Interpretable time series neural representation for classification purposes

Etienne Le Naour, Ghislain Agoua, Nicolas Baskiotis, Vincent Guigue

Interpretable time series neural representation for classification purposes

Etienne Le Naour
Sorbonne Université, EDF R&D
Paris, France
etienne.le-naour@edf.fr

Ghislain Agoua
EDF R&D
Palaiseau, France
ghislain.agoua@edf.fr

Nicolas Baskiotis
Sorbonne Université
Paris, France
nicolas.baskiotis@isir.upmc.fr

Vincent Guigue
AgroParisTech
Palaiseau, France
vincent.guigue@isir.upmc.fr

***Abstract*—Deep learning has made significant advances in creating efficient representations of time series data by automatically identifying complex patterns. However, these approaches lack interpretability, as the time series is transformed into a latent vector that is not easily interpretable. On the other hand, Symbolic Aggregate approximation (SAX) methods allow the creation of symbolic representations that can be interpreted but do not capture complex patterns effectively. In this work, we propose a set of requirements for a neural representation of univariate time series to be interpretable. We propose a new unsupervised neural architecture that meets these requirements. The proposed model produces consistent, discrete, interpretable, and visualizable representations. The model is learned independently of any downstream tasks in an unsupervised setting to ensure robustness. As a demonstration of the effectiveness of the proposed model, we propose experiments on classification tasks using UCR archive datasets. The obtained results are extensively compared to other interpretable models and state-of-the-art neural representation learning models. The experiments show that the proposed model yields, on average better results than other interpretable approaches on multiple datasets. We also present qualitative experiments to asses the interpretability of the approach.**

***Index Terms*—Representation learning, unsupervised learning, time series, interpretability, classification**

I. Introduction

Unsupervised representation learning approaches aim to build representation for time series by capturing their underlying distribution without expert knowledge or human supervision. They have demonstrated good performances for clustering [1], classification [2]–[4], missing values imputation or forecasting [5]. Despite these good performances for downstream tasks, the neural representations models in the literature lack interpretability. In [6] a review of representation learning, the authors emphasize that good representations should have the ability to extract *Explanatory Factors* and should guarantee *Temporal Consistency*. Current approaches do not meet these criteria. Indeed, for most existing approaches [2], [4], [7], the representation results from mapping signals to a latent vector with no temporal consistency and in which weights have no meaning. These representations fail to provide interpretability when used for downstream tasks like classification, which is problematic for critical decision-making.

However, interpretability is a concept that is not universally agreed upon [8], with confusion arising from the different meanings of interpretability and explicability. For time series models, [9] offers a clear taxonomy of the interpretability shown in Figure 1. Post-hoc interpretability refers to methods that analyze the model after the training and are generally model-agnostic. It is often related to the eXplainable Artificial Intelligence (XAI) research field. However, post-hoc methods only explain the decision for a specific instance (or specific features), and additional methods are required to understand the overall model.

On the other hand, in-situ interpretable models are self interpretable. The interpretability arises directly from the model without any other process being applied after the training phase [9], [10]. The level of this interpretability can be local or global. In [8], global interpretability is defined as a model that is easy for a human to understand and requires low computational complexity. In contrast, local interpretability is a way to interpret a model's decision for a particular instance. Global interpretability often implies local interpretability, but the reverse is not true.

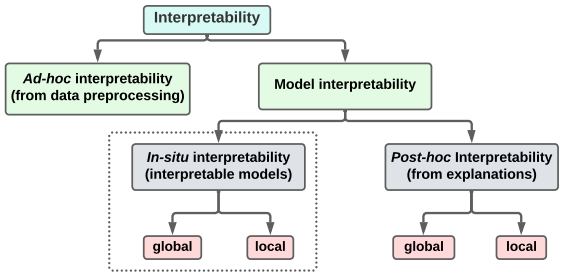

Fig. 1. Interpretable time series model taxonomy introduced in [9]

In this work, we focus on global in-situ interpretability rather than post-hoc explainability, as global in-situ models are inherently interpretable and can be understood both for individual instances and the model as a whole. This paper aims to develop a global in-situ interpretable neural method for time series representation. The first contribution of this work is to define the requirement to bridge the gap between symbolic representation and neural representation to ensure a global interpretable neural symbolic representation for time series data. Indeed, most successful interpretable models come from symbolic machine learning, such as symbolic aggregate approximation (SAX), which creates interpretable symbolic

representations of time series data. However, the information captured by these symbols is limited and does not provide global interpretability of the representation. On the other hand, neural representation learning methods achieve great performances but are definitively not in-situ interpretable. Section III analyses the criteria that must be respected to guarantee global in-situ interpretability.

We propose a novel unsupervised neural network that fills these requirements in Section IV. The neural network is based on an auto-encoder architecture and vector quantization mechanism [11]–[13]. The unsupervised setting is a crucial choice for the generalization of the learned representation. Moreover, it allows to re-use the extracted representation for several classification tasks.

To demonstrate the qualities of the proposed architecture, Section V presents an application of our learned symbolic neural representation to classification tasks. A simple linear classifier over the interpretable symbolic representations is used to solve classification tasks efficiently. The linearity of the classifier preserves interpretability in the representation, providing both global and local interpretability for understanding the decision made by the classifier.

Our main contributions can be summarized as follows:

- We define and formalize the fundamental requirements to construct interpretable symbolic neural representations for time series.
- We propose an unsupervised neural network architecture that satisfies the above requirements.
- We use these representations for downstream classification tasks (while preserving interpretability of the representation) and evaluate them through quantitative experiments on the UCR archive.
- We provide qualitative experiments to capture local and global interpretability.

II. Related content

Constructing time series representations is a fundamental challenge that can be performed unsupervised or based on a specific task. This related content presents both neural representation methods, which are not easy to interpret, and classical representation methods, which are interpretable. In most cases, classical representation methods are combined with a classification task.

a) Unsupervised neural representation learning for time series classification: In recent works, models of neural representations of time series have emerged. These models typically learn the representation in an unsupervised way and then use the learned representation for a specific task in a second step. In [3], the authors use a deep unsupervised representation to solve a subsequent classification task for time series. They used an architecture composed of Recurrent Neural Networks to build a representation that would later be used for classification. In [2], authors build a time series representation using a convolutional encoder and a contrastive loss [14]. The representation space brings together series (and sub-series) that are similar. Afterward, a support vector machine (SVM) is applied on top of the representation to solve the classification problem. Then, several papers attempted to construct vector representations of time series using contrastive loss [4], [7]. Some recent works [5] have tried to build an unsupervised representation of time series using transformers mechanisms [15] inside an auto-encoder. These neural representations can capture a lot of information, and the downstream tasks learned from them are very efficient. However, the representations and thus the downstream tasks cannot be interpreted with these models.

b) Attempt to construct an interpretable neural representation for time series: Recently, progress has been made in making neural representations of time series interpretable. In [16], the authors have attempted to decompose time series into disentangled semantic factors (for both individual factors and group segment factors) using Variational Auto-Encoder (VAE), LSTM, and a disentanglement strategy. The limitation of disentangled representations is that it is difficult to assess the disentanglement of latent factors when the initial semantic properties of the time series are unknown. This unsupervised representation is interesting for generation, but difficult to adapt for classification. Other work, such as [17], has attempted to obtain interpretable differential operators from multivariate time series. However, they are specific to forecasting.

