Received 23 January 2023, accepted 13 February 2023, date of publication 16 February 2023, date of current version 23 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3246159

RESEARCH ARTICLE

A Granularity Invariant Method for the Classification of Energy Production Time Series in Photovoltaic Plants

IVÁN DE-PAZ-CENTENO[1,2], MARÍA TERESA GARCÍA-ORDÁS[3], ÓSCAR GARCÍA-OLALLA[2], CARLOS GIRÓN-CASARES[1], AND HÉCTOR ALAIZ-MORETÓN[3]

[1]Department of Artificial Intelligence, SMARKIA ENERGY SL, 24002 León, Spain
[2]Department of Electric, Systems and Automatic Engineering, Escuela de Ingenierías Industrial e Informática, Universidad de León, 24071 León, Spain
[3]SECOMUCI Research Group, Escuela de Ingenierías Industrial e Informática, Universidad de León, 24071 León, Spain

Corresponding author: Iván de-Paz-Centeno (ivan.depaz@smarkia.com)

This work was supported by the Ministry of Science and Innovation Department of the Government of Spain, under the Industrial Doctorate Program, within the company SMARKIA ENERGY SL and the University of León, under Grant DIN2018-009733.

ABSTRACT The treatment of photovoltaic power production time series often faces the challenge of unifying the granularity of the series when generating a predictive model. This can limit the generation of a dataset in terms of the time covered and the number of examples. In addition, models built with data of static granularities tend to show rigidity when facing granularity variations, invalidating them for scenarios different from those of the data they were trained on. To address this issue, this paper presents a novel method specifically indicated for Deep-Learning models that shows invariance to granularity called Synthesis. This operation can be added as a layer to an artificial neural network, allowing it to be applied to any power production time series and synthesizing the content of an arbitrarily long time series into a fixed-size vector which can be used for classification or regression regardless of the initial time series length. The experiments with the NIST Campus Photovoltaic dataset demonstrate the effectiveness of the method, showing an F1-Score of 1.0 for the classification of series with granularities between 2 minutes and 2 hours, and an F-Score of 1.0 for the classification of time series with variations of granularity throughout time when training with 5-minute granularity samples.

INDEX TERMS PV production, classification, time series, invariance, granularity, deep-learning.

I. INTRODUCTION

The power production of photovoltaic plants lacks a well-defined standard in terms of the granularity with which the time series are stored and/or communicated, making a multitude of them expressed in different granularities depending on the device and/or the Energy Management System (EMS) that governs it. Among the different possible granularities with which the power production time series of photovoltaic plants can be expressed, the hourly, quarter-hourly, 5-minute, and minute granularities stand out; although others can be found without being limited to a fixed granularity. As stated in [1], it is common to find studies on energy forecasting in various granularities, with the most commonly used granularity being hourly. In addition, it is also possible to find power production time series whose granularity varies over time, for example due to an improvement in the equipment that monitors energy production.

The creation of a model that allows for prediction on a power production series typically requires the synthesis of a training dataset in which the granularities are well defined and unified. This usually implies preprocessing the data to transform them to the most restrictive granularity through a down-sampling operation, or the discarding of more restrictive granularities from the data set. In addition, models trained with datasets whose granularities are unified tend to acquire rigidity in terms of the granularity accepted in the input, either because the receptive window of the models is limited to a

The associate editor coordinating the review of this manuscript and approving it for publication was Li He.

specific input size or because the models lack of any mechanism for the treatment of series with different granularities than those observed during training.

To support variations in granularities and avoid input rigidity when different granularities are provided, this paper proposes a mechanism that can be incorporated into an artificial intelligence model, which is especially indicated for Deep-Learning models. This mechanism is defined as a network layer that is able to provide invariance to the granularity of the input data. The mechanism consists of an operation called *Synthesis* that allows a model trained with a power production time series of any granularity to acquire, after training, the ability to make predictions from inputs with different granularities or even with granularity variations within the same time series. In addition, the method allows the input, which may have a variable length, to be compressed into a fixed size vector, enabling the classification and/or regression of the entire time series regardless of its size and granularity.

To demonstrate the proposed method and the mentioned invariance property, several experiments were conducted using the NIST Campus Photovoltaic dataset, which is a public dataset of power production data. In the experiments, a model based on a Neural Network (NN) was trained to classify one month of power production into its corresponding numerical month. This work demonstrates that the network is able to learn to discriminate the class to which each monthly production corresponds despite being expressed in different granularities, even when the time series itself presents sections expressed in several different granularities. Finally, the proposed model is evaluated by simulating different scenarios of extreme granularities and providing results that validate the invariance of the developed method.

The contributions of the paper are enumerated as follows:

- Definition of the *Synthesis* operation to map a variable-size time series into a fixed-size vector. This operation provides invariance to granularity.
- Definition and demonstration of a network architecture to classify variable-length power production time series using the *Synthesis* operation.
- Experimentation and evaluation of the invariance to granularity provided by the method with the public photovoltaic power production dataset NIST Campus Photovoltaic [2]. This experimentation validates that the method is able to work with different granularities after trained, allowing variations even in sections of the same time series.

The paper is structured with an introduction, related work, and justification of the novelty provided in section I; a methodology explaining the method, the network architecture, and the error function in section II; an experimentation explaining the dataset and metrics used, the configuration of the experimentation and the results in section III; a discussion in section IV and conclusions in section V.

A. RELATED WORK

Dealing with different granularities is a common issue that most works with time series address, since it involves variations in the length that condition the creation of a Machine Learning model. It is common to find adaptations of the series to the most restrictive granularity, forcing the input to have a specific granularity. In [3] they propose to perform downsampling of the resolution of the MERIS time series to the most restrictive format of Landsat-like spatial resolution measurements in order to unify the monitoring of heterogeneous landscapes and generate two vegetation indices. Other procedures, such as time-based feature extraction, can also be carried out, as in the work of [4] in which it is proposed the use of an ensemble of different features operated in the temporal domain of the time series to achieve a classification of time series that surpasses Dynamic Time Warping (DTW), with a warping window configured through cross validation.

In [5], a Convolutional Neural Network (CNN) is proposed to predict a limited window of forecast in the future from a limited window of the past, based on a 15-minute dataset. The NN is connected to a signal decomposition using wavelets in order to support variations in the length of the time series; however, their model is highly coupled to the granularity of the time series with which it was trained and therefore cannot generalize well to other granularities. According to [6], NNs are limited in the treatment of variable-length time series and their input must be constrained to a fixed length. To address this issue, an alternative method is proposed that enables the treatment of time series of different lengths through the extraction of symbolic features Symbolic Aggregate approXimation (SAX) to perform a classification. To solve the problem of different resolutions, the authors propose the extraction of multiple symbolic features at different resolutions, combining them, and training a linear model for their classification. In [7], the authors carry out a comprehensive study of non-linear methods for the treatment of time series to identify the most effective techniques. Their analysis covers recurrent networks, visibility graphs, and Markov chain-based networks, among others, stating that the use of recurrency is a possible alternative for the treatment of series of multiple granularities. In the work of [8] a combination of several recurrent neural network methods, as an ensemble, is proposed for forecasting the Algerian Market. Although recurrent networks can accept granularity variations, their accuracy depends on them being part of the training. As reported in [9], the authors suggest using a Long Short-Term Memory (LSTM) trained with wavelet decomposition using Discrete Wavelet Transform (DWT) and the Daubechies-20 wavelet for multiresolution analysis using stationary wavelet transform. This wavelet has 20 coefficients and ensures invariance to displacement. However, the trained model is closely tied to the wavelets generated for the granularity of the original series and does not generalize well to other granularities unless it is re-trained. In [10], the authors present a method for predicting energy demand based on a

CNN that incorporates timestep and weather information as extra features. However, the model is highly dependent on the granularity used in the training set. In the work of [11], a CNN with four inputs of different kernel sizes is proposed to support more input granularities, however the receptive field of each one is static and limits the ability to acquire different representations of granularity variations. In [12], an unsupervised framework called Temporal Neighborhood Coding (TNC) is proposed for learning multivariate and non-stationary complex time series representations. This representation can be used for supervised learning in classification tasks, and also for clustering algorithms.

