measures is outside the scope of this article, we direct interested readers to [9], [10] for more information.

A. Time series classification

The goal of TSC is to assign labels to a set of time series of unknown classes, given a training set of series with known classes. The idea is first to learn what are the distinctive *features* that distinguish classes apart from each other. Then, when an unlabeled set is input to the system, it can try to automatically assign each series to one class from the set of predefined classes.

An early approach to time series classification was proposed in [2]. However, it was based on simple trends whose results are therefore hard to apply to more complex data. A piecewise representation was later proposed in [19] which provides robustness to noise but also allows relevance feedback. To overcome the obstacle of high dimensionality, Jeng and Huang [15] used Singular Value Decomposition to select essential frequencies. However, it implies higher computational costs. Several other techniques have been introduced, such as Hidden Markov Models (HMM) [37], which can be enhanced by using discriminative HMMs in order to maximize inter-classes differences [23]. Several machine-learning techniques have also been introduced such as Neural Networks or Bayesian classification [1]. However, all of these proposals have been repeatedly shown to be overpowered by a simple 1NN-DTW classifier [35] in terms of accuracy (while computing speed is significantly affected by repeated DTW computations). Recently, the idea of building time series *ensembles* has been proposed [24]. This approach computes a whole set of distance measures and performs the classification decision based on a weighted sum of these measures. This method was shown to outperform previous state-of-art approaches and will be used as a baseline for comparison in our experiments. However, it should be noted that this method implies a very heavy computational cost, as it requires to compute several (quadratic or even cubic) distance measures against the *complete* training dataset (which might also be kept in live memory). Therefore, even though this approach provides an increased classification accuracy, it appears impracticable for most real-life applications.

As listed in the previous section, a variety of distance measures have been devised but all seem to converge towards performance ceilings. The only advance in TSC accuracy comes from merging several distance measures which is computationally inefficient. This points out to a limitation in hand-crafted distance measures. Finally, the apparent inadequacy of machine learning approaches for TSC problems stems from their inability to handle the curse of dimensionality. All these problems are directly addressed by the recent advances in deep learning, which is yet to be applied adequatly to univariate TSC.

B. Deep learning

The aim of deep learning is to construct neural networks (similar to the well-known multi-layer perceptron) but with a deeper architecture (higher number of layers). The general idea is that with multiple hidden layers, each provides increasingly higher-level and complex features over the input by uncovering correlations through non-linear transformations of the previous layer. Although this idea of depth was long-going in the neural network community, training seemed infeasible because of the gradient diffusion problem. However, the recent breakthrough of *unsupervised layer-wise pre-training* [4] allows to learn each layer at a time, by training each layer to minimize the reconstruction of the hidden representation output by the previous layer. Furthermore, this also provides a way to train the networks in a purely unsupervised manner.

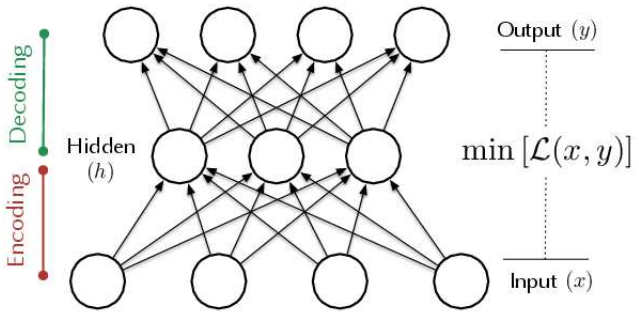

Fig. 2. An auto-encoder is composed of an encoding and a decoding layer. The aim is to find a way to reconstruct the input from a reduced representation, thus exploiting statistical correlations inside the input.

Overall, two interpretations can be given to these connexionnist architectures. First, the probabilistic view lead to models driven by probabilistic graphical models, interpreting the hidden units as latent random variables. Second, the computational view following neural networks lead to model constructed through computation graphs, where hidden units are considered as computational nodes. These two views are not entirely dichotomic as their similarities seem to outweigh their differences and can in fact become almost equivalent under certain assumptions [33].

1) Auto-encoders: The Auto-Encoder (AE) was first introduced as a dimensionnality reduction technique [30], where an encoder provides a reduced (compressed) representation of the input, while a decoder allows to reconstruct the orginal input from this encoded representation. The main idea is that if you are able to deconstruct an object in fewer components and then reconstruct it, you are grasping some aspects of its inner structure. Hence, it tries to exploit statistical correlations inside the data to find a higher-level representation by decomposing and then reconstructing the input, as depicted in Figure 2. As opposed to their historical dimensionality-reduction architectures, current auto-encoders are instantiated as *over-complete* (with an encoder layer of higher dimensionality than the input), in order to learn increasingly higher-level abstractions.