Although neural methods for representing time series are relatively new, non neural interpretable methods for representing time series are widely studied.

c) The Symbolic Aggregate approximation (SAX): SAX methods [18], [19] create a symbolic representation of a time series by combining local statistics, which are calculated by taking the average of different segments of the time series. Each average is assigned a symbol based on its value. A sequential symbolic representation of the original time series is formed by mapping these local averages to symbols. Multiple SAX representations can be obtained by taking more or less local averages. SAX symbolic elements operate in the time domain and are interpretable, unlike the Symbolic Fourier approximation (SFA) method [20], which operates in the frequency domain. In [18], the authors introduce the SAX method as a versatile unsupervised approach for various downstream tasks, but in practice, SAX methods are typically used in combination with a classifier on top of the representation. There are several methods to classify on top of SAX representations. For example, the *SAX-VSM* method [21] is based on the frequency of subsequences within each class. The *SAX-SEQL* method [22] searches the entire space of subsequences for the most discriminating subsequences using logistic regression based on coordinate descent. SAX methods have proven useful for making interpretable classification decisions from the constructed symbolic representation. However, this interpretability is only local. Indeed, when we obtain a discriminating symbolic subsequence, we can highlight the corresponding sub-part of the time series. Nevertheless, we cannot reconstruct what the model has learned because several sub-series can give the same symbolic sub-sequence. Additionally, as stated in [8],

when the number of representations used for classification is excessive, such as in the case of multi-SAX-SEQL [23], it becomes difficult to interpret the classification results.

d) Shapelet methods: These methods aim to find the subsequences of the time series that most discriminate between classes [24]. This representation method is supervised because it relies on a downstream task. In addition, the resulting representations are partial, since only the shapelets that discriminate between classes are extracted at the end of the process. Once the optimal shapelets are found, if the classifier applied on top of them is interpretable (linear regression or decision tree), the whole process is interpretable at both local and global levels. However, the original method is costly because it is necessary to search the whole space for possible subsequences. In *Fast Shapelets* (FS) [25], the authors proposed to speed up the discovery of discriminative subsequences by using a SAX representation to discover the subparts of the series where shapelets should be searched. Although the method speeds up the discovery of discriminative shapelets, it remains computationally expensive and the accuracy could be better. To avoid these problems, in [26], authors propose the *Learning Shapelets* (LTS) method. The goal is to learn the discriminating shapelets rather than to search for them in the whole space of possibilities. This method has led to improvements in accuracy. However, it generates a large number of shapelets that are almost the same. This negatively affects interpretability. Afterward, methods have been proposed to learn a limited number of shapelets and thus reinforce the interpretability of the model. In [27], authors propose learning a small number of discriminating shapelets by training a Generative Advertial Network (GAN) and a classifier.

To the best of our knowledge, there is no unsupervised learning method capable of learning an interpretable neural representation for time series. In the next section, we set out the requirements for constructing interpretable symbolic neural representations. These criteria will link traditional methods of representing time series using symbols and unsupervised neural representation methods.

III. Requirements for an interpretable symbolic neural representation

In order to ensure that a neural representation of a time series is interpretable, we set out several requirements that we consider essential.

For this purpose, we introduce some notation for this section. We consider that we have a dataset of N samples. For an instance i ($i \in \{1, ..., N\}$), the univariate time series is denoted by the vector $\boldsymbol{x_i}$ of length T: $\boldsymbol{x_i} = (x_{i,1}, \ldots, x_{i,T}) \in \mathbb{R}^T$. Let $\boldsymbol{r_i}$ be the symbolic neural representation composed of $T^{'}$ elements for $\boldsymbol{x_i}$. We denote by $\mathbb{A}$ the support (alphabet) common to all these elements: $\boldsymbol{r_i} = (r_{i,1}, \ldots, r_{i,T^{'}}) \in \mathbb{A}^{T^{'}}$. Then ϕ_θ is the function that maps the time series into the representation, and $\psi_{\theta'}$ is the function that goes from the representation to the reconstruction space of the time series. To simplify the reading, we omit the indices i for the vectors $\boldsymbol{r}$ and $\boldsymbol{x}$.

a) Requirement n°1 - discrete symbolic representation: The purpose of a symbolic neural representation method is to capture complex phenomena within the representation (as neural representations do) while being able to interpret and visualize the representation elements (as symbolic representations tend to do). Thus, the support $\mathbb{A}$ of each element must be discrete and limited (e.g. $Card(\mathbb{A}) = 32$). In addition, the support must be common to all elements of the symbolic representation. This limits the number of possible patterns. Once we obtain the symbolic representation $\boldsymbol{r}$, we can use the classifiers used in the dictionary methods [22], [23], [28] (see Section V).

b) Requirement n°2 - temporal consistency: For a time series $\boldsymbol{x}$ of length T, learning a contracted representation $\boldsymbol{r}$ of length $T^{'}$ will mechanically lead to a contraction of the time dimension ($T^{'} < T$). We then define temporal consistency by two properties. First, each element of the representation is a function of a portion of the original time series. Thus, for each element of the representation, we must be able to compute the pre-image of the element. Second, the representation must preserve the original temporal order despite the contraction of the temporal dimension. To illustrate this property, consider the case where $r_{t_1'}$ is an element of the representation and $r_{t_2'}$ is another element of the representation that occurs after $r_{t_1'}$. As shown in Figure 2, the temporal consistency of the representation ensures that the pre-image of $r_{t_1'}$ must precede the pre-image of $r_{t_2'}$ in time.

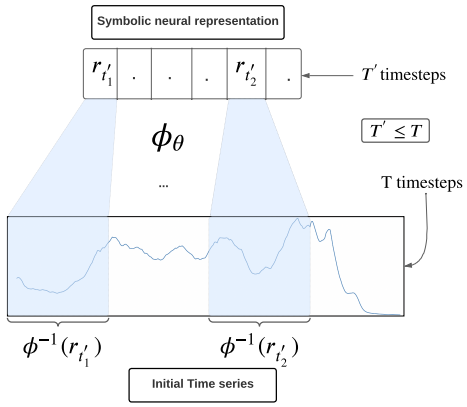

Fig. 2. Temporal consistency visualization. *Coffee dataset*

c) Requirement n°3 - a decodable representation: Being able to visualize the portion of the time series related to a specific element of the representation is important, but it is also important to be able to see what the model has learned overall. As shown in Figure 3, $\psi_{\theta'}$ should be able to reconstruct the entire time series, as well as specific parts of it, while maintaining temporal consistency.

d) Requirement n°4 - shift equivariance properties: We want the shift equivariance property for both ϕ_θ and $\psi_{\theta'}$. The ϕ_θ shift equivariance means that two patterns in the initial time series that are identical but do not occur at the same time should be encoded with the same value but not at the same

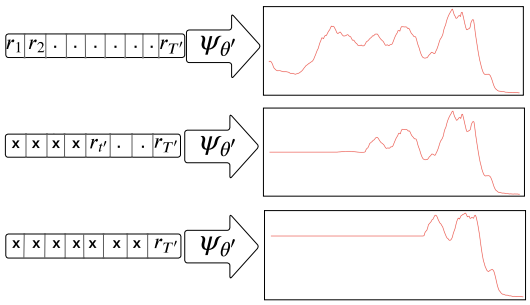

Fig. 3. Visualization of how the representation is decoded for the complete representation, as well as for half of the representation and for only the last element. *Coffee dataset*.

place in the representation. The $\psi_{\theta'}$ shift equivariance means that two elements of the representation that have the same value but do not occur at the same time should represent the same pattern when decoded (with a time shift). This property is essential to interpret the representation elements and ensure that the same value in the representation, regardless of its position, represents the same pattern. We illustrated the shift equivariance property for $\psi_{\theta'}$ in Figure 4.