Other methods for condensing time series into fixed-size vectors were developed based on self-attention, such as the work of [13] where a LSTM network with attention is proposed for machine reading, the work of [14] where a self-attention structure for sentence embedding is proposed, the work of [15] an attention model for Natural Language Processing (NLP) is proposed and [16] where a deep reinforced model is proposed for abstractive summarization. The work of [17] is especially relevant for proposing the use of self-attention as a standalone and unique mechanism instead of recurrence, which gave rise to the transformer architecture in NLP tasks. Later, in [18], it is proposed to use the encoder of a transformer architecture to perform unsupervised learning of features that constitute a representation of a time series and that can be used for regression and classification tasks. Although transformer-based architectures can capture the dynamism of time series and are well suited for representing them, the periodicity with which patterns repeat in time series is a factor that must be modeled in the problem and may not be successfully automatically captured by the originally proposed positional embeddings for the transformer architecture in [17], requiring fine-tuning and manual adjustment of the positional embeddings. A proper description of positional embeddings in these architectures can help generate some degree of granularity invariance and improve prediction results. This is the case of the work done in [19] where the effectiveness of the transformer architecture in time series forecasting is questioned, as comparisons with linear models reveal the difficulty of the models in making accurate predictions. However, attention-based models heavily rely on positional embeddings that provide knowledge of the order and structure of the time series, and the effectiveness of these models greatly depends on the proper selection of these positional embeddings, as shown in the work of [20]. In this regard, various solutions have emerged for describing time in a time series, such as the Time2Vec method proposed in [21] that captures periodicity through learnable parameters attached to sinusoidal functions. This allows the model to automatically capture periodicity by expressing it through a scalable sinusoidal function at the frequency of the periodic pattern. The Time2Vec variant proposed in [22] also adds an extension of its learnable parameters, which allows for the generation of positional embeddings with greater representation of the time's seasonality. However, although Time2Vec and its variants can adapt to the periodicity of the initial series, their learnable parameters are adjusted with the initial training data that allows them to identify the repetition frequency of their patterns during the training process, implying initial granularity coupling by the model. In [23], the authors attempt to unite Time2Vec with transformers by using Stationary Wavelet Transform (SWT) as a preprocessing technique for extracting wavelets corresponding to signals of different frequencies and using them as input features in a deep transformer-based architecture that allows forecasting by predicting the sub-band of the next wavelet. They rely on the Time2Vec model for its temporal scale invariance in the first layers of their model. However, the initial granularity of the time series training is also coupled to the model. On the other hand, in [24], they propose a framework called TS2Vec (TS2Vec) for compressing time series into a fixed-size vector using CNNs with temporal dilations and the use of contrastive losses at different granularities at the same time. In the work of [25], a method called Pyraformer based on attention for forecasting a time series is proposed, which allows obtaining a temporal representation of the input that captures long-range temporal dependencies from a time series that can be expressed in different granularities simultaneously. However, their method does not take into account that there may be a single representation of the time series whose granularity varies over time.

Other attempts to support multiple resolutions in other fields of study have been carried out, such as in the work of [26] in which they treat the learning of resolution-invariant representations in images. In this work, they propose a CNN that uses a Global Average Pooling (GAP) layer to compress any resolution into a fixed-size vector, which allows them to classify the content. On the other hand, GAP is an operation that produces information loss as the resolution increases, because it must average all the input axes and has no hyperparameters that can regulate this loss. In addition, their method requires training with a multitude of examples of different resolutions simultaneously to achieve resolution invariance.

Unlike the examples explored above, the method presented in this paper does not require an adaptation of the time series to a specific resolution nor any additional feature extraction from the time series. Instead, the proposed method performs a synthesis operation that allows for trainable model parameters to extract relevant features for classification. In contrast to all the referenced methods, the method applied to a time series classification or prediction model based on the *Synthesis* operation allows the resulting model to be applied in many scenarios not initially covered during training with regard to granularity variations, allowing the model to achieve greater generalization than other methods after training.

B. PROBLEM FORMULATION

Given a temporal series $\boldsymbol{S}$ expressed in $\boldsymbol{T}$ timesteps with granularity $\Delta\boldsymbol{T} = \boldsymbol{T}_{n+1} - \boldsymbol{T}_n$, it is desired to perform a value prediction operation that is invariant to the number of timesteps $\#\boldsymbol{T}$ and to the granularity of the temporal series $\Delta\boldsymbol{T}$. To do this, a function $\mathrm{f}(\boldsymbol{S})$ must be found that satisfies $\boldsymbol{P} = \mathrm{f}(\boldsymbol{S})$ independently of the value of $\#\boldsymbol{T}$ and allowing any value of $\Delta\boldsymbol{T}$, which implies that $\mathrm{f}(\boldsymbol{S})$ must predict the same $\boldsymbol{P}$ for the temporal series $\boldsymbol{S}$ expressed with any granularity. In this work, a new function $\mathrm{f}(\boldsymbol{S})$ called *Synthesis* is proposed, which is derivable and can be embedded in the input of any Machine Learning model to provide invariance to granularity and to the size of the temporal series in a regression and/or classification prediction.

C. METHOD NOVELTY

The proposed method of *Synthesis* has two possible interpretations that lie in its novelty. First, it can be interpreted as a variation of self-attention similar to those used by [13], [14], [15], [16], and [17] in which the queries $\boldsymbol{Q}$ are predefined, both in quantity and in value, to cover the entire time series. It also presents a modification in the formulation so that each element of $\boldsymbol{Q}$ can increase or decrease the number of contiguous keys $\boldsymbol{K}$ with which they have similarity through the dot-product. A detailed explanation of the difference with self-attention can be seen in section II-B.

In addition, it has a second interpretation in which the process resembles the operation of *Convolution* similar to the one used in convolutional layers. In this context, both *Convolution* and *Synthesis* relate contiguous and local elements to a receptive field. In the case of the *Convolution* operator, the receptive field is limited to a number of timesteps spatially placed in the data sample and normally contiguous or pseudocontiguous in the case of dilated convolutions in time [27], that largely depend on the size of the filter. In the case of *Synthesis*, this receptive field is limited to a temporal section, which depends on the exponent λ. Since *Convolution* has a spatial dependence on the input, the filters learned for one granularity will not, in general, be compatible for a different granularity. This is because the receptive field is static in that it covers a constant number of timesteps. On the contrary, the *Synthesis* operator does not have a spatial dependence on the input, but a temporal one. This means that for as many timesteps that exist, if the temporal range covered by all of them is the same as in lower granularity, all of them will be related within the same receptive field. This means that, consequently, the *Synthesis* operator provides invariance to the scale and granularity of the data.