Hence, the goal is to learn an encoding function e and a decoding function d such that $d\left(e\left(\mathbf{x}\right)\right) = \tilde{\mathbf{x}} \approx \mathbf{x}$, therefore, being able to reconstruct an $\tilde{\mathbf{x}}$ similar to the input $\mathbf{x}$ via a hidden representation. The encoding function $e : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_h}$ maps an input $\mathbf{x} \in \mathbb{R}^{d_x}$ to an hidden representation $\mathbf{h_x} \in \mathbb{R}^{d_h}$ by producing a deterministic mapping

$$\mathbf{h_x} = e\left(\mathbf{x}\right) = s_e\left(\mathbf{W_e}\mathbf{x} + \mathbf{b}_e\right)$$

where s_e is a nonlinear activation function (usually the *sigmoid* function), $\mathbf{W}_e$ is a weight matrix , and $\mathbf{b}_e$ is a bias vector. The decoding function $d : \mathbb{R}^{d_h} \rightarrow \mathbb{R}^{d_x}$ then maps back this encoded representation $\mathbf{h_x}$ into a reconstruction $\mathbf{y}$ of the same dimensionnality as $\mathbf{x}$

$$\mathbf{y} = d\left(\mathbf{h_x}\right) = s_d\left(\mathbf{W}_d\mathbf{h_x} + \mathbf{b}_d\right)$$

where s_d is the activation function of the decoder. Hence, training an auto-encoder consists in finding the set of parameters θ that minimizes the reconstruction error of a training dataset $\mathcal{D}_n$

$$\mathcal{J}_{AE}\left(\theta\right) = \sum_{\mathbf{x} \in \mathcal{D}_n} \mathcal{L}\left(\mathbf{x}, d\left(e\left(\mathbf{x}\right)\right)\right)$$

where the reconstruction error function $\mathcal{L}$ is usually the squared error $\mathcal{L}(x, y) = \left\|x - y\right\|^2$ or the cross-entropy loss $\mathcal{L}(x, y) = -\sum_{i=1}^{d_x} x_i log\left(y_i\right) + \left(1 - x_i\right) log\left(1 - y_i\right)$.

Unfortunately, based on the objective function solely minimizing the reconstruction error, nothing prevents an over-complete auto-encoder with an encoding layer of the same (or higher) dimensionnality as the input to simply learn the identity function [4]. Therefore, various regularization techniques have been introduced to skew the learning towards useful representation.

a) Sparse auto-encoders: One solution to avoid the identity function degeneracy is to add a *sparsity* constraint to the cost function [22], [27]. The goal is to ensure that hidden units are mostly inactive for a large portion of the dataset (ie. features learned are *specific*). By computing the average

activation of each hidden unit i across the dataset

$$\hat{\rho}_i = \frac{1}{n}\sum_{j=1}^{n}\left[a_i(\mathbf{x}_j)\right]$$

we can add a penalty term to the objective that penalizes the unit which mean activation $\hat{\rho}_i$ deviates from a *target* sparsity ρ. This is achieved by minimizing the Kullback-Leibler (KL) divergence between all $\hat{\rho}_i$ and the target ρ. Therefore, the complete cost function is defined by the sum of the normal cost function and this sparsity constraint

$$\mathcal{J}_{\text{sparse}}\left(\theta\right) = \mathcal{J}\left(\theta\right) + \beta\sum_{i}\text{KL}\left(\rho, \hat{\rho}_i\right)$$

where β controls the influence of the sparsity on the global cost. Overcomplete sparse AEs can be seen as an attempt to learn a series of traditionnal AE for different types of training data, while sharing some of their hidden structures.

b) Denoising auto-encoders: Another successful way to regularize AEs is the concept of Denoising Auto-Encoders (DAE) [34]. Intuitively, the idea is to corrupt the input $\mathbf{x}$ to obtain a "noisy" version $\tilde{\mathbf{x}}$ that is passed to the AE, but the goal of the network is to reconstruct the original (clean) version of $\mathbf{x}$ (producing an overall denoising process). Training the autoencoder to reconstruct a clean input from a corrupted version of itself forces the hidden layer to uncover robust features but also inherently prevents it from learning the identity function. Hence, the objective function for the DAE is defined as

$$\mathcal{J}_{DAE}\left(\theta\right) = \sum_{\mathbf{x}\in\mathcal{D}_n}\mathbb{E}_{\tilde{\mathbf{x}}\sim q(\mathbf{x}|\tilde{\mathbf{x}})}\left[\mathcal{L}\left(\mathbf{x}, d\left(e\left(\tilde{\mathbf{x}}\right)\right)\right)\right]$$

where the corrupted versions $\tilde{\mathbf{x}}$ are obtained by applying a stochastic function $q\left(\tilde{\mathbf{x}} \mid \mathbf{x}\right)$ to the examples $\mathbf{x}$. The corruption processes are typically additive Gaussian noise and binary masking (a randomly selected subset of input components are set to 0). Variable amounts of corruption (variance of the Gaussian noise or number of dropped components) can be considered to control the degree of regularization.

