Application of Deep Learning in Recurrence Plots for Multivariate Nonlinear Time Series Forecasting

Sun Arthur A. Ojeda, Elmer C. Peramo, and Geoffrey A. Solano

Abstract We present a framework for multivariate nonlinear time series forecasting that utilizes phase space image representations and deep learning. Recurrence plots (RP) are a phase space visualization tool used for the analysis of dynamical systems. This approach takes advantage of recurrence plots that are used as input image representations for a class of deep learning algorithms called convolutional neural networks. We show that by leveraging recurrence plots with optimal embedding parameters, appropriate representations of underlying dynamics are obtained by the proposed autoregressive deep learning model to produce forecasts.

1 Introduction

Forecasting time series has been instrumental in providing scientific and operational insights on various problem domains. Statistical and machine learning (ML) methods are commonly used in time series modelling. Such methods include autoregressive (AR) models, vector autoregression (VAR), exponential smoothing, neural networks, and gradient boosting algorithms. Traditional statistical methods rely on significant effort in model tuning and adequate domain knowledge to obtain meaningful results. Machine learning algorithms are more flexible in terms of these modelling factors, but still require appropriate data processing treatments, which must be selected

Sun Arthur A. Ojeda
University of the Philippines Manila
e-mail: saojeda@up.edu.ph

Geoffrey A. Solano
University of the Philippines Manila
e-mail: gasolano@up.edu.ph

Elmer C. Peramo
DOST - Advanced Science and Technology Institute
e-mail: elmer@asti.dost.gov.ph

carefully with consideration of the nature of the data and the problem's complexity. With the abundance of data and computational resources, deep learning (DL) offers the most flexibility due to its capacity to learn parameter space feature abstractions without feature engineering, which is necessary for machine learning counterparts.

Forecasting time series can be framed as a regression task in which a forecast $\hat{\mathbf{x}}_{t+1}$ is modelled as $\hat{\mathbf{x}}_{t+1} = \mathrm{F}(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \ldots, \mathbf{x}_{t-k-1}; \Theta)$ where F is an arbitrary model with a parameter tensor Θ learned and estimated from data and $\mathbf{x}_t \in \mathbb{R}^d$ is a vector containing observations at time t with dimensionality d, i.e. the number of variables. The model obtains $\hat{\mathbf{x}}_{t+2}$ by feeding $\hat{\mathbf{x}}_{t+1}$ as a new input to F taking k lagged inputs. This is repeated to obtain forecast values for the succeeding time steps. This process is known as a repeated *one-step-ahead forecast*.

In this work, we develop a framework based on deep learning that builds on the one-step-ahead forecasting approach to modelling and takes advantage of time series image representations to learn discriminative features on its own, which allows for flexibility and convenience since it generally requires minimal data processing and analysis in order to achieve considerable results. First, evidence for nonlinearity in data is obtained through statistical tests. Second, the theoretical aspects of the proposed imaging transformation are discussed. Third, the proposed model based on a convolutional neural network is outlined. Afterwards, a model training and inference pipeline is presented. Finally, we perform statistical comparisons between different configurations of the proposed model and state-of-the-art models adapted to multivariate time series regression.

2 Related Work

2.1 Background on Recurrence Plots

Time series data generated from real-world processes are generally a product of dynamical systems, which exhibits nonlinear behavior. They are usually non-stationary. These dynamical systems are an active field of study, and Eckmann et al. in 1987 [1] proposed a graphical tool called recurrence plots (RP) for measuring its time constancy. State-space reconstruction lays the foundation of nonlinear time-series analysis. Through recurrences in the state space, this tool helps researchers analyze dynamical systems by showing information that are not easily discernible by other methods. Since then, there has been a growing interest in the topic because of its wide applicability, as most techniques would require that a time series be stationary or can be transformed to one by some algebraic operation. This is evident in the comprehensive work of Casdagli (1997) [2] where he utilized RPs to give a more detailed characterization, prediction, change point detection, and hypothesis testing of a time series realized by dynamical systems and provided a more computationally efficient technique in doing these. The continued development in computation hardware, innovations in algorithms and software, and novel approaches in mathematics support the development of the field. Another interesting technical work about the

topic was explored by Thiel et al. [3] where they expound on the amount of information contained in an RP. They shed light on the conditions for the reconstructability of the underlying dynamics from which the time series was realized, which effectively validates Taken's Theorem.