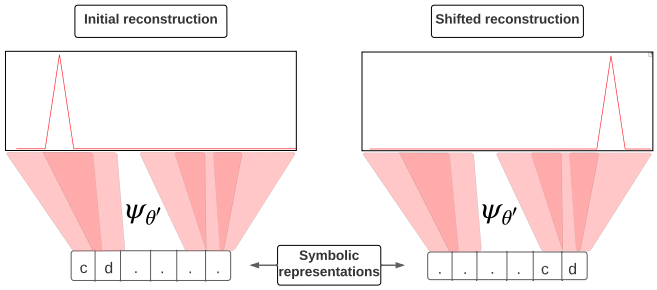

Fig. 4. Shift equivariance of $\psi_{\theta'}$. The subsequence *c d* appear at two different locations in the representation: $\psi_{\theta'}$ should decode the same pattern with a shift in time.

Let's formally define the shift equivariance property. Let S be an element of a sequence (a single element or a subsequence), and G_T the group of discrete translations along the temporal axis. If we take τ to be any discrete translation in G_T and f to be a function equivariant by discrete translation for G_T, then there exists $\tau' \in G_{T'}$ such that: $f(\tau(S)) = \tau'(f(S))$.

e) Requirement n°5 - a representation adjustable to the frequency level: Like an image that cannot be interpreted at the pixel level, a time series is difficult to interpret at the point level. This requirement aims to control the amount of information captured when creating the representation. For the representation to be easily understood, it is crucial to capture the appropriate frequency levels that define the time series. As illustrated in Figure 5, the depth of the representation determines whether it focuses on lower or higher frequency features, which in turn affects the length of the representation.

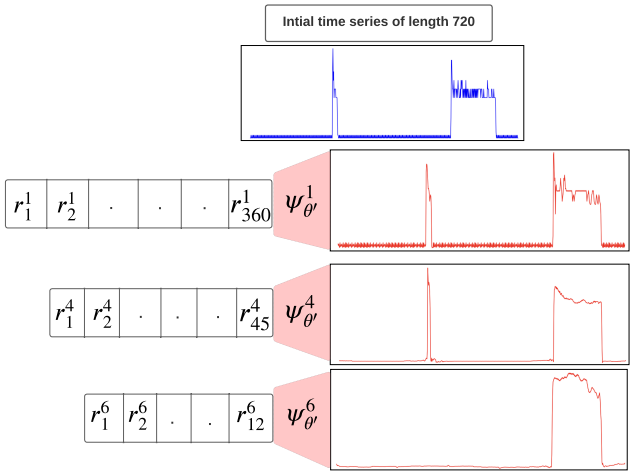

Fig. 5. Visualization of different reconstructions for different level of representation. We can see that the deeper the architecture, the shorter the representation and the smoother the reconstruction. *Computers dataset.*

IV. Model

This section proposes an unsupervised model that respects the different requirements for building an interpretable symbolic neural representation.

A. Architecture

The proposed unsupervised model architecture consists of an encoder decoder structure with a discretization mechanism within the representation space. First, the time series $\boldsymbol{x}$ is given as input to the encoder. Second, the output of the encoder is discretized using the vector quantization mechanism. This step allows us to obtain $\boldsymbol{r}$ after the learning process. Finally, the discretized elements are passed to the decoder, which returns a reconstruction of the time series $\hat{\boldsymbol{x}}$.

The architecture takes elements from the Vector Quantization Variationnal Auto-Encoder (VQ-VAE) [12], which was the first model to establish a latent space of discrete representation within an auto-encoder. However, we add constraints to the architecture and remove the variational part in order to meet the requirements defined in Section III. The different parts of the architecture are described below and Figure 8 illustrates the global unsupervised architecture. Let us look at the architecture in detail, starting with the encoder.

a) Encoder: The encoder can be divided into a sequence of consecutive blocks with the same structure. As shown in Figure 6, a block consists in three operations: a 1D convolution layer, a downsampling operation, and a non-linear activation function.

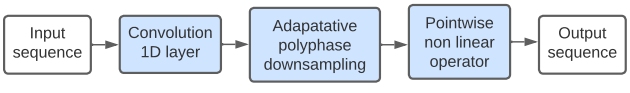

Fig. 6. Inside an encoder block.

The convolution layer is non-strided and has a dilation factor of zero. The input sequence is padded with a zero on each side,

and the kernel size is three. Therefore, the convolution layer's output sequence length remains unchanged. Subsequently, we perform a downsampling operation on the sequence. This operation aims to reduce the sequence length by two while keeping the shift equivariance property. For this purpose, the adaptive polyphase downsampling (APS-D) proposed in [29] is used.

To explain the APS-D operation, let us introduce $s(t)$ the convolutions output sequence of length T_0. We can then define the two sub-sequences s_0 and s_1 such that $s_0(t) = s(2t)$ and $s_1(t) = s(2t+1)$. The APS-D operation D_2^A consists in sub-sampling the sub-sequence s_l such that:

$$D_2^A(s) = s_l \quad \text{where} \quad l = \arg\max ||s_j||_1 \ (\text{for } j \in \{0, 1\})$$

Now, when the downsampling process is applied to the sequence, the length of the sequence is reduced by half. Finally, a non-linear pointwise operation is applied. In practice, this operation is the LeakyReLU.

All blocks inside the encoder have the same structure. Consider an encoder with B blocks and convolutions with Z channels. Then the initial time series is projected into the representation space on a sequence of $T^{'}(T' \approx T/2^B)$ time steps, where each point of the sequence is a vector of size Z.

b) Decoder: Now we can define the decoder blocks symmetrically to the encoder. The decoder is a function that projects a sequence of T' vectors of size Z into a reconstructed time series $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ of size T. If the encoder has B blocks, then the decoder will have B blocks. However, the structure of a decoder block is slightly different. First, an upsampling operation (adaptive polyphase upsampling) increases the length of the sequence by two. In practice, the values of the input sequence are re-used and 0's are inserted between each value to double the size of the sequence. The 0's are inserted in an even or odd manner according to the subsequence extracted in the corresponding encoder block during the downsampling phase (if l was 0 or 1). Next, a 1D non-strided convolutional layer with a kernel size of 3 is applied. Finally, a nonlinear pointwise operation is applied (except in the last block).