c) Contractive auto-encoders: Another form of regularization called *contractive* [29] have been proposed. The main idea is to add a penalty based on the derivative of the representation with respect to the input. This encourages the learned features to have low variations in the directions found in the data. This penalty term can be computed through the Frobenius norm of the Jacobian matrix $J_e\left(\mathbf{x}\right)$ of the activations with respect to the input. Formally, given an input $\mathbf{x} \in \mathbb{R}^{d_x}$ mapped to a hidden representation $\mathbf{h} \in \mathbb{R}^{d_h}$, the contractive penalty term is given by

$$\left\|J_e\left(\mathbf{x}\right)\right\|_F^2 = \sum_{ij}\left(\frac{\partial h_j\left(\mathbf{x}\right)}{\partial x_i}\right)^2 = \sum_{i=1}^{d_h}\left(h_i\left(1-h_i\right)\right)^2\sum_{j=1}^{d_x}W_{ij}^2$$

While DAEs (indirectly) encourages the robustness of the reconstruction through corruption, CAEs tries to analytically encourage the robustness of the representation itself by penalizing its variations in the neighborhood of training points. One problem with the penalty introduced by CAE is that it might be limited to only *infinitesimal* variations in the input (because of the use of the first-order derivative). An extension to all higher-order derivatives has been proposed [28] leading to the CAE+H model.

2) Restricted Boltzmann Machine: A Restricted Boltzmann Machine (RBM) is a particular type of Markov random field composed of one layer of binary stochastic hidden units and another layer of stochastic visible units [13]. Visible and hidden units are densely connected by undirected weighted links as depicted in Figure 3. The joint probability distribution $p\left(v, h; \theta\right)$ of the visible units v and hidden units h, given the model parameters θ is defined through an energy function $E\left(v, h; \theta\right)$ such that

$$p\left(v, h; \theta\right) = \frac{e^{-E(v,h;\theta)}}{Z}$$

where $Z = \sum_v\sum_h e^{-E(v,h;\theta)}$ is a normalization factor

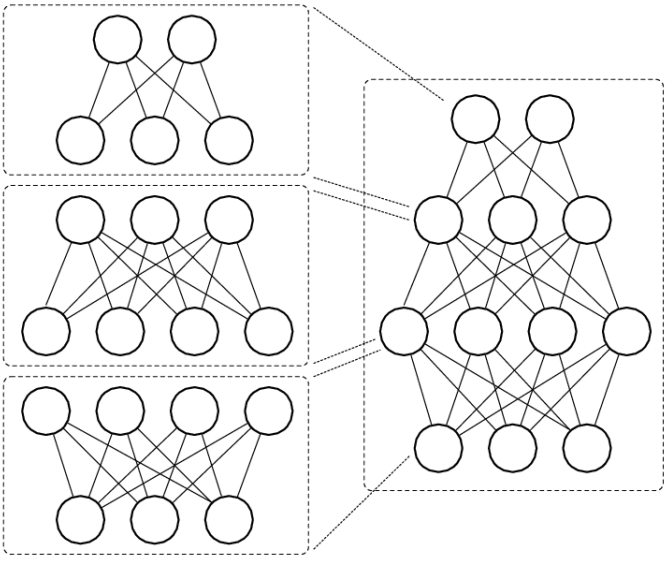

Fig. 3. A Restricted Boltzmann Machine (RBM) is composed of a visible and a hidden layer, which are densely connected by undirected and weighted links.

called the *partition function.* In the case where all units are considered binary variables, the energy function of I visible and J hidden units is defined as

$$E\left(v,h;\theta\right)=\sum_{i=1}^{I}\sum_{j=1}^{J}w_{ij}v_ih_j-\sum_{i=1}^{I}b_iv_i-\sum_{j=1}^{J}a_jh_j$$

where w_{ij} is the symmetric weight between visible unit v_i and hidden unit h_j, and b_i, and a_j are their biases. Similarly to the AE, the conditional probabilities that an hidden or visible unit is active is given by the sigmoid applied to the sum of its weighted input. The probability that the network assigns to an input vector can be raised by lowering its energy (by adjusting the weights and biases of various units). Hence, the goal of learning is that the network places high probabilities to the vectors that are part of the input dataset and low probabilities to the vectors that are not. The gradient of the log-likelihood of an input vector with respect to a weight [13] is surprisingly simple.

$$\frac{\partial\log\,p\left(v;\theta\right)}{\partial w_{ij}}=\mathbb{E}_{data}\left[v_ih_j\right]-\mathbb{E}_{model}\left[v_ih_j\right]$$

where $\mathbb{E}_{data}\left[v_ih_j\right]$ denotes the expectation under the distribution of the training data and $\mathbb{E}_{model}\left[v_ih_j\right]$ is that same expectation under the distribution defined by the model. Unfortunately, the expectation under the distribution of the model $\mathbb{E}_{model}\left[v_ih_j\right]$ is intractable. Hinton et al. [13] proposed a Contrastive Divergence (CD) approximation to the gradient, where $\mathbb{E}_{model}\left[v_ih_j\right]$ is replaced by running a Gibbs chain initialized at the data. Interestingly, it appears that a single full step of the Gibbs chain is sufficient to obtain satisfactory results.