The interest in RP as a visualization tool prompted researchers to explore its interpretability and effectiveness through some systematic and structural measures. This led to the development of the field called recurrence quantification analysis (RQA), which established useful quantitative characterizations of the structures that appear on the RP. To produce an effective recurrence plot from a realization of a time series, two important parameters must be chosen. These are the amount of time delay or lag τ and the embedding dimension m. Most of the work done in the estimation of these parameters employs meta-heuristic approaches: For the lag parameter, the most popular technique is choosing the first minimum of the logarithm of the generalized correlation integral [4] or based on the mutual information [5] between the original time series and the lagged time series. Estimating the embedding dimension m is relatively easier than estimating the lag value, and the most popular approach is the use of false nearest neighbor algorithm [6]. In 1998, Iwanski and Bradley [7] argued that the reconstructability of the hidden dynamics of a time series appear to be independent of the embedding dimension as derived from several experimental data. Based on their work, it's immaterial whether to embed or not to embed the data into a space with higher dimensions. Their experiments suggest that qualitative features that are visible in an RP using a particular value of m persists or are also visible in another RP with a different embedding dimension. Gao and Cai [8] did further studies on RPs exploring their patterns and structures on the different values of the embedding parameters. They were able to identify patterns and derived the features from the time series that generated those patterns. Based on differential entropy, Gautama et al. [9] hypothesized that optimal values for both the lag and the embedding dimension can be simultaneously determined. With the introduction of random surrogate data, they proposed an entropy ratio method to compute these values. Application of RPs and RQA goes beyond univariate time series as Marwan [10] and et al. proposed a generalized method in higher-dimensional spatial data—CT images.

Aside from the lag value and the embedding dimension, another interesting parameter of RP construction is the threshold. Sipers [11] et al. studied unthresholded RPs and determined theoretically and analytically which information can and cannot be extracted from a signal. They validated their results experimentally from EEG data. Penatti et al. [12] used RPs for human activity recognition based on mobile inertial sensors. They computed RPs from sensor data and applied machine learning techniques for prediction. More and application-centric of RP research have been published in recent years. Noteworthy is the work of Marwan [13], who compiled a survey of the different research and practical applications and review how this interesting field had advanced and proliferated over the years. Since the invention of RP more than two decades ago, more than a hundred research papers have been published with its application ranging from medicine, biology, chemistry, physics, engineering, psychology, material science, economics, geology, meteorology, acous-

tics, finance, and engineering. What the researchers are attempting to do here is to marry the field of RP and the successful innovations in deep learning, leveraging on each of the domain's strength that can be generalized to come up with a framework that can be applied to any real-world time series. We would also like to hope that by this work, the field would still gain more attention because we believe that a lot more potential still have to be realized from the synergy between these two fields.

2.2 Time Series Imaging and Convolutional Neural Networks

Wang and Oates [23] proposed two algorithms: Gramian Angular Fields (GAF) and Markov Transition Fields) for time series imaging. Both of these algorithms take time series data to produce their respective image representations. Gramian Angular Field takes advantage of polar coordinates to encode radial and angular information in the Gramian Matrix, which is essentially an image matrix. Markov Transition Field, on the other hand, identifies quantile bins and then encodes their pairwise transition probabilities in a Markov Transition matrix representing an image. These images are fed to a Tiled Convolutional Neural Network (Tiled CNN) that is used for solving classification problems. Benchmarks reveal that the GAF-MTF approach outperforms 1NN, SAX-VSM, and other state-of-the-art machine learning algorithms in 12 datasets from timeseriesclassification.com.

Hatami et al. [24] demonstrated the use of recurrence plots (RP) and convolutional neural networks for time series classification. They proposed a pipeline to convert time series into texture images using recurrence plots that are fed to the model to produce predictions. Results show that in 20 univariate time series classification datasets from the UCR archive the method outperforms approaches that rely on time series encoding such as Gramian Angular Fields (GAF) and Markov Transition Fields (MTF).