c) Discretization mechanism: Now that the encoder and decoder are defined, we need to specify the discretization method used within the representation space. After passing the time series to the encoder and thus obtaining a sequence of vectors (sequence of embeddings: $(\boldsymbol{e_1}, \boldsymbol{e_2}, ..., \boldsymbol{e_{T'}})$), we use the vector quantization (VQ) mechanism [11], [12]. Given a set of K centroids $\{\boldsymbol{c_j} \in \mathbb{R}^Z,\ j \in 1, ..., K\}$, VQ consists of assigning an encoder output point $\boldsymbol{e_{t'}} \in \mathbb{R}^Z$ to the nearest centroid:

$$\boldsymbol{e}^{\boldsymbol{q}}_{\boldsymbol{t'}} \leftarrow \boldsymbol{c_k} \quad \text{where} \quad k = \arg\min_j ||\boldsymbol{e_{t'}} - \boldsymbol{c_j}||_2^2. \tag{1}$$

In this way, the sequence of embedding vectors is transformed into a sequence of quantized vectors (each quantized vector has only K possible values). This mechanism is illustrated in the Figure 7. Then, the sequence of quantized vectors $(\boldsymbol{e}^{\boldsymbol{q}}_{\boldsymbol{1}}, \boldsymbol{e}^{\boldsymbol{q}}_{\boldsymbol{2}}, ..., \boldsymbol{e}^{\boldsymbol{q}}_{\boldsymbol{T'}})$ is given as input to the decoder. We describe how centroids are moving through epochs in the training subsection.

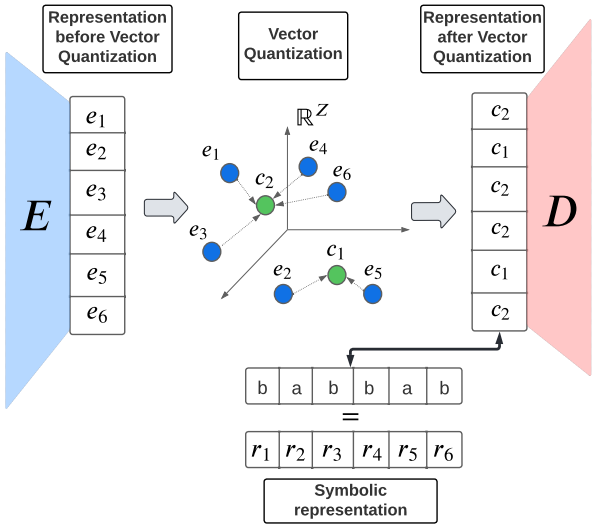

Fig. 7. Visualization of the vector quantization mechanism in the context of an auto-encoder. E and D stand respectively for Encoder and Decoder.

B. An architecture that meets the requirements

The VQ mechanism satisfies the requirement for a discrete symbolic representation. The VQ mechanism, also known as a self-organizing map [30], allows the centroids to move as iterations progress. The vector quantization mechanism assigns nearby vectors to the same centroid while maintaining temporal consistency. Thus, our architecture encourages similar patterns to be quantized to the same vector. The symbolic representation is obtained directly from the centroid indices used during vector quantization. Since the index represents the vector it characterizes, the sequence of centroid indices chosen during the vector quantization phase serves as an excellent symbolic representation of the time series (thus $Card(\mathbb{A}) = K$).

The proposed architecture uses convolution operations that guarantee the temporal consistency of the representation. We can easily compute the input that produced a given output from a convolution sequence by determining the receptive field [31]. In addition, convolutions are local operations that slide over the input, which allows them to preserve the temporal order of the input. Using a pointwise VQ mechanism does not affect the temporal consistency property of our architecture.

The symmetric and unbiased encoder/decoder design guarantees that the representation is decodable. Our loss function (eq. 2) ensures that the reconstruction converges towards the original time series. It enables the visualization of the learned representation while maintaining temporal consistency and enabling separate decoding of each representation component.

Due to the adaptive polyphase upsampling/downsampling [29], both encoder and decoder are shift equivariant. The shift equivariance properties would be lost with classical stridded convolutions [32]. This property is crucial for understanding

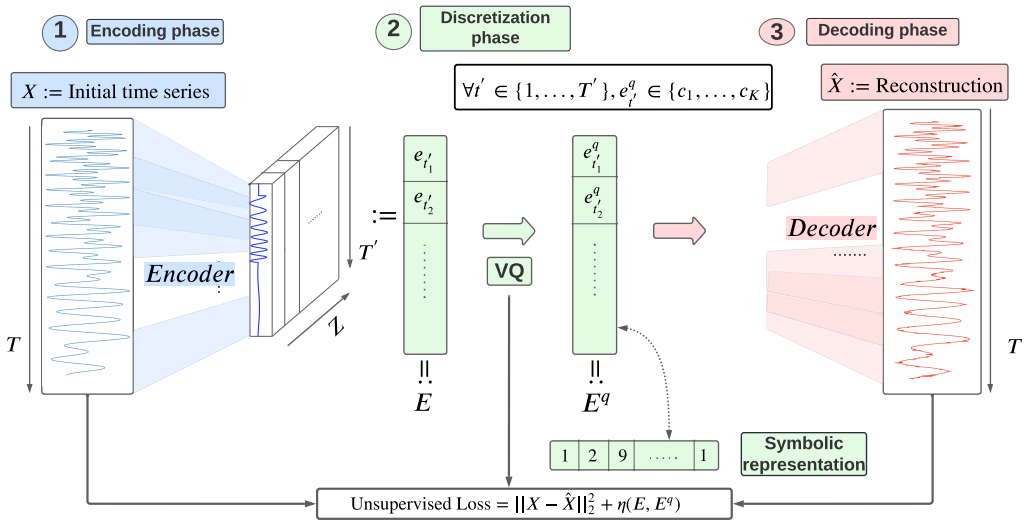

Fig. 8. Global unsupervised architecture overview. η refers the last two terms in the loss eq. 2.

the meaning of the elements in the symbolic representation. It ensures that identical symbolic elements always represent the same pattern (when there are no boundary effects), regardless of their position.

By changing the number of B blocks in the encoder (and by the symmetry in the decoder), we can adjust the information learned in the representation, enabling us to adapt the representation to different frequency levels. The higher the number of blocks, the shorter the representation sequence and the more global the information captured for an element of the representation. Thus, we define the number of blocks B as an hyperparameter of the architecture.