3) Multi-layer stacking and perceptrons: The DAE and RBM models are the building blocks of deep networks in which a single layer of hidden activations provides the set of learned features. However, the goal of deep networks is to learn a structured hierarchy, based on the hypothesis that higher-level concepts can be formed by grouping lower-level ones. Hence, each layer forms increasingly complex features by exploiting the statistical regularities and learning how to optimally group lower-level concepts formed at the previous layer.

Hence, we could construct a deep network by stacking these single-layer models and feeding the hidden representation of a specific layer as input to its following layer. In that case, the stacked layers can be interpreted as a traditional *Multi-Layer Perceptron* (MLP). After all layers have been trained iteratively, the full network is trained through a *fine-tuning* step, usually by minimizing the error rate of a supervised task. In that case, a classification layer can be added on top of the network in order to perform back-propagation on all layers of the network (which makes it exactly similar to the learning procedure of a traditional MLP).

4) Stacked auto-encoders: A stacked autoencoder is formed from a set of k autoencoders, each parametrized by its own weight matrix $\mathbf{W}_e^{(l)}$ and bias vector $b_e^{(l)}$. The output (activation) of each layer is produced by performing a forward pass of the encoding step from one layer to the next

$$a^{(l)} = \phi\left(\mathbf{W}_e^{(l-1)}\mathbf{a}^{(l-1)} + b_e^{(l-1)}\right)$$

by considering that the first layer receives the output $\mathbf{a}^{(0)} = \mathbf{x}$. The stacked auto-encoder ultimately produces a representation $a^{(n)}$ (activations of the deepest hidden units), which can be seen as the highest-order features extracted from the input. These can be used for classification problems which allow to perform a finetuning of this deep model. The error gradients from the supervised classification can then be backpropagated into all the encoding layers together. This global network can thus be trained altogether by improving upon the weights of all layers at each iteration.

5) Deep Belief Network: Deep Belief Networks (DBNs) [13] are a class generative models formed by multiple layers of stochastic computation units. The lowest layers are connected with directed top-down connections while the two upper layers are created with a dense array of undirected symmetric connections, which form an associative memory. Hence a l-layers DBN models the joint distribution between all hidden layers $\mathbf{h}^k$ and the observed data $\mathbf{x}$

$$P\left(\mathbf{x}, \mathbf{h}^1, \ldots, \mathbf{h}^k\right) = \left(\prod_{k=0}^{(l-2)} P\left(\mathbf{h}^k \mid \mathbf{h}^{k+1}\right)\right) P\left(\mathbf{h}^{l-1}, \mathbf{h}^l\right)$$

with $\mathbf{h}^0 = \mathbf{x}$. Hence, stacking layers of RBMs from the lowest (*visible* data) to the upper (*associative memory*) layer can provide a DBN architecture. In that case, we consider that the activation probabilities of the hidden layer of one RBM becomes the visible data for the next RBM layer. It has been shown that this stacking procedure improves the variational lower bound of the data log-likelihood [13]. This means that learning a DBN with this procedure provides a close approximation to the true maximum likelihood learning.

6) Deep Boltzmann Machine: The joint training of layers in a DBN is problematic as its inference problem is often intractable. The Deep Boltzmann Machine (DBM) has been proposed to allow a joint training of all layers in a purely unsupervised manner [31]. Similarly to the RBM, a DBM is a specific type of Boltzmann machine with layered hidden units. However, the DBM is composed of multiple layers, in which conditionnal independence is imposed between odd-numbered layers and even-numbered layers. Unlike DBN, their inference procedure can incorporate top-down feedback, which allows performing a better propagation of uncertainty.