On the other hand, the proposed operation in this work has the advantage of requiring fewer parameters and training time to establish relationships between local elements than self-attention based solutions, and is therefore particularly suitable for inputs containing local features, such as the granularity in energy time series.

II. METHODOLOGY

In this section, the *Synthesis* operation is described in detail in subsection II-A; a justification of the proposed method in subsection II-B; a description of the NN architecture that implements the method in subsection II-C; and the description of the loss function used in training in subsection II-D

A. SYNTHESIS OPERATION

The $\mathrm{Synthesis}(\boldsymbol{S}, \boldsymbol{E}, \boldsymbol{E}', \beta, \epsilon)$ operation is a function that allows a variable-length time series $\boldsymbol{S}$ to be expressed as a fixed-size vector $\boldsymbol{V}$ based on the hyperparameters β and ϵ, positional embeddings $\boldsymbol{E}$, and synthesis embeddings $\boldsymbol{E}'$. This representation $\boldsymbol{V}$ has the peculiar property of being invariant to the granularity of the time series $\boldsymbol{S}$ and to its size defined by $\boldsymbol{T}$. This means that the fixed-size representation $\boldsymbol{V}$ presents similar characteristics regardless of the granularity, as long as it refers to the same time series. The *Synthesis* operator requires positional embeddings $\boldsymbol{E}$ associated with the timesteps $\boldsymbol{T}$ of the time series $\boldsymbol{S}$, and synthesis embeddings $\boldsymbol{E}'$. Consequently, given a machine learning model defined as $\mathrm{M}(\boldsymbol{V}, \boldsymbol{\theta})$ where $\boldsymbol{V}$ is the model's input and $\boldsymbol{\theta}$ are its parameters, the model's prediction is $\hat{\boldsymbol{Y}}$ and the synthesis operator can be applied as shown in Eq. 1.

$$\begin{aligned} \boldsymbol{V} &= \mathrm{Synthesis}(\boldsymbol{S}, \boldsymbol{E}, \boldsymbol{E}', \beta, \epsilon) \\ \hat{\boldsymbol{Y}} &= \mathrm{M}(\boldsymbol{V}, \boldsymbol{\theta}) \end{aligned} \tag{1}$$

For the model $\mathrm{M}(\boldsymbol{V}, \boldsymbol{\theta})$, the input $\boldsymbol{V}$ will be a representation of $\boldsymbol{S}$ encoded by the *Synthesis* operator that will have a constant length defined by the hyperparameter β regardless of the granularity $\Delta\boldsymbol{T}$ and the length $\#\boldsymbol{T}$ of the time series $\boldsymbol{S}$.

A definition of positional embeddings can be seen in subsection II-A1; a definition of synthesis embeddings associated with the hyperparameter β can be seen in subsection II-A2; the weights of the *Synthesis* operator and an explanation of the hyperparameter ϵ can be seen in subsections II-A3 and II-A4 respectively; and the calculation of the final values in subsection II-A5.

1) POSITIONAL EMBEDDINGS

Positional embeddings $\boldsymbol{E}$ allow the position of an element in the time series to be identified relative to the entire series, and therefore, allow the order of the elements to be determined. This facilitates the establishment of temporal relationships between each element without the need for the model to have a receptive field of the entire time series. The *Synthesis* operator requires positional embeddings $\boldsymbol{E}$ to perform information compression.

Using sinusoidal functions as positional embeddings, as proposed in [17], this paper suggests establishing relationships between elements based on their order. However, to relate contiguous elements under a time-dependent receptive field, this work proposes using a single arc defined by a single sine and a single cosine as the positional embeddings representation, resulting in a total of two features for one-dimensional time series. This allows the similarity relationship to be established only between contiguous elements.

To compute $\boldsymbol{E}$ corresponding to a time series $\boldsymbol{S}$, a timestep must be marked as the starting positional element. This timestep is called $l_{\min}$ and must satisfy $l_{\min} \leq \min(T)$. Simi-

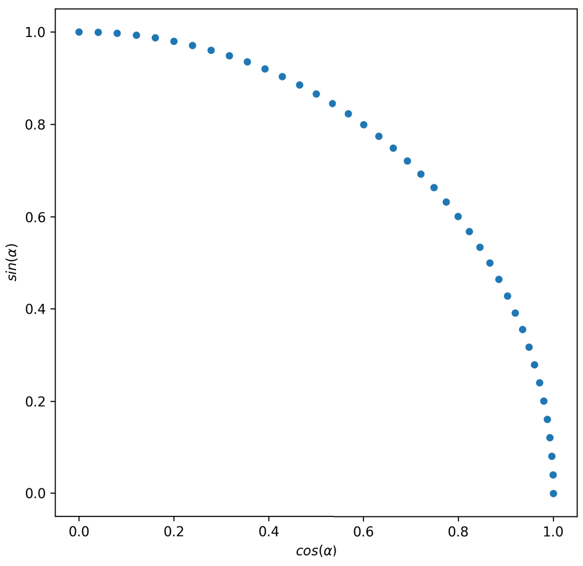


FIGURE 1. Positional Embeddings E for a time series with $\#T = 40$ and a constant granularity ΔT.

larly, a timestep must be established as the limiting element, called $l_{\max}$, and must satisfy $l_{\max} \geq \max(T)$. Consequently, $\boldsymbol{E}$ is defined as can be seen in equation 2.

$$\begin{aligned}\boldsymbol{\alpha} &= \frac{\pi(l_{\max} - \boldsymbol{T})}{2(l_{\max} - l_{\min})} \\ \boldsymbol{E} &= [\sin(\boldsymbol{\alpha}), \cos(\boldsymbol{\alpha})]\end{aligned} \tag{2}$$

where $\boldsymbol{T}$ is the timesteps of the time series expressed as milliseconds in the form of timestamps. The matrix $\boldsymbol{E}$ then contains all the timesteps of the time series dumped into an arc corresponding to the first quadrant of a circle of radius 1, as can be seen in Fig 1.

By using $\frac{\pi}{2}$ as the scale of the embeddings, both the sine and cosine are represented in the first quadrant of the circle, acquiring only positive values, regardless of the length of the time series.

2) SYNTHESIS EMBEDDINGS

The synthesis embeddings, defined as $\boldsymbol{E'}$, are positional embeddings that are periodically distributed along the arc defined by the points in $\boldsymbol{E}$. Formally, the synthesis embeddings $\boldsymbol{E'}$ can be obtained through the function $\text{Embs}(\boldsymbol{x}, \beta)$ as defined in Eq. 3.

$$\begin{aligned}\alpha'(\boldsymbol{x}, \beta) &= \boldsymbol{x}\frac{1}{\beta - 1}\frac{\pi}{2} \\ \text{Embs}(\boldsymbol{x}, \beta) &= \left[\sin\left(\alpha'(\boldsymbol{x}, \beta)\right), \cos\left(\alpha'(\boldsymbol{x}, \beta)\right)\right] \\ \boldsymbol{E'} &= \text{Embs}([0, 1, 2, \ldots, \beta - 1], \beta)\end{aligned} \tag{3}$$

The number of synthesis embeddings is a hyperparameter and is defined as β. This value determines the number of values in which the input to the *Synthesis* operator will be synthesized. Low values of β will reduce the size of the bottleneck and, consequently, produce loss of information. On the other hand, high values of β will allow more detailed information to pass through to the final synthesis, although they will require more computation and memory.