C. Training

The unsupervised architecture is trained using the loss (eq. 2) introduced by [12]. To simplify the notations, we present the loss for an instance $\boldsymbol{x}$. The notations remain the same as those introduced above, except for the following. The operator sg is the stop gradient operator whose derivative is zero during the backward computation time. The encoder is defined as ϕ_θ and the decoder as $\psi_{\theta'}$. In the Section III, ϕ_θ characterizes the function that maps the time series to the symbolic representation. Here ϕ_θ stands for the function that maps the time series to the embeddings (before the discretization phase). On the other hand, here, $\psi_{\theta'}$ characterizes the function that maps the quantized vector representation to the reconstruction. The temporal vector sequence after quantization is denoted $\boldsymbol{E^q} \in \mathbb{R}^{T' \times Z}$. Thus, the total loss of the unsupervised model is:

$$\underset{\theta,\theta',E}{\arg\min}\ ||\boldsymbol{x} - \psi_{\theta'}(\boldsymbol{E^q})||_2^2 + ||sg[\phi_\theta(\boldsymbol{x})] - \boldsymbol{E^q}||_2^2 + \beta||\phi_\theta(\boldsymbol{x}) - sg[\boldsymbol{E^q}]||_2^2. \quad (2)$$

The first term of the loss refers to the reconstruction ability of the model. In practice, we consider the mean square error pointwise between the real-time series and the decoder output. This part of the loss optimizes the decoder and the encoder. It's important to note that the arg min operator in eq. 1 is not differentiable at the VQ level. In [12], the authors suggest passing the gradients of the quantized vectors to their corresponding encoder vector outputs. This allows the reconstruction loss to optimize the encoder. The second term in the loss updates the used centroids $\boldsymbol{E^q}$ by moving them toward the embedding vectors $\phi_\theta(\boldsymbol{x})$ assigned to them. The centroids are initialized with a Gaussian distribution. This approach has proven to be stable in training and allows an independent choice of solver for the rest of the loss. In the model implementation, the Adam solver [33] is used for the other terms of the loss. The last part of the loss is the commitment loss and ensures that the encoder outputs do not land too far from the used centroids. It guarantees better training stability.

V. Downstream task: classification over extracted representations

Similar to dictionary methods for time series classification [18], [20], the proposed representation learning process leads to an interpretable sequential symbolic representation of the time series. The series is represented by the sequence of the indices of the quantized vectors (the symbolic representation). We propose to evaluate the obtained symbolic representation with downstream classification tasks. When classifying symbolic representations, authors commonly extract histograms of symbolic subsequences and feed them into a K-NN, or linear classifier [22], [23]. For example, in [22], authors propose an efficient algorithm for a logistic regression classifier based on coordinate gradient descent to find the most relevant subsequences. In the following, we propose a linear classifier based on the presence or absence of discriminative subsequences inspired by existing methods. This classification method preserves the interpretability provided by the representation and ensures an in-situ, global interpretable classification decision.

A. Classification using a unique symbolic representation

In our classification process, we focus only on using subsequences of lengths one and two to characterize the symbolic representation. This approach effectively captures local dynamics while maintaining a low-dimensional feature space. Let d be the cardinal of the set of possible symbolic subsequences of length one and two, and $\boldsymbol{r_i}$ be a symbolic representation (extracted using our auto-encoder) for individual i. We can extract for the representation $\boldsymbol{r_i}$, the vector $\boldsymbol{h_i}$ which indicates if a subsequence (of length one or two) is present in $\boldsymbol{r_i}$. Thus, $\boldsymbol{h_i}$ is a vector of size d composed of 0 and 1 elements ($\boldsymbol{h_i} \in \{0,1\}^d$). The features vectors $\{\boldsymbol{h_i}, i \in \{1 \dots N\}\}$ representing the training set can be used after that to solve the classification problem using logistic regression (eq. 3):

$$\arg\min_{w,b} \frac{1-\rho}{2}\boldsymbol{w^T w} + \rho\|\boldsymbol{w}\|_1 + \lambda \sum_{i=1}^{n} \log\left(\exp\left(-y_i\left(\boldsymbol{h_i}^T\boldsymbol{w} + b\right)\right) + 1\right) \quad (3)$$

with λ the regularization parameter, ρ the trade-off between the ℓ_1 penalty and the ℓ_2 penalty and b the bias. Sparsity in interpretable classification models is desirable to more easily understand the decision [34], [35]. In the case of multiclass classification, the usual One vs All method [36] is applied to provide better interpretability.

However, classification using a unique representation (e.g., a representation where the encoder/decoder is composed of B blocks) does not allow the use of features of different frequencies. Therefore, it is desirable to train several representations corresponding to different temporal dimension reductions (deeper or shallower architectures) in order to classify using different frequency features.

B. Classification using multiple symbolic representations

According to Subsection IV-A the depth of the architecture is directly related to the number of blocks B in the encoder (and by symmetry in the decoder). We train the proposed unsupervised model for D different depths ($B \in \{1, 2, ..., D\}$). Thus, we get D representations of different length, which capture different features ($\boldsymbol{h_i^1}, \boldsymbol{h_i^2}, ..., \boldsymbol{h_i^D}$). Then we apply the following steps:

1) A penalized logistic regression (eq. 3) is performed separately on each extracted features vector $\boldsymbol{h_i}$.
2) For each $\boldsymbol{h_i}$, we recover the features whose logistic regression coefficients have non-zero values. Thus, we obtain an aggregated discriminative set of features for the D representations.
3) We perform a final penalized logistic regression on the obtained set.

It is important to highlight two observations. Firstly, this classification method retains interpretability because each feature is interpretable by the construction of the representation, making it straightforward to interpret the coefficients of the logistic regression associated with a feature. Secondly, the number of representations we use corresponds to the number of times the initial time series can be halved while being greater than a threshold (T' should be greater than this threshold). In practice, the number of representations we use for the UCR datasets rarely exceeds five. Keeping the number of representations small is essential for preserving interpretability. In particular, a large number of representations has a strong negative impact on interpretability [8].

VI. Experiments

In our experiments, the unsupervised architectures are trained using only the training set. The symbolic test representations are constructed by passing the test data through the trained architecture. We also set the number of available centroids to 32 ($K = 32$) and the dimension of the latent space is 64 ($Z = 64$). For training the unsupervised models, we set β to 0.25 in the loss (eq. 2) as realized in [12].

A. Quantitative experiments

The quantitative experiments are performed on 25 selected datasets from the UCR archive [37]. These datasets meet specific criteria: variety of application types (sensor, motion, image, device), a minimum of 50 training and 50 test instances, and a maximum of seven classes. Table I presents the results of our model compared to the interpretable in-situ methods SAX SEQL, SAX VSM, FS and LTS presented in Section II. We also compare our accuracy results to the well-known K-NN classifier coupled with Dynamical Time Warping (DTW) distance [38]. This classifier is sometimes presented as interpretable, but it does not allow the extraction of localizable discriminative features. We use the accuracy metrics from [23] for comparison. Referring to Table I, our method gives better results than the other in-situ methods interpretable on average on these datasets.

Our method results in a minimum 2.2 percentage points increase compared to other interpretable in-situ methods. Even if our method does not come first in all cases, the quantitative results are very promising for two main reasons. First, only a small subpart of all possible symbolic subsequences are used as our method aims to enforce sparsity in order to favour interpretability. Secondly, our representations are learned without any supervision and thus are not guided by the underlined task.

In addition, we evaluate the accuracy of logistic regression on our symbolic neural representation against SVM on top of neural representation. This framework is often used to evaluate the best unsupervised neural representation when the neural representation is a simple vector [2], [4]. Previous research has found that the best-unsupervised methods with this framework are TS2Vec [4] and T-Loss [2]. We compare these two methods with our own on the 25 previous datasets and find that all three perform similarly in accuracy (0.789 ± 0.15 for T-Loss and 0.807 ± 0.15 for TS2Vec compared to 0.793 ± 0.13 for our method). These results suggest that the constraints we impose on our architecture to satisfy the interpretability requirement do not significantly deteriorate the expressiveness of the unsupervised representation.