A 2-layer DBM (with one visible and two hidden layers $\left\{\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2\right\}$) can be defined by specifying its energy function

$$E\left(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2; \theta\right) = -\mathbf{v}^T\mathbf{W}^1\mathbf{h}^1 - \mathbf{h}^{1T}\mathbf{W}^2\mathbf{h}^2$$

The probability assigned to a visible vector is given by

$$p\left(\mathbf{v}, \theta\right) = \frac{1}{Z}\sum_{\mathbf{h}^1, \mathbf{h}^2} \exp\left(-E\left(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2; \theta\right)\right)$$

Even though the conditional distributions over the visible and last hidden units are defined similarly to the RBM, the distribution over the "middle" layer of hidden units is defined as

$$p\left(h_j^1 \mid \mathbf{v}, \mathbf{h}^2\right) = \sigma\left(\sum_i W_{ij}^1 v_i + \sum_m W_{jm}^2 h_j^2\right)$$

Hence, the major problem with DBM is that these interactions between hidden units make the posterior of hidden units untractable. To solve this problem, [31] propose to rely on a mean-field approximation (similar to the ideas of *variational learning*), where the posterior distribution $P\left(\mathbf{h}^1, \mathbf{h}^2 \mid \mathbf{v}\right)$ is approximated with a factored distribution $Q\left(\mathbf{h}^1, \mathbf{h}^2\right) = \prod_j Q\left(h_j^1\right)\prod_i Q\left(h_i^2\right)$ such that the KL divergence between the real and approximated distributions $KL\left(P\left(\mathbf{h}^1, \mathbf{h}^2 \mid \mathbf{v}\right) \parallel Q\left(\mathbf{h}^1, \mathbf{h}^2\right)\right)$ is minimized.

III. Evaluation

A. Datasets

We evaluate the algorithms on the complete UCR time series datasets [20], which is the reference collection for TSC

Algorithm 1 Experimental evaluation procedure

Input
Dataset $\mathcal{S}_{train} = \langle \mathcal{S}^1_{train}, \ldots, \mathcal{S}^n_{train} \rangle$
Dataset $\mathcal{S}_{test} = \langle \mathcal{S}^1_{test}, \ldots, \mathcal{S}^m_{test} \rangle$
Distances set $\mathcal{D}_{set}$ (section **??**)
Criterias $\mathcal{C}_{set}$ (section **??**)
1: **for** each distance $\mathcal{D}_i$ in $\mathcal{D}_{set}$ **do**
2: **if** $\mathcal{D}_i$ requires parameters $\mathcal{P}_{\mathcal{D}_i}$ **then**
3: optimize parameters $\mathcal{P}_{\mathcal{D}_i}$ through LOO on $\mathcal{S}_{train}$
4: **for** each criteria $\mathcal{C}_i$ in $\mathcal{C}_{set}$ **do**
5: optimize space $\mathcal{M}_{\mathcal{C}_i}$ through LOO on $\mathcal{S}_{train}$
7: classify $\mathcal{S}_{test}$ with parameters $\{\mathcal{M}_{\mathcal{C}_i}, \mathcal{P}_{\mathcal{D}_i}\}$
8: **return** error rate on $\mathcal{S}_{test}$

evaluation and also include all datasets used for the evaluation of the time series ensembles [24]. Overall, our collection contains 75 datasets, coming from a variety of scientific fields. All datasets are composed of univariate time series separated into *training* and *testing* dataset with disparate characteristics. Their properties range from 2 to 60 classes (mean 7.72, median 4), from 16 to 8926 training samples (mean 583.6, median 322) and from 28 to 8926 testing samples (mean 1109.3, median 390) with a train-to-test ratio between 0.01 and 4.49 (mean 0.9, median 0.98). The time series themselves also have a wide distribution of length ranging from 24 to 1882 points (mean 386.9, median 300).

B. Experimental procedure

In order to evaluate our approach, we follow an experimental procedure similar to previous benchmarking studies [1], [9]. We underline the argument put forward by Keogh and Kasetty [17] that such comparisons should try to be free of both *implementation* and *data bias*. Hence, we rely on all state-of-art datasets and report the error rates from other authors' implementations. This implies that we avoid any form of cherry-picking (against data bias) and that implementation problems can only hamper our own results. We compute the error rates of each dataset by following the established train/test datasets, as detailed in [20] and [24]. This allows us to directly compare our error rates to previously published results from [20] and [24].

The evaluation procedure takes as input a training dataset $\mathcal{S}_{train}$ and a testing dataset $\mathcal{S}_{test}$ of time series, a set of [...]

1) Models: For the models, we explore the different variants described in Section II-B. First, we evaluate both AE and RBM as single-layer models. Once these are pre-trained following a specific architecture (topology of connexions, detailed in the following section), we add a logistic classifier layer and evaluate their classification accuracy by considering the whole network as an MLP. These models are simply termed AE and RBM in the following. For the AE, we further tested its various regularizations, by comparing both DAE and CAE with the same parameters.