3) SYNTHESIS WEIGHTS

The *Synthesis* operation is carried out by generating weights $\boldsymbol{W}$ that relate the positional embeddings $\boldsymbol{E}$ to the synthesis embeddings $\boldsymbol{E'}$, so that each synthesis embedding $\boldsymbol{E'}$ has the focus set on a section of the input. The weights are defined as can be seen in Eq. 4.

$$\boldsymbol{W} = (\boldsymbol{E}\boldsymbol{E'}^{\intercal})^{\lambda} \tag{4}$$

Since $\boldsymbol{E}$ and $\boldsymbol{E'}$ are composed of vectors with equal length 1 in the first quadrant of the circle, the vector product between them is similar to the similarity returned by the Cosine Similarity function as can be seen in Eq. 5 but with $|\boldsymbol{E}| = |\boldsymbol{E'}| = 1$.

$$\cos(\gamma) = \frac{\boldsymbol{E}\boldsymbol{E'}^{\intercal}}{|\boldsymbol{E}||\boldsymbol{E'}|} \tag{5}$$

Therefore, the values of $\boldsymbol{W}$ will contain a similarity value that will range from 0 to 1, with 1 being the highest degree of proximity and 0 being the lowest degree of proximity. The exponent λ in Eq. 4 determines the degree of attention of the synthesis operation. A value of $\lambda = 1$ will generate weights that are equivalent to the cosine similarity, while a value of $\lambda > 1$ will increase the weight of the most similar embeddings and a value of $\lambda < 1$ will decrease the weight of the most similar embeddings. The exponent λ modulates the projection focus of the similarity, making the focal distance of each synthesis embedding in $\boldsymbol{E'}$ increase or decrease with respect to the positional embeddings in $\boldsymbol{E}$. This effect constitutes the receptive field that each synthesis embedding within $\boldsymbol{E'}$ has with each positional embedding within $\boldsymbol{E}$. A more detailed explanation of the exponent λ can be seen in the subsection II-A4. A representation of these weights can be seen in Figure 2.

4) SYNTHESIS EXPONENT

The exponent λ allows to increase or decrease the receptive field of each synthesis embedding. It is calculated through the $\text{lamb}(\beta, \epsilon)$ function, which depends on the hyperparameter β (defined in II-A2) and the hyperparameter ϵ, which specifies the similarity value that a positional embedding will have that matches the central point between two synthesis embeddings. The $\text{lamb}(\beta, \epsilon)$ function is defined as can be seen in Equation 6.

$$\begin{aligned}\text{lamb}(\beta, \epsilon) &= \frac{\log \beta}{\log\left(\text{Embs}(0, \beta) \times \text{Embs}(\frac{1}{2}, \beta)\right)} \\ \lambda &= \text{lamb}(\beta, \epsilon)\end{aligned} \tag{6}$$

The value of ϵ must be a number bounded between $[0, \frac{1}{2}]$, since it represents the value that the similarity will have in a central point between two synthesis embeddings. The establishment of this hyperparameter is necessary since the central point between two synthesis embeddings presents overlap in the receptive field of both points. Therefore, the value of

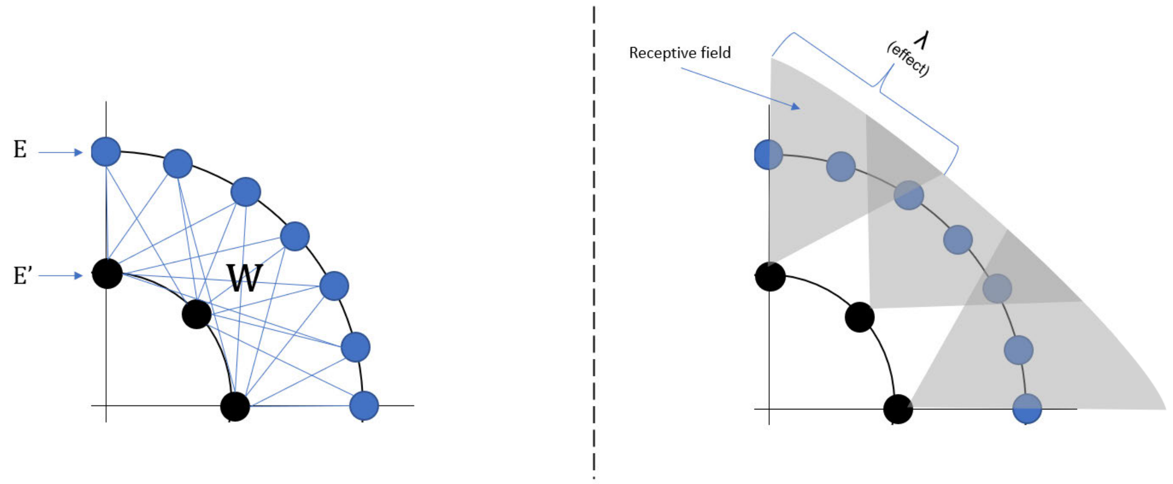


FIGURE 2. Representation of the weights W relating E with E' (left) and the effect of λ over the receptive field of every synthesis embedding (right).

$\frac{1}{2}$ makes each synthesis embedding acquire a maximum of similarity with that central point that allows the final sum of the projections of the receptive fields to produce values approximately bounded between [0, 1] for the entire $\boldsymbol{S}$ series.

5) SYNTHESIS VALUES

The final synthesis values $\boldsymbol{V}$ resulting from the application of the *Synthesis* operator to the time series $\boldsymbol{S}$ can be computed as observed in Eq. 7

$$\boldsymbol{V} = \text{Synthesis}(\boldsymbol{S}, \boldsymbol{E}, \boldsymbol{E'}, \beta, \epsilon) = \boldsymbol{W}^{\intercal} S \tag{7}$$

B. SYNTHESIS VERSUS SELF-ATTENTION

One of the interpretations described in section I-C is that the *Synthesis* operator can be interpreted as a variant of self-attention. Using the Scaled Dot-Product Attention proposed by [17] as reference, as can be seen in Eq. 8,

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\intercal}}{\sqrt{(d_k)}}\right)\boldsymbol{V} \tag{8}$$

The positional embeddings $\boldsymbol{E}$ would be equivalent to the keys $\boldsymbol{K}$, the synthesis embeddings $\boldsymbol{E'}$ would be equivalent to the queries $\boldsymbol{Q}$, and the value of the time series $\boldsymbol{S}$ would be equivalent to $\boldsymbol{V}$. While in self-attention the values of $\boldsymbol{Q}$ are usually learned during training, in the *Synthesis* operation these values are pre-defined so that each one covers a contiguous section of the input, with this input being entirely distributed among the synthesis embeddings $\boldsymbol{E'}$. In addition, the receptive field of each synthesis embedding $\boldsymbol{E'}$ on the positional embeddings $\boldsymbol{E}$ is regulated by the exponent λ. Unlike in self-attention, the goal of the *Synthesis* operator is to limit the visibility of a synthesis embedding to a very specific and contiguous section of the input. Since $\boldsymbol{E'}$ is equitably distributed among the input $\boldsymbol{S}$, the model has no parameters to adjust during training in this layer, and consequently, semantic relationships are not established between elements. However, the establishment of semantic relationships between elements is delegated to the subsequent layers that have visibility of the fixed-length vector resulting from applying *Synthesis* and that have learnable parameters.