TABLE I
ACCURACY ON 25 UCR DATASETS COMPARE TO IN-SITU INTERPRETABLE METHODS. THE BEST RESULTS ARE IN BOLD AND THE SECOND BEST RESULTS ARE UNDERLINED.

Datasets	Ours	SAX SEQL	SAX VSM	FS	LTS	DTW CV
Coffee	0.964	**1.000**	0.929	0.929	**1.000**	**1.000**
Computers	**0.728**	0.676	0.620	0.500	0.584	0.620
DistalPhalanxOAG	0.755	0.818	**0.842**	0.655	0.779	0.626
DistalPhalanxOC	0.732	0.718	0.728	**0.750**	0.719	0.725
DistalPhalanxTW	0.640	**0.748**	0.604	0.626	0.626	0.633
Earthquakes	0.734	**0.789**	0.748	0.705	0.741	0.727
ECG5000	**0.932**	0.924	0.910	0.923	**0.932**	0.925
FordA	0.883	0.851	0.827	0.787	**0.957**	0.691
GunPoint	0.940	0.987	0.987	0.947	**1.000**	0.913
Ham	0.705	0.705	**0.810**	0.648	0.667	0.600
Herring	**0.656**	0.578	0.625	0.531	0.625	0.531
ItalyPowerDemand	0.906	0.734	0.816	0.917	**0.970**	0.955
LargeKitchenApp	0.864	0.760	**0.877**	0.560	0.701	0.795
PhalangesOC	0.748	0.717	0.710	0.744	**0.765**	0.761
ProximalPhalanxOC	0.818	0.818	0.828	0.804	**0.834**	0.790
ProximalPhalanxOAG	0.839	0.844	0.824	0.780	**0.849**	0.785
ProximalPhalanxTW	0.771	**0.792**	0.610	0.702	0.776	0.756
RefrigerationDevices	0.533	0.541	**0.653**	0.333	0.515	0.440
ScreenType	0.499	0.461	**0.512**	0.413	0.429	0.411
ShapeletSim	0.994	0.994	0.717	**1.000**	0.950	0.700
SmallKitchenApp	**0.795**	0.776	0.579	0.333	0.664	0.672
Strawberry	**0.962**	0.954	0.957	0.903	0.911	0.946
Wafer	0.975	0.993	**0.999**	0.997	0.996	0.995
Wine	0.759	0.556	**0.963**	0.759	0.500	0.611
Worms	**0.714**	0.536	0.558	0.649	0.610	0.532
Mean	**0.793**	0.770	0.769	0.715	0.764	0.725

B. Qualitative experiment

Among the 25 previous UCR datasets, not all are suitable for interpretability because a limited number of features cannot discriminate between the classes. In most papers dealing with in-situ interpretability for time series, the commonly used datasets are GunPoint, ShapeletSim, or Coffee datasets [21], [22]. For the qualitative analysis, we decide to focus on the GunPoint and ShapeletSim datasets [37].

1) GunPoint dataset: The GunPoint dataset consists of two actors performing a movement with their right hand, with two classes: Gun-Draw (class 0) and Point (class 1).

a) Representations: The time series are z-normalized for each instance. We train five models with different architecture depths to capture different features in the time series. The architectures is only trained using the train time series, then a simple forward pass is used to get the representation for each time series in the test dataset. After training, we obtain five representations of different lengths. As shows in Table II, the representation generalizes easily to the test.

b) Classification: For each trained representation $\boldsymbol{r_B}$ we construct the binary vector $\boldsymbol{h^B}$ ($B \in \{1, 2, 3, 4, 5\}$). We then fit a logistic regression for each representation separately. For each logistic regression (eq. 3), ρ and λ are found by cross-validation [39]. We encourage a strong penalty ℓ_1 to set to zero the coefficients for non-discriminating features for each representation. Table III shows for each logistic regression the initial number of features and the number of features whose regression coefficient is different from zero.

TABLE II
INFORMATION ON THE DIFFERENT REPRESENTATIONS LEARNED FOR THE *GunPoint* DATASET.

Depth	**Temporal downscaling**	**Symbolic representation length**	**Pointwise train MAE**	**Pointwise test MAE**
$B = 1$	$2^1 = 2$	75	0.044	0.045
$B = 2$	$2^2 = 4$	38	0.038	0.045
$B = 3$	$2^3 = 8$	19	0.032	0.047
$B = 4$	$2^4 = 16$	10	0.027	0.085
$B = 5$	$2^5 = 32$	5	0.023	0.085

TABLE III
INFORMATION ON LOGISTIC REGRESSION FOR EACH REPRESENTATION (STEP 1 IN PROCESS V-B).

Depth	$\boldsymbol{h^B}$ **size**	**Number of features actually used to classify**
$B = 1$	115	23
$B = 2$	203	26
$B = 3$	293	24
$B = 4$	328	26
$B = 5$	187	11

Then, we fit the final logistic regression on each extracted feature (whose regression coefficients differ from 0). The final logistic regression performs on 110 features. After training, 89 coefficients are set to zero in this regression because of the penalty effect. Initially, the concatenation of feature vectors is a vector of size 1126, but only 21 features are ultimately used for the classification problem.

The test accuracy is 0.94, and the train accuracy is 1. Now that the final logistic regression is fitted, we can look at the coefficients of this regression. Table IV shows the most critical features (whose relative importance, their absolute value divided by the value of the others, is greater than 5 percent).

TABLE IV
DETAILS OF THE MOST DISCRIMINATIVE FEATURES IN THE FINAL LOGISTIC REGRESSION (STEP 3 IN PROCESS V-B), INTERCEPT IS 5.13.

Depth	**Symbolic subsequence**	**Logistic regression coefficients**	**Relative importance**
$B = 1$	bb	- 0.92	7.0 %
$B = 1$	fc	- 0.93	7.2 %
$B = 1$	hk	- 1.20	9.2 %
$B = 1$	kg	- 0.98	7.4 %
$B = 1$	kh	- 0.76	5.8 %
$B = 3$	Fx	1.23	9.4 %

Our representation is interpretable and allows us to decode symbolic subsequences, so we can use the extracted features and logistic regression coefficients to gain insight into the problem. We can interpret the classification decision at the global level as well as at the local level.

c) Global interpretability: It consists in visualizing which feature at the model level allows the classification of the time series correctly. With our architecture, decoding the discriminative symbolic subsequences suffices to understand what the unsupervised model learned, and visualize the recon-

structed subseries. Let us consider the symbolic sub-sequence *'Fx'* (for depth $B = 3$ in Table IV). This sub-sequence is the most discriminative sub-sequence for class one. When we decode this subsequence in Figure 9, we obtain a subseries that characterizes the way the finger is raised. This subseries differs from the way the gun is raised.