Second, we investigate more advanced multi-layer models, based on the same pre-trained layers. Hence, after the pre-training operation, we instantiate an SAE, DBM or DBN (depending on whether the pre-trained layers are AE or RBM). All the models are summarized in Table

2) Architectures: The architecture of the network can be of crucial importance, but is also a complex set of parameters that can hardly be driven by the data. Three major factors come into play regarding the architecture of connexions inside the network. First, the *depth* of the network is defined by the number of layers it contains. Second, the *width* of the network is defined by the number of units contained in the layers. Finally, the *topology* of the network can be loosely defined as the relative size of layers from one to another. We delineated four major categories, namely *linear, increasing*, *decreasing* and *bottleneck*. In order to test various combinations of parameters, we instantiated a set of architectures detailed in Table

3) Computational units:

4) Parameters:

C. Statistical significance

Based on our evaluation framework, we obtain the error rates for each classifier over every datasets. However, as noted by [32], there are several pitfalls to avoid when comparing

4	inc	1000 1500
4	dec	1500 1000
4	lin	1500 1500
5	inc	500 1000 1500
5	dec	1500 1000 500
5	lin	1000 1000 1000
5	inc	500 1000 1500 2000
6	bot	500 1500 1000 2000
6	dec	2000 1500 1000 500
6	lin	1000 1000 1000 1000
7	bot	250 500 1000 2000 250
7	inc	500 1500 1000 2000 250

TABLE I. ARCHITECTURES EVALUATED

classifiers. Therefore, in order to assess the relative performances of various classifiers, we compute the statistical significance of these results by following the recommendations of Salzberg [32] and Demsar [8]. First, we avoid the *multiplicity effect* by relying on the Tukey-Kramer Honestly Significant Difference (HSD) test over the results of Friedman's ANOVA [8]. Then, we use a post-hoc Nemenyi test to exhibit the critical differences that might exist between classifiers. Finally, we present the *critical difference graphs* [8], which show the statistical differences and eventual cliques of statistical equivalence between various methods. We compute all these statistics between the results provided on the UCR website [20] for the best results of 1-NN on DTW, time series ensembles [24] and all our evaluated models of AE, RBM, SAE, DBN and DBM.

D. Hardware

As our testing methodology is computationally intensive, we performed all our calculations on the Mesu supercomputer from the Pierre & Marie Curie University funded by the Equip@Meso project carried by GENCI. The Mesu server is equipped with 64 computes nodes each containing 2 Intel Xeon (Sandy Bridge) with 8 cores and 256 GB RAM.

E. Repeatability

The complete code (distance measures, classification methods and optimization functions) for repeating our experiments is available as an open source package in Matlab at

CHANGE https://github.com/esling/hvmots. **CHANGE**

All the results from this paper can be obtained simply by running the `deepFunction` script with various parameters. The datasets used are either freely available for download from the UCR collection website [20] or the time series ensembles website [24].

IV. RESULTS

We start by providing an in-depth analysis of the results depending on the various models, architectures and parameters . Then, we provide a comparison between the best performing deep models and various state-of-art methods (Section IV-B)

A. Deep learning benchmarking

1) Models:

2) Architectures: The architecture of the network

3) Learning variants:

4) Parameters:

B. State-of-art comparison

We provide in Table II the comparison of error rates obtained by different methods on each dataset. We display the results provided by the UCR website [20] for 1-NN selection with DTW with best window (DTW) and time series ensembles (TSE) [24]. The last columns display the results of the auto-encoder (AE), Restricted Boltzmann Machine (RBM), Stacked auto-encoder (SAE), Deep Belief Network (DBN) and Deep Boltzmann Machine (DBM). Dark grey rows identify the datasets where deep learning models obtain a lower error rate than all state-of-art methods, while light grey rows show an equivalent error rate.

As we can see [...]

Finally, we provide in Figure 4 the *critical difference graphs* [8], in order to exhibit the true statistical differences between classifiers across all datasets within a single representation. These critical differences are computed with [...]

TABLE II. COMPARISON OF OVERALL ERROR RATES FOR THE DTW WITH BEST WINDOW (DTW), TIME SERIES ENSEMBLES (TSE) [24] AND THE BEST PERFORMING AUTO-ENCODER (AE), RESTRICTED BOLTZMANN MACHINE (RBM), STACKED AUTO-ENCODER (SAE), DEEP BELIEF NETWORK (DBN) AND DEEP BOLTZMANN MACHINE (DBM).