C. NETWORK ARCHITECTURE

To verify the validity of the method, a NN architecture was designed that uses the *Synthesis* operation embedded as an intermediate layer of the NN to provide granularity invariance. This architecture allows for classification into C classes of a time series of variable length and/or granularity disparity. The network consists of a 1-dimensional *Synthesis* layer with $\beta = 30$ and $\epsilon = 0.25$, in which the positional embeddings $\boldsymbol{E}$ and the time series $\boldsymbol{S}$ are combined. The result of the *Synthesis* operation is then passed to two convolutional layers, with the first consisting of 32 filters of size 1 timestep with Rectified Linear Unit (*ReLU*) activation, and the second convolutional layer consisting of C filters of size 30 timesteps, which is equal to β, with *Softmax* activation for classification. Finally, a *Squeeze* is performed on the first dimension to maintain only the classification values. All hyperparameters were set experimentally, although any other configuration could serve the purpose of the experiments defined in section III. A visualization of the architecture can be seen in Fig. 3.

In total, the model has 11,596 trainable parameters.

D. LOSS FUNCTION

The error function used to train the proposed model is Categorical Cross-Entropy, as can be observed in Eq. 9

$$\text{CE}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = -\sum_{i=1}^{N} \boldsymbol{Y}_i - \log(\hat{\boldsymbol{Y}}_i) \tag{9}$$

where CE stands for Cross-Entropy, N is the number of classes, i is the index of the class, $\hat{\boldsymbol{Y}}$ is the prediction made by the model M (formally defined in II-A and detailed in II-C), and $\boldsymbol{Y}$ is the ground truth. Through the minimization of the error reported by CE for multiple samples simultaneously, the parameters $\boldsymbol{\theta}$ of M are optimized.

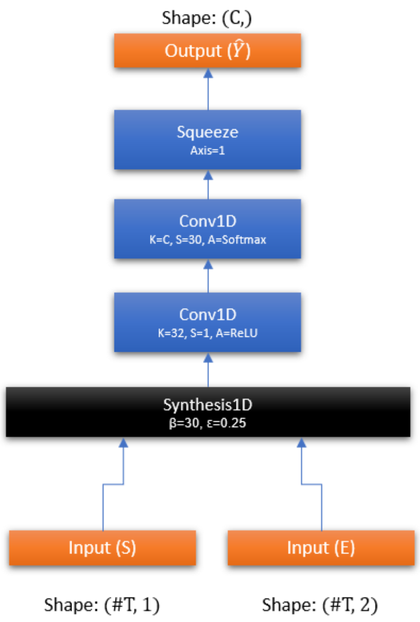


FIGURE 3. Network architecture proposed for the experimentation.

III. EXPERIMENTATION

The goal of the experiments is to demonstrate the granularity invariance of the model thanks to the incorporation of the *Synthesis* operator in its architecture. To this end, a model was proposed to be trained that is able to classify the energy production of each month in its corresponding numeric month only using the production power time series in that month.

A description of the dataset used can be observed in subsection III-A. A description of the configuration carried out in the experimentation process, the possible scenarios, and the data adjustment process can be seen in subsection III-C. The results of the experimentation can be seen in subsections III-D and III-E. Finally, a discussion of the results is presented in section IV.

A. DATASET NIST CAMPUS PHOTOVOLTAIC

In this study, the public NIST Campus Photovoltaic dataset [2] was used, from which the data with 1-minute granularity at ground level was selected. The data covers from *2015/01/01 00:00* to *2018/12/31 23:59*, which makes a total of *2,103,810* measurements of various signals, among which stand out atmospheric measurements such as wind, ambient and device temperature sensors, irradiance and production signals, such as the energy produced by each inverter and power. For the purpose of the experiments, the average power signal of the meters measured in Kilo-Watt (kW) was selected, which was normalized by replacing the negative values with 0 and dividing the entire signal by the maximum value. A visualization of the normalized graph corresponding to the dataset can be seen in Fig. 4.

B. METRICS

To evaluate the problem, the *precision*, *recall* and *F1* metrics defined in Eq. 10 are used to measure the quality of

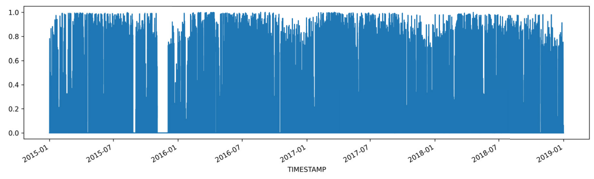


FIGURE 4. NIST Campus Photovoltaic dataset. Representation of the average meter power normalized between 0 and 1 during 4 years (2015-2018).

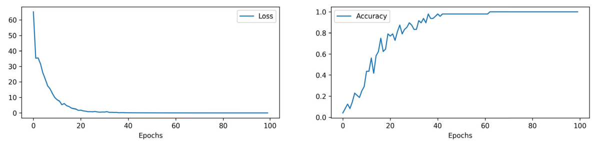


FIGURE 5. Training Cross-Entropy loss (left). Training accuracy (right). Training performed for 100 epochs.

classification performed by the network when different granularities are presented in the input.

$$\begin{aligned}\text{precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}}\\ \text{recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}}\\ \text{F1} &= 2\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}\end{aligned} \tag{10}$$

The TP value indicates the number of true positives, the FP value indicates the number of false positives, and the FN value indicates the number of false negatives.

C. SETUP

An experiment was carried out to demonstrate the granularity invariance provided by the *Synthesis* operator when embedded in a neural network. To do this, the dataset described in Section III-A was preprocessed to generate a sample for each month of photovoltaic production. In this way, the input time series $\boldsymbol{S}$ is constituted by the photovoltaic production series of a month with a padding of 31 days, filling with 0s at the end of each sample if necessary. The goal of the prediction $\hat{\boldsymbol{Y}}$ is the classification in the month to which the time series $\boldsymbol{S}$ belongs.

The model proposed in subsection II-C was configured with $C = 12$ in order to classify each time series $\boldsymbol{S}$ into one of the 12 months of the year. All samples generated using the NIST Campus Photovoltaic dataset were downsampled to a granularity $\Delta\boldsymbol{T}$ of 5 minutes and were used to train the model for 100 epochs on a Xeon W-2123 equipped with a Nvidia GeForce RTX 2080Ti GPU. The Adam optimizer with parameters $\text{lr} = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-7}$ was used in the optimizer, adjusted experimentally, with a batch_size = 16. A visualization of the training history can be seen in Fig. 5.

Once trained, the experiments were divided into two cases in order to demonstrate the invariance of the method to granularity in two different contexts. The first case demonstrates the ability to predict the class invariantly to the granularity of the time series as a whole. This first case is described in

TABLE 1. Performance metrics of the classification model for different granularities. The bold row is the reference granularity used for training. #T is the number of timesteps of the series.

Granularity	Precision	Recall	F1	#T	#T % Change
1 min	1.0	1.0	1.0	44640	+400.0%
2 mins	1.0	1.0	1.0	22320	+150.0%
5 mins	**1.0**	**1.0**	**1.0**	**8928**	**0.0%**
10 mins	1.0	1.0	1.0	4464	-50.0%
15 mins	1.0	1.0	1.0	2976	-67.7%
30 mins	1.0	1.0	1.0	1488	-83.3%
1 hour	1.0	1.0	1.0	744	-91.6%
2 hours	0.983	0.979	0.979	372	-95.8%
4 hours	0.950	0.938	0.937	186	-97.9%
8 hours	0.617	0.521	0.517	93	-99.0%
16 hours	0.325	0.375	0.317	47	-99.4%
1 day	0.230	0.250	0.221	31	-99.6%

subsection III-D. The second case demonstrates the ability to predict the class invariantly to how many different granularities the same time series has over time. This second case is described in subsection III-E.