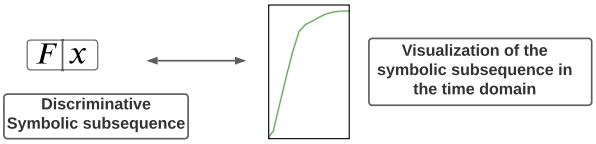

Fig. 9. Global interpretability decision. Visualization of the most discriminative symbolic subsequence for class 1 (Point).

d) Local interpretability: It consists in visualizing for an instance the regions of the time series that make the decision. For example, take the symbolic subsequence *'hk'* (for depth $B = 1$ in Table IV), which is the most discriminating subsequence for class 0. Figure 10 presents an instance whose representation ($B = 1$) contains the subsequence *'hk'*. Then, we highlight in red the pre-image of this subsequence using the receptive fields of the convolutions.

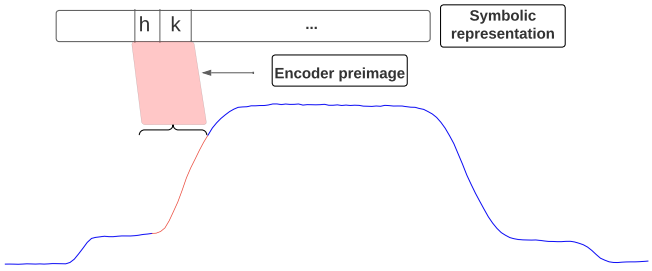

Fig. 10. Local interpretability for a class 0 (Gun) instance.

2) ShapeletSim dataset: The ShapeletSim dataset comprises two classes, class 0 is purely white noise, and class 1 includes a triangle shape at a random location. The classification problem is easily interpretable because only the presence or absence of a triangle characterizes the difference between the two classes.

As in the previous use case, we construct the different symbolic representations in an unsupervised manner and then extract the discriminative features. The highest coefficient in the final logistic regression is associated with the symbolic subsequence *'wjdddjw'*. Figure 11 presents visualization for global interpretable classification decision. Using the decoder, we decoded the subsequence *'wjdddjw'* and examined the resulting decoded shape. We observed that the decoded shape matches the vertex of the decoded triangle. It is worth noting that this demonstration of global interpretability does not require any specific instance for visualization. On the other hand, Figure 12 shows a visualization of the local interpretability for the symbolic subsequence *'wjdddjw'* for a given sample. We retrieve the triangular shape by computing the pre-image of the discriminating subsequence for this given sample.

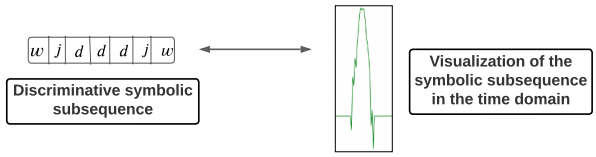

Fig. 11. Global interpratibility decision. Visualization of the most discriminative symbolic subsequence for class 1 (includes triangle).

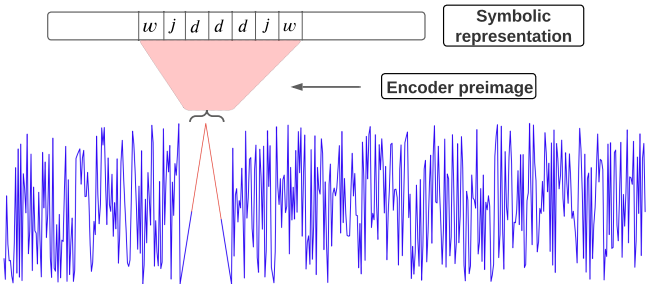

Fig. 12. Local interpretability for a class 1 (includes triangle) instance.

In the analysis of the ShapeletSim problem, we recognize its inherent simplicity, as indicated by the high test accuracy of 0.994 shown in Table I when identifying the discriminant feature. However, it is an interesting dataset to visualize interpretability performance.

VII. Conclusion

We first present essential requirements for building an interpretable neural representation for time series and then present an architecture that satisfies these requirements. The proposed unsupervised symbolic neural model fills a gap between symbolic and neural representations for time series. It has the advantage of allowing global interpretability of downstream classification tasks, while guaranteeing high expressiveness and good performance. The constructed representation has been evaluated both qualitatively and quantitatively on classification tasks. We show promising results for accuracy compared to both in-situ interpretable and neural methods. Additionally, the proposed interpretability provides an understanding of the model's classification decisions at both global and local levels for a broad range of time series. Much of our work has been devoted to the study of an unsupervised and interpretable neural architecture for time series. The choice of reconstruction loss in unsupervised architecture seems interesting for future work. Furthermore, coordinate descent, as described in [40], may improve the accuracy of these models by identifying longer symbolic subsequences that help distinguish between classes.

VIII. Acknowledgment

We would like to thank Tahar Nabil for the valuable discussions on this project.

REFERENCES

[1] Q. Ma, J. Zheng, S. Li, and G. W. Cottrell, "Learning representations for time series clustering," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 3776–3786.

[2] J. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 4652–4663.

[3] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pretrained deep recurrent neural network for time series classification," in *25th European Symposium on Artificial Neural Networks, ESANN 2017, Bruges, Belgium, April 26-28, 2017*, 2017.

[4] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "Ts2vec: Towards universal representation of time series," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI*. AAAI Press, 2022, pp. 8980–8987.

[5] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2021, pp. 2114–2124.

[6] Y. Bengio, A. C. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[7] X. Zhang, Z. Zhao, T. Tsiligkaridis, and M. Zitnik, "Self-supervised contrastive pre-training for time series via time-frequency consistency," *CoRR*, vol. abs/2206.08496, 2022.

[8] Z. C. Lipton, "The mythos of model interpretability," *Commun. ACM*, vol. 61, no. 10, pp. 36–43, 2018.

[9] Y. Wang, "Interpretable time series classification. (classification interprétable de séries temporelles)," Ph.D. dissertation, University of Rennes 1, France, 2021.

[10] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nat. Mach. Intell.*, vol. 1, no. 5, pp. 206–215, 2019.

[11] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, ser. The Kluwer international series in engineering and computer science. Kluwer, 1991, vol. 159.

[12] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 6306–6315.

[13] V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch, "SOM-VAE: interpretable discrete representation learning on time series," in *7th International Conference on Learning Representations, ICLR*, 2019.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR*, 2013.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5998–6008.

[16] Y. Li, Z. Chen, D. Zha, M. Du, J. Ni, D. Zhang, H. Chen, and X. Hu, "Towards learning disentangled representations for time series," in *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2022, pp. 3270–3278.

[17] Y. Luo, C. Xu, Y. Liu, W. Liu, S. Zheng, and J. Bian, "Learning differential operators for interpretable time series modeling," in *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2022, pp. 1192–1201.

[18] J. Lin, E. J. Keogh, S. Lonardi, and B. Y. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003, pp. 2–11.

[19] J. Lin, E. J. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: a novel symbolic representation of time series," *Data Min. Knowl. Discov.*, vol. 15, no. 2, pp. 107–144, 2007.

[20] P. Schäfer and M. Högqvist, "SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets," in *15th International Conference on Extending Database Technology, EDBT*. ACM, 2012, pp. 516–527.

[21] P. Senin and S. Malinchik, "SAX-VSM: interpretable time series classification using SAX and vector space model," in *2013 IEEE 13th International Conference on Data Mining*. IEEE Computer Society, 2013, pp. 1175–1180.