	DTW	TSE	AE	RBM	SAE	DBN	DBM
50Words	24.2	**18.0**				27.9	25.9
Adiac	39.1	35.3				36.1	**27.9**
ArrowHead	21.7	**16**				20.1	24.1
ARSim	44.3	10.3				**0**	**0**
Beef	46.7	36.7				**31**	34.5
BeetleFly	35	40				26.3	**21.1**
BirdChicken	35	35				**21.1**	**21.1**
Car	23.3	**16.7**				21.7	**16.7**
CBF	0.5	0.2				1.1	**0**
Chlorine	35	36				**33.6**	**33.6**
Cinc_ECG	7	6.2				6.1	**2.9**
Coffee	17.9	**0**				**0**	**0**
Computers	12.4	**11.6**				16	**11.6**
CricketX	23.6	**20.3**				21.3	**20.3**
CricketY	19.7	**15.6**				22.1	19.2
CricketZ	18	**15.6**				23.4	21.3
DiatomSize	6.5	5.9				**5.2**	**5.2**
DistalPOA	20.1	22.3				24.5	**17.4**
DistalPOC	25.4	**23.2**				23.5	**20.5**
DistalPTW	32.4	31.7				32.4	**30.9**
Earthquakes	30.9	**28.1**				**28.1**	**28.1**
ECGFiveD	20.3	17.8				**2.9**	**4.9**
ElectricDevices	29.5	27.7				27.2	**26.5**
Face (all)	19.2	**15.2**				20.2	20.2
Face (four)	11.4	9.1				6.9	**5.7**
FacesUCR	8.8	6.3				4.3	**3.7**
Fish	16	3.4				8.1	**3.1**
GunPoint	8.7	7				2.7	**2**
Haptics	58.8	58.4				57.3	**55.1**
Herring	34.4	**29.7**				33.3	34.9
InlineSkate	61.3	56.7				55.7	**55.5**
ItalyPower	4.5	3.9				4.1	**3**
LargeKitchen	26.4	23.2				26.4	**22.3**
Lightning2	13.1	11.5				13.1	**10**
Lightning7	28.8	23.3				30.5	**16.7**
MALLAT	8.6	5				8.1	**3.8**
Medical	25.3	**24.5**				27.4	**24.5**
MiddlePOA	53.9	47.4				46.4	**43.5**
MiddlePOC	**19.9**	21				**19.9**	**19.9**
MiddlePTW	68.2	63				57	**54.5**
MoteStrain	13.4	**11.4**				14.5	12.5
N-ECG 1	18.5	17.8				16.2	**15.9**
N-ECG 2	12.9	11.2				11.2	**10.5**
OliveOil	16.7	13.3				**10.3**	**10.3**
OSU Leaf	38.4	**19.4**				25.3	19.9
Phalanges	22.8	**21.7**				21.9	21.9
Planes	**0**	**0**				**0**	**0**
ProximalPOA	12.7	11.7				11.5	**11.2**
ProximalPOC	19.6	17.2				17.5	17.5
ProximalPTW	27.8	24.4				24.4	**22.5**
Refrigeration	51.5	42.4				36.5	**36.5**
ScreenTypes	44.5	44				44	**42.7**
ShapesAll	24.7	**18.7**				19	19
SmallKitchen	25.6	**23.2**				**23.2**	**23.2**
SonyI	30.5	29.3				31.3	**28**
SonyII	14.1	12.4				19	**8.4**
StarLight	9.5	7.9				9.2	**6.9**
SwedishL	15.7	8.5				11.4	**7.2**
Symbols	6.2	4.9				3.8	**3.4**
Synthetic	1.7	**1**				**1**	**1**
ToeSegment1	10.1	7.9				8.9	**7.5**
ToeSegment2	10.8	8.5				8.5	**8.4**
Trace	1	1				1	**0**
TwoPatt.	0.1	**0**				**0**	**0**
TwoLead	13.2	6.7				8.8	**5.9**
uWaveX	22.7	**19.9**				22.5	22.5
uWaveY	30.1	**28.3**				30.5	30.1
uWaveZ	32.2	**29**				31.8	30
Wafer	0.5	**0.3**				0.5	**0.3**
Words	25.2	22.6				25.4	**20.1**
Yoga	15.5	**12.1**				25.4	**12.1**

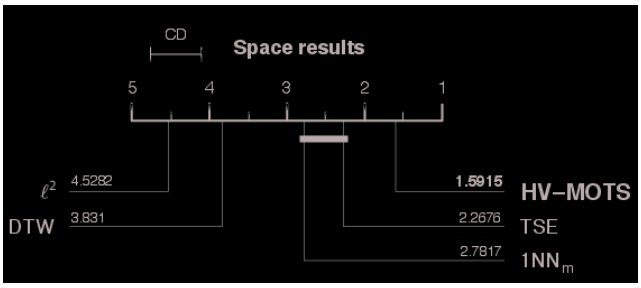

Fig. 4. Critical differences graphs comparing the performances of the state-of-art approaches (ℓ^2, DTW and TSE) and the multidimensional assessment of similarity by relying either on a nearest neighbor ($1NN_m$) or hypervolume (HV-MOTS) criterion. These critical differences are computed either between the *best* distance space (top) or the *space* automatically selected by the optimization procedure (bottom).

1) Computational efficiency:

2) Interpretability and generalization:

C. Motif discovery

V. DISCUSSION

First evidence

* ML For TSC

* Interpretability

* Framework providing simultaneous motifs & classification

+ Generic + Computation

=¿ But we don't even enforce smoothness

We on purpose did not use models such as semi-RBM

Could further improve the results by enforcing ...