D. CASE 1: OVERALL GRANULARITY

In this experimental case, a time series $\boldsymbol{S}$ with a single granularity is presented to the model. A comparison of the model predicting the corresponding month of the same time series expressed with different granularities can be seen in Figures 6, 7, and 8. The first row of each figure represents the original time series expressed in the chosen granularity or resolution for each column, and the second row represents the corresponding encoding generated by the *Synthesis* operation.

In order to determine the invariance capacity, the method was evaluated using the same time series used in the training but resampled at the granularities of 1 minute, 2 minutes, 5 minutes, 10 minutes, 15 minutes, 30 minutes, 1 hour, 2 hours, 4 hours, 8 hours, 16 hours, and 1 day. They were evaluated under the premise that if the model shows invariance to granularity, it should be able to classify the same series used in the training even if they are expressed in different granularities than those observed in the training during the prediction. This evaluation can be seen in Table 1.

E. CASE 2: MANY GRANULARITIES

Another evaluation case is having different granularities within the same time series $\boldsymbol{S}$; that is, a time series whose granularity changes over time, as expressed in Eq. 11.

$$\Delta \boldsymbol{T} = [\Delta \boldsymbol{T}_{a_0->a_1}, \Delta \boldsymbol{T}_{a_1->a2}, \ldots, \Delta \boldsymbol{T}_{a_{n-1}->a_n}] \quad (11)$$

where $a \in \boldsymbol{T}$; $a_i > a_{i-1}$.

To carry out this experiment, the following 6 granularities of best performance in the first case were selected: 1 minute, 5 minutes, 15 minutes, 30 minutes, 60 minutes, and 120 minutes. Then, for each sample, a downsample of random sections was made for each of the chosen granularities. Therefore, an evaluation dataset was built iteratively.

In order to evaluate the performance of the classification with the presence of several resolutions at the same time, 5 different versions were also generated for each sample, in which each version has a greater number of granularity variations. An example of a time series $\boldsymbol{S}$ expressed in a multitude of versions can be observed in Fig. 9, where the same series $\boldsymbol{S}$ is shown with 1, 2, 3, 4, and 5 different granularities over time, corresponding to each row of the figure. The evaluation of the performance of different granularities in the same series can be seen in Table 2, where the result of applying the evaluation metrics on series that have from 2 to 6 different granularities is grouped.

TABLE 2. Performance metrics evaluated under series with many granularities.

#Granularities	Precision	Recall	F1
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	1.0	1.0
5	1.0	1.0	1.0
6	1.0	1.0	1.0

IV. DISCUSSION

In this section, the performance of the model under the different experiments executed is evaluated and its behavior is analyzed.

First of all, it is interesting to note the speed with which the model adjusted to the training data, obtaining a 100% accuracy, since the limit of the metric was reached in less than 80 epochs, as can be seen in Fig. 5. This indicates that both the proposed model and the *Synthesis* operator perform the expected work correctly to be used in a classification process. On the other hand, this also confirms that the month corresponding to a given monthly production is easily determinable. This is of interesting application in anomaly detection, since a poorly predicted month can indicate an anomaly in the data.

In the first experimental case carried out, it can be observed that, despite being trained with 5-minute measures, the model is able to correctly classify the measures for smaller granularities (1-minute) and larger granularities (up to almost 2 hours of granularity) without needing to be modified or re-trained, as seen in Table 1. This means that the model shows invariance to granularity, making any granularity able to be classified by the same model. It should be noted that at lower granularities, the amount of representation timesteps is smaller and, therefore, information loss occurs that explains that the classification deteriorates from 2 hours of granularity onwards, since the number of timesteps #$\boldsymbol{T}$ in the 2-hour case is 95.8% smaller than in the 5-minute case with which the model was trained. In figures 6 and 8, it can be observed that, for different granularities manifested with the same production time series (each column), the representation of the encoding is similar, despite the input having differences both in scale and in number of timesteps. In figure 7, a correct classification was made even with a daily granularity, which means that the model has prediction capacity even with a reduction of 99.6% of the information in the data in terms of granularity. However, Table 1 indicates that the model loses reliability from an approximate loss of 98% of data informa-

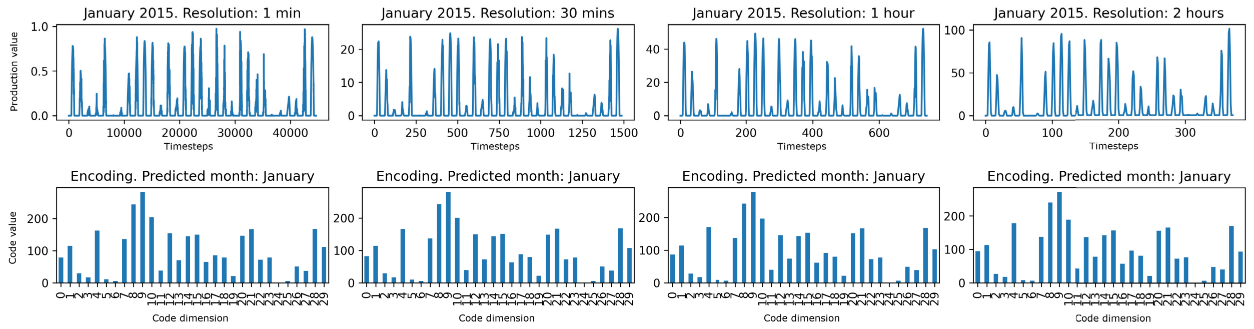


FIGURE 6. A time series S corresponding to January 2015 presented to the model under different granularities. first row indicates the input series S and the second row indicates the coding output of the *Synthesis* method. All the cases were correctly classified as January by the model.

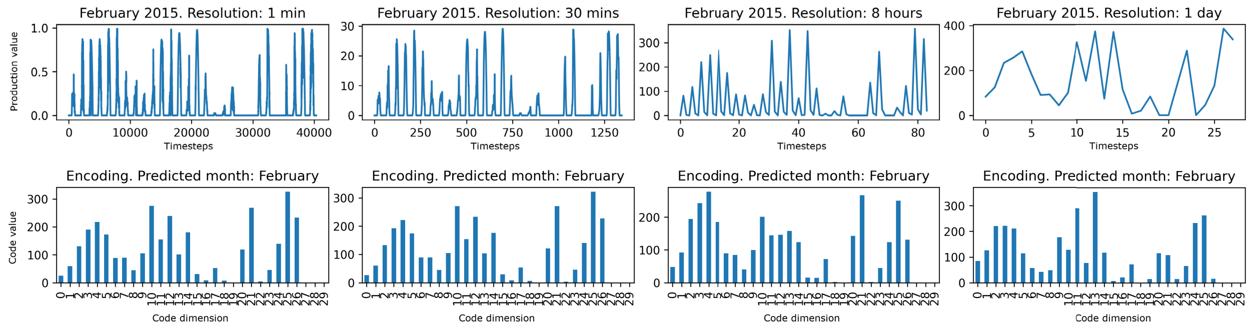


FIGURE 7. A time series S corresponding to February 2015 presented to the model under different granularities. first row indicates the input series S and the second row indicates the coding output of the *Synthesis* method. All the granularities were corrrectly classified by the model.