[22] T. L. Nguyen, S. Gsponer, and G. Ifrim, "Time series classification by sequence learning in all-subsequence space," in *33rd IEEE International Conference on Data Engineering, ICDE*. IEEE Computer Society, 2017.

[23] T. L. Nguyen, S. Gsponer, I. Ilie, M. O'Reilly, and G. Ifrim, "Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations," *Data Min. Knowl. Discov.*, vol. 33, no. 4, pp. 1183–1222, 2019.

[24] L. Ye and E. J. Keogh, "Time series shapelets: a novel technique that allows accurate, interpretable and fast classification," *Data Min. Knowl. Discov.*, vol. 22, no. 1-2, pp. 149–182, 2011.

[25] E. J. Keogh and T. Rakthanmanon, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *Proceedings of the 13th SIAM International Conference on Data Mining*. SIAM, 2013, pp. 668–676.

[26] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*. ACM, 2014, pp. 392–401.

[27] Y. Wang, R. Emonet, É. Fromont, S. Malinowski, and R. Tavenard, "Adversarial regularization for explainable-by-design time series classification," in *32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI*. IEEE, 2020, pp. 1079–1087.

[28] P. Schäfer, "The BOSS is concerned with time series classification in the presence of noise," *Data Min. Knowl. Discov.*, vol. 29, no. 6, pp. 1505–1530, 2015.

[29] A. Chaman and I. Dokmanic, "Truly shift-equivariant convolutional neural networks with adaptive polyphase upsampling," in *55th Asilomar Conference on Signals, Systems, and Computers, ACSSC*. IEEE, 2021, pp. 1113–1120.

[30] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[31] A. Araujo, W. Norris, and J. Sim, "Computing receptive fields of convolutional neural networks," *Distill*, vol. 4, no. 11, p. e21, 2019.

[32] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *Proceedings of the 33nd International Conference on Machine Learning, ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48, 2016, pp. 2990–2999.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, 2015.

[34] M. Wu, M. C. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez, "Beyond sparsity: Tree regularization of deep models for interpretability," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence,*. AAAI Press, 2018, pp. 1670–1678.

[35] Y. Li, S. Lin, B. Zhang, J. Liu, D. S. Doermann, Y. Wu, F. Huang, and R. Ji, "Exploiting kernel sparsity and entropy for interpretable CNN compression," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 2800–2809.

[36] R. M. Rifkin and A. Klautau, "In defense of one-vs-all classification," *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, 2004.

[37] H. A. Dau, A. J. Bagnall, K. Kamgar, C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. J. Keogh, "The UCR time series archive," *IEEE CAA J. Autom. Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.

[38] T. Rakthanmanon, B. J. L. Campana, A. Mueen, G. E. A. P. A. Batista, M. B. Westover, Q. Zhu, J. Zakaria, and E. J. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*. ACM, 2012, pp. 262–270.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[40] G. Ifrim and C. Wiuf, "Bounded coordinate-descent for biological sequence classification in high dimensional predictor space," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 708–716.

APPENDIX A
HOW TO COMPUTE RECEPTIVES FIELDS REGIONS

With our model, it is very useful to be able to calculate the receptive fields and in particular the receptive field regions relative to an element (or a region) of the representation. To do this, we just need to adapt the following formulas computed in [31] to our architecture:

$$v_0 = v_L \prod_{i=1}^{L} s_i - \sum_{l=1}^{L} (1 + p_l - k_l) \prod_{i=1}^{l-1} s_i$$

$$u_0 = u_L \prod_{i=1}^{L} s_i - \sum_{l=1}^{L} p_l \prod_{i=1}^{l-1} s_i$$

Where:

- v_0 stands for the left-most coordinates of the receptive field in the intial time series
- u_0 stands for the right-most coordinates of the receptive field in the intial time series
- v_L stands for the left-most coordinates in the representation
- u_L stands for the left-right coordinates in the representation
- k_l stands for the kernel size at depth l
- s_l stands for the stride at depth l
- p_l stands for the padding at depth l
- L stands for the depth of the network

Thanks to these operations, we can highlight the receptive fields of the elements of the representation in Figure 12 and Figure 10.

APPENDIX B
IMPACT ON ACCURACY RESULTS OF THE NUMBER OF AVAILABLE CENTROIDS (K)

The quantitive experiments in Subsection VI-A were performed for 32 centroids available during the vector quantization (K=32). It is interesting to see how the number of centroids affects the classification performance. Table V shows the accuracy results averaged over the 25 UCR datasets on which the experiments were conducted.

TABLE V
MEAN ACCURACY ON THE PREVIOUS 25 UCR DATASETS FOR DIFFERENT K ($k \in \{8, 16, 32\}$). THE BEST RESULT ARE IN BOLD AND THE SECOND BEST RESULT ARE UNDERLINED.

	$k = 8$	$k = 16$	$k = 32$
Mean accuracy	0.753	<u>0.778</u>	**0.793**

We observed that increasing the number of available centroids improves accuracy. However, it harms the centroids' expressiveness and, thus, the interpretability of the representation.

APPENDIX C
PROOF OF THE SHIFT EQUIVARIANCE PROPERTY

In this appendix, we propose to describe how the adaptive polyphase downsampling/upsampling mechanism preserves the shift equivariance property despite downsampling (or upsampling). To do so, we pick up the proof given in [29]. We first present the proof of shift equivariance for the adaptive polyphase downsampling mechanism and then the adaptive polyphase upsampling case is straightforward.

a) Definition:: We define the shift equivariance property as follows. Let S be an element of a sequence (a single element or a subsequence), and G_T the group of discrete translations along the temporal axis. If we take τ to be any discrete translation in G_T and f to be a function equivariant by discrete translation for G_T, then there exists $\tau' \in G_{T'}$ such that: $f(\tau(S)) = \tau'(f(S))$.

b) Case 1: Adapative polyphase downsampling (APS-D) for downsampling by half: : To explain the APS-D operation, let us introduce $s(t)$ the convolutions output sequence of length T_0. We can then define the two sub-sequences s_0 and s_1 such that $s_0(t) = s(2t)$ and $s_1(t) = s(2t+1)$. The APS-D operation D_2^A consists in sub-sampling the sub-sequence s_l such that:

$$D_2^A(s) = s_l \quad \text{where} \quad l = \arg\max ||s_j||_1 \text{ (for } j \in \{0, 1\})$$

We introduce the following notations:

- τ_k is a discrete translation by k
- $s_{APS} = D_2^A(s)$ is the non shift sequence output
- $s_{APS}^{(k)} = D_2^A(\tau_k(s))$ is the shifted sequence output

Then we have:

$$s_{APS}^{(k)} = \begin{cases} \tau_{\frac{k}{2}}(s_{APS}), \text{when } k \text{ is even} \\ \tau_{\frac{k+2i-1}{2}}(s_{APS}), \text{when } k \text{ is odd} \end{cases}$$

c) Case 2: Adapative polyphase usampling (APS-U) for upsampling by two:: For the APS-U, the s signal is upsampled by two. We insert 0 elements in each even or odd position according to the phase chosen in the block of the corresponding encoder. By construction, this operation is shift equivariant. For more details on this operation, see the original paper [29].