VI. CONCLUSION AND FUTURE WORK

We showed in this paper ...

REFERENCES

[1] A. Bagnall, L. Davis, J. Hills, and J. Lines. Transformation based ensembles for time series classification. *Proc. 12th SDM*, 2012.

[2] BR Bakshi and G. Stephanopoulos. Representation of process trends–IV. Induction of real-time patterns from operating data for diagnosis and supervisory control. *Computers & Chemical Engineering*, 18(4):303–332, 1994.

[3] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[4] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[5] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAI-94 workshop on knowledge discovery in databases*, pages 229–248, 1994.

[6] L. Chen and R. Ng. On the marriage of Lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 792–803. VLDB Endowment, 2004.

[7] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *Principles of data mining and knowledge discovery: First European Symposium, PKDD'97, June 24-27*, volume 1263, pages 88–100, Trondheim, Norway, 1997. Springer Verlag.

[8] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[9] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[10] P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12, 2012.

[11] S. Gudmundsson, T.P. Runarsson, and S. Sigurdsson. Support vector machines and dynamic time warping for time series. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2772–2776. IEEE, 2008.

[12] N. Hammami and M. Sellam. Tree distribution classifier for automatic spoken arabic digit recognition. In *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*, pages 1–4. IEEE, 2009.

[13] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[14] Eric J Humphrey, Juan P Bello, and Yann LeCun. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, 2013.

[15] S.L. Jeng and Y.T. Huang. Time Series Classification Based on Spectral Analysis. *Communications in Statistics-Simulation and Computation*, 37(1):132–142, 2008.

[16] Young-Seon Jeong, Myong K Jeong, and Olufemi A Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 44(9):2231–2240, 2011.

[17] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[18] E. Keogh, S. Lonardi, and C.A. Ratanamahatana. Towards parameter-free data mining. In *Proceedings of 10th ACM international conference on Knowledge discovery and data mining*, pages 206–215, 2004.

[19] E. Keogh and M. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Proceedings of the 4th International Conference of Knowledge Discovery and Data Mining*, pages 239–241. AAAI Press, 1998.

[20] E. Keogh, Q. Zhu, B. Hu, Hao. Y., X. Xi, L. Wei, and C. A. Ratanamahatana. The ucr time series classification/clustering homepage. http://www.cs.ucr.edu/~eamonn/time_series_data/, 2011.

[21] L.J. Latecki, Q. Wang, S. Koknar-Tezel, and V. Megalooikonomou. Optimal subsequence bijection. In *IEEE Int. Conf. on Data Mining (ICDM)*, pages 565–570, Omaha, USA, 2007.

[22] Honglak Lee, Chaitanya Ekanadham, and Andrew Y Ng. Sparse deep belief net model for visual area v2. In *Advances in neural information processing systems*, pages 873–880, 2008.

[23] T. Lin, N. Kaminski, and Z. Bar-Joseph. Alignment and classification of time series gene expression in clinical studies. *Bioinformatics*, 24(13):147–155, 2008.

[24] Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, pages 1–28, 2014.

[25] A. Perina, M. Cristani, U. Castellani, and V. Murino. A new generative feature set based on entropy distance for discriminative classification. *Image Analysis and Processing–ICIAP 2009*, pages 199–208, 2009.

[26] M. Radovanović, A. Nanopoulos, and M. Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *The Journal of Machine Learning Research*, 9999:2487–2531, 2010.

[27] Marc-Aurelio Ranzato, Y-lan Boureau, and Yann L Cun. Sparse feature learning for deep belief networks. In *Advances in neural information processing systems*, pages 1185–1192, 2007.

[28] Salah Rifai, Grégoire Mesnil, Pascal Vincent, Xavier Muller, Yoshua Bengio, Yann Dauphin, and Xavier Glorot. Higher order contractive auto-encoder. In *Machine Learning and Knowledge Discovery in Databases*, pages 645–660. Springer, 2011.

[29] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature

extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840, 2011.

[30] DE Rumelhart, GE Hinton, and RJ Williams. Learning internal representations by error propagation. In *Neurocomputing: foundations of research*, pages 673–695. MIT Press, 1988.

[31] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 448–455, 2009.

[32] S.L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and knowledge discovery*, 1(3):317–328, 1997.

[33] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[34] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[35] X. Xi, E. Keogh, C. Shelton, L. Wei, and C.A. Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning ICML 06*, volume 150, pages 1033–1040. ACM Press, 2006.

[36] H. Xia, G.A. Garcia, J.C. McBride, A. Sullivan, T. De Bock, J. Bains, D.C. Wortham, and X. Zhao. Computer algorithms for evaluating the quality of ecgs in real time. 2012.

[37] S. Zhong and J. Ghosh. HMMs and coupled HMMs for multi-channel EEG classification. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 1154–1159, 2002.