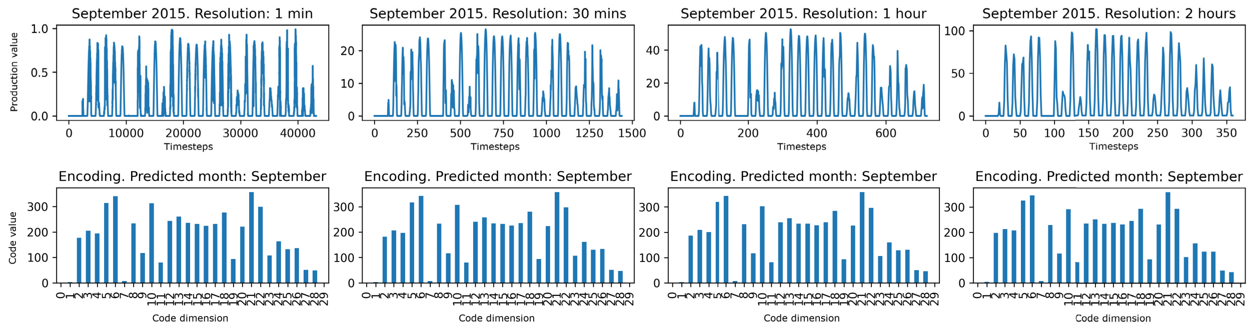


FIGURE 8. A time series S corresponding to September 2015 presented to the model under different granularities. first row indicates the input series S and the second row indicates the coding output of the *Synthesis* method. All the granularities were correctly classified by the model.

tion in terms of number of timesteps, due to a decrease in granularity.

In the second case, it is shown that the proposed model is able to handle any level of granularity within a time series, as demonstrated in Table 2. This table shows that the model was able to automatically identify all granularity variations within a single time series without needing to be retrained, even in cases where 5 granularity variations coexist. An example of this is shown in Fig. 9, where each row represents the same time series at a different level of granularity, ranging from one granularity in the first row to five granularities in the last row. The intermediate rows show an increasing number of granularities. It is interesting to note that having different granularities in the same time series can cause a small number of timesteps to contain a large amount of information, while the majority of timesteps may contain little information, as shown in the second row of Fig. 9. In this example, two granularities coexist, with the first being 15 minutes for 1250 timesteps and the second being 1 hour for 500 timesteps. However, the last 500 timesteps contain the condensed information of 30, 000 measurements, while the first 1250 timesteps contain condensed 18, 750 measurements. In other words, in this case, 61% of the information of the time series is condensed in the last 500 timesteps, and the *Synthesis* operator is able to correctly handle data and prediction in these scenarios. This means that the proposed model is invariant to changes in granularity within the time series, allowing it to generalize to unseen environments and non-standard granularities. Therefore, the *Synthesis* operator is a promising method for generating granularity-invariant models for predicting energy consumption and production time series.

In addition, in the *Synthesis* operator, the receptive field has a dynamic dimension and is enlarged or reduced according to

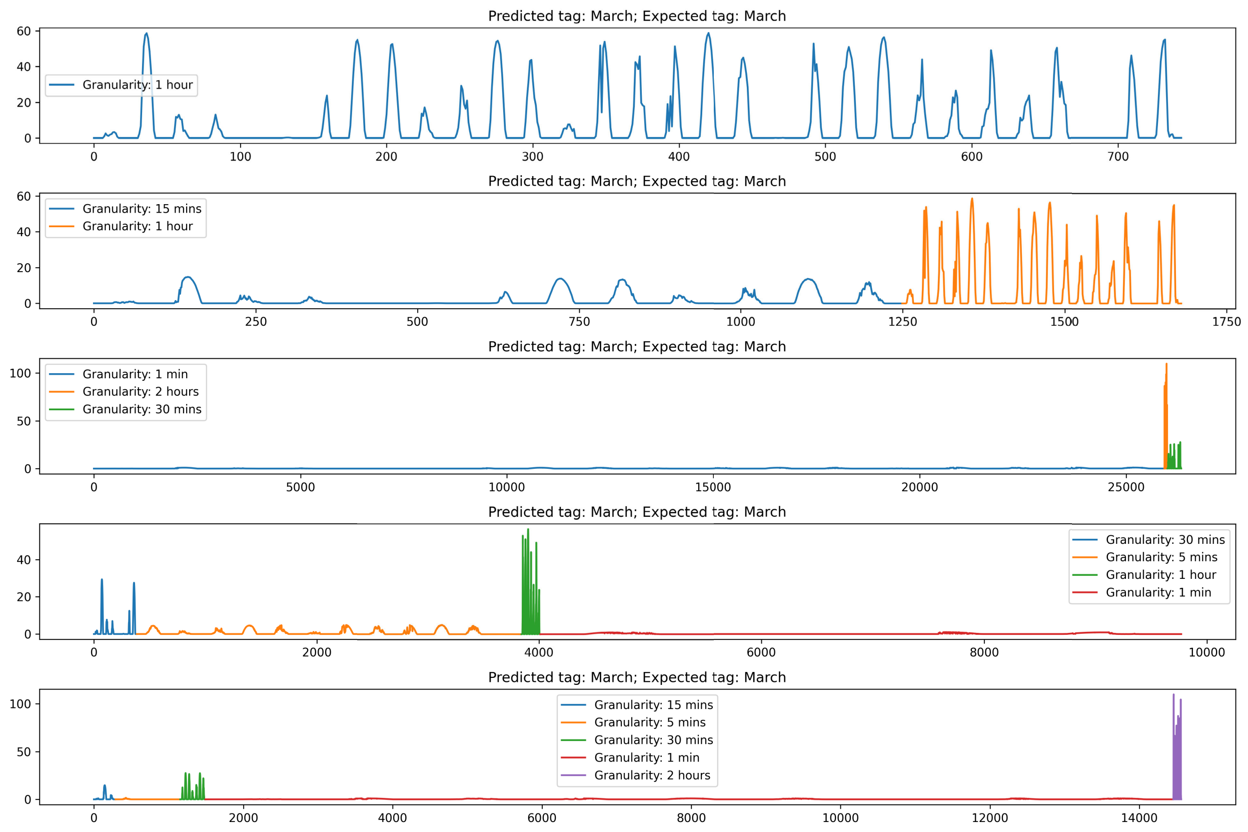


FIGURE 9. **Representation of the same sample expressed as 5 different versions containing different number of granularities. Every granularity has a different color in the series. Note the different representation and the length of the series, even though belong to the same original *S*. The model was able to classify all of the versions correctly. Best viewed in color.**

the size of the input, so that the output maintains a constant size regardless of the size of the input, making it also invariant to the length of the time series. This makes it suitable for compressing dynamic-sized data. On the other hand, it is interesting to note that, since the model is able to represent different granularities under a single encoding, it could be used as profiling features of a production and/or as a hashing function. This makes it an ideal representation for clustering processes.

V. CONCLUSION

A new time series compression method called *Synthesis* is presented, which is able to convert arbitrarily long production time series into fixed-size vectors. The main advantage of the method is that it provides granularity and length invariance. This means that, once trained with a custom granularity, the model is able to generalize the predictions to other granularities as well. An artificial NN architecture that implements the *Synthesis* method in its first layer is also presented. Experiments are conducted to validate the granularity invariance of the method, in which the proposed NN is trained to classify a monthly time series into the month of the year to which it belongs. The same time series expressed in different granularities is then evaluated, showing that the method correctly classifies the time series even when 98% of the information has been lost in terms of granularity, without having to retrain the model. The model innately supports time series that have multiple different granularities over time. The proposed method is conceptually compared to self-attention and convolution, showing its superiority in handling time series by generating granularity invariance. The method can compress variable-size information into a fixed-size vector, which means that the encoding generated from series of different granularities maintains the same representation. This makes it applicable in scenarios such as data profiling, as a hashing function, and as a data source for clustering algorithms.

REFERENCES

[1] M. Khalil, A. S. McGough, Z. Pourmirza, M. Pazhoohesh, and S. Walker, "Machine learning, deep learning and statistical analysis for forecasting building energy consumption—A systematic review," *Eng. Appl. Artif. Intell.*, vol. 115, Oct. 2022, Art. no. 105287.

[2] M. Boyd, T. Chen, and B. Dougherty, "NIST campus photovoltaic (PV) arrays and weather station data sets, [Data set]," Nat. Inst. Standards Technol., 2017, Accessed: Dec. 1, 2022, doi: 10.18434/M3S67G.

[3] R. Zurita-Milla, G. Kaiser, J. G. P. W. Clevers, W. Schneider, and M. E. Schaepman, "Downscaling time series of MERIS full resolution data to monitor vegetation seasonal dynamics," *Remote Sens. Environ.*, vol. 113, no. 9, pp. 1874–1885, 2009.

[4] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining Knowl. Discovery*, vol. 29, no. 3, pp. 565–592, 2015.

[5] H. Wang, H. Yi, J. Peng, G. Wang, Y. Liu, H. Jiang, and W. Liu, "Deterministic and probabilistic forecasting of photovoltaic power based on deep convolutional neural network," *Energy Convers. Manag.*, vol. 153, pp. 409–422, Dec. 2017.

[6] T. Le 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 Mining Knowl. Discovery*, vol. 33, no. 4, pp. 1183–1222, Jul. 2019.

[7] Y. Zou, R. V. Donner, N. Marwan, J. F. Donges, and J. Kurths, "Complex network approaches to nonlinear time series analysis," *Phys. Rep.*, vol. 787, pp. 1–97, Jan. 2019.

[8] D. Hadjout, J. F. Torres, A. Troncoso, A. Sebaa, and F. Martínez-Álvarez, "Electricity consumption forecasting based on ensemble deep learning with application to the Algerian market," *Energy*, vol. 243, Mar. 2022, Art. no. 123060.

[9] K. A. Althelaya, S. A. Mohammed, and E.-S.-M. El-Alfy, "Combining deep learning and multiresolution analysis for stock market forecasting," *IEEE Access*, vol. 9, pp. 13099–13111, 2021.

[10] H. Sha, P. Xu, M. Lin, C. Peng, and Q. Dou, "Development of a multi-granularity energy forecasting toolkit for demand response baseline calculation," *Appl. Energy*, vol. 289, May 2021, Art. no. 116652.

[11] X. Ren, F. Zhang, H. Zhu, and Y. Liu, "Quad-kernel deep convolutional neural network for intra-hour photovoltaic power forecasting," *Appl. Energy*, vol. 323, Oct. 2022, Art. no. 119682.

[12] S. Tonekaboni, D. Eytan, and A. Goldenberg, "Unsupervised representation learning for time series with temporal neighborhood coding," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–17.

[13] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," 2016, *arXiv:1601.06733*.

[14] Z. Lin, M. Feng, C. N. D. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," 2017, *arXiv:1703.03130*.

[15] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Austin, TX, USA, Nov. 2016, pp. 2249–2255.

[16] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017, *arXiv:1705.04304*.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[18] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Aug. 2021, pp. 2114–2124.

[19] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 1–15.

[20] I. De-Paz-Centeno, M. T. García-Ordás, Ó. García-Olalla, and H. Alaiz-Moretón, "Imputation of missing measurements in PV production data within constrained environments," *Exp. Syst. Appl.*, vol. 217, May 2023, Art. no. 119510.

[21] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. A. Brubaker, "Time2Vec: Learning a vector representation of time," 2019, *arXiv:1907.05321*.

[22] Y. Shen, X. Jiang, Y. Wang, X. Jin, and X. Cheng, "Dynamic relation extraction with a learnable temporal encoding method," in *Proc. IEEE Int. Conf. Knowl. Graph (ICKG)*, Aug. 2020, pp. 235–242.

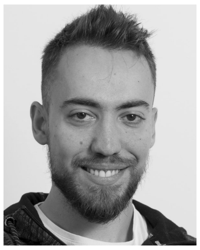
[23] L. S. Saoud, H. Al-Marzouqi, and R. Hussein, "Household energy consumption prediction using the stationary wavelet transform and transformers," *IEEE Access*, vol. 10, pp. 5171–5183, 2022.

[24] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "TS2Vec: Towards universal representation of time series," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, vol. 36, no. 8, pp. 8980–8987.

[25] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–20.

[26] Y.-C. Chen, Y.-J. Li, X. Du, and Y.-C. F. Wang, "Learning resolution-invariant deep representations for person re-identification," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, vol. 33, no. 1, pp. 8215–8222.

[27] A. V. D. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*.



IVÁN DE-PAZ-CENTENO received the undergraduate and M.Sc. degrees in computer science from the University of León, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree in intelligent systems. Since 2018, he has been a Data Scientist and a Research and Development Engineer with SMARKIA ENERGY SL. His research interests include machine-learning and deep-learning applied to energy time-series.

MARÍA TERESA GARCÍA-ORDÁS received the degree in computer science and the Ph.D. degree in intelligent systems from the University of León, in 2010 and 2017, respectively. Since 2019, she has ben a Teaching Assistant with the University of León. She has published several articles in impact journals and patents. She has participated in many conferences all over the world. Her research interests include computer vision and deep-learning. She was a recipient of the Special Mention Award for the best doctoral thesis on digital transformation from Tecnalia.

ÓSCAR GARCÍA-OLALLA received the degree in computer science and the Ph.D. degree in intelligent systems from the University of León, in 2010 and 2017, respectively. Since 2016, he has been a Data Scientist and Research and Development Engineer. Since 2020, he has been a Professional Data Engineer certified by Google Cloud. He has participated in many national and international research projects. His research interests include deep-learning and computer vision.

CARLOS GIRÓN-CASARES received the degree in telecommunications, the M.Sc. degree in electronics, and the Ph.D. degree in advanced electronics systems from Universidad de Alcalá, in 2004, 2007, and 2017, respectively. He worked for ten years at GEISER Research Group, as a CTO in several companies, and colaborates with several researching groups. He is a CTO with SMARKIA ENERGY SL.

HÉCTOR ALAIZ-MORETÓN received the degree in computer science and the Ph.D. degree in information technologies from Universidad de León, in 2003 and 2008, respectively. Since 2005, he has been a Lecturer with the School of Engineering, Universidad de León. He has published in international journals, conferences and books; with more than 80 scientific publications. His research interests include knowledge engineering, machine-learning, deep-learning, and networks communication and security. He has headed several Ph.D. thesis and competitive research projects. He is the Vice Principal of the Research Institute in Applied Science in Cybersecurity (RIASC). He has been a member of scientific committees in conferences.