Several works focused on applying recurrence plots and convolutional neural networks were motivated by time series imaging using recurrence plots after having achieved state-of-the-art results in various domains. Hsueh [25] applied recurrence plot imaging and used a convolutional neural network model to diagnose induction motor's fault conditions including bearing axis deviation and stator and rotor friction for monitoring induction motor health and obtained 99.81% validation accuracy. Using a similar approach for fetal hypoxia diagnosis, Zhao et al. [26] implemented a Computer-Aided Diagnosis System that also utilizes a convolutional neural network to obtain a 10-fold cross-validation accuracy of 98.69%. Finally, for human activity recognition based on raw sensor data, Ceja et al. [27] achieved better results using recurrence plots against models that use feature-extraction methods such Deep Belief Network and Multi-layer Perceptron by obtaining a 10-fold cross-validation accuracy of 94.2%. Finally, we extend the methodology of our previous work [28] designed for time series classification applied to the meteorological domain. In this work, our framework for multivariate nonlinear time series forecasting can be used in any

domain, as long as the forecasting task can be appropriately represented as described in the succeeding sections.

3 Time Series Nonlinearity

Data complexity is an important factor that fundamentally affects time series modelling. Several data complexity measures have been reviewed by Sotoca et al. in [15] particularly designed for classification models. Li and Abu-Mustafa [16] developed several measures based on Kolmogorov complexity. These methods, however, are not model-agnostic. To characterize the inherent complexity in time series data that is independent of the model in question, we consider *time series nonlinearity* as an ideal specification of data complexity.

A univariate time series $\{x_t\}$ is linear if it that can be represented as

$$x_t = \sum_{j=-\infty}^{\infty} \theta_j \varepsilon_{t-j}$$

where $\{\varepsilon_t\}$ are independent and identical distribution (i.i.d.) random variables with finite variance. A nonlinear time series is one that does not satisfy this representation. To identify if there exists nonlinearity in a multivariate time series $\{\mathbf{x}_t\}$ with d variables, each univariate time series $\{\mathbf{x}_{it}\}$, $i = 1, \ldots, d$ is tested for linearity.

The Brock, Dechert, and Scheinkman (BDS) test [17, 18, 19] is designed to test the null hypothesis of i.i.d. To test for linearity, the BDS test is applied to the residuals of a linear model fitted on $\{\mathbf{x}_{it}\}$. If the null hypothesis is rejected, the model is misspecified. This departure can be interpreted as the detected presence of nonlinearity unaccounted for by the model.

Consider $\{x_t\}$, the correlation integral is estimated by

$$c_{m,n}(\epsilon) = \frac{2}{(n-m+1)(n-m)} \sum_{t=s+1}^{n} \sum_{s=m}^{n} \prod_{j=0}^{m-1} I_\epsilon(x_{t-j}, x_{s-j})$$

where n is the sample size, m is the embedding dimension, and I_ϵ is the indicator function

$$I_\epsilon = \begin{cases} 1 & \text{if } |x_{t-j} - x_{s-j}| < \epsilon \\ 0 & \text{otherwise} \end{cases}$$

which is used to estimate the joint probability of two m-dimensional points within a distance ϵ

$$\Pr(|x_t - x_s| < \epsilon, |x_{t-1} - x_{s-1}| < \epsilon, \ldots, |x_{t-m+1} - x_{s-m+1}| < \epsilon),$$

note that under the null hypothesis, $\mathrm{E}(c_{m,n}(\epsilon)) = (\mathrm{E}(c_{1,n}(\epsilon)))^m$.

The BDS test statistic is defined as

$$w_{m,n}(\epsilon) = \sqrt{n-m+1}\frac{c_{m,n}(\epsilon) - c^m_{1,n-m+1}(\epsilon)}{\sigma_{m,n}(\epsilon)}$$

where $\sigma_{m,n}(\epsilon)$ is the standard deviation of $\sqrt{n-m+1}(c_{m,n}(\epsilon) - c^m_{1,n-m+1}(\epsilon))$ that can be estimated as detailed in [17].

In the BDS test, we use the residuals $\{e_t\}$ of an Autoregressive Integrated Moving Average (ARIMA) model to test for i.i.d. The ARIMA(p,d,q) model is defined as

$$x'_t = \mu + \sum_{i=1}^{p} \alpha_i x'_{t-i} + \sum_{j=1}^{q} \beta_j x'_{t-j} + \varepsilon_t$$

where $\{x'_t\}$ is the d-th differenced series, p is the autoregressive (AR) order, and q is the moving average (MA) order [20]. To eliminate manual procedures in finding the optimal p, d, q, the Hyndman-Khandakar algorithm can be used. Hyndman-Khandakar algorithm is a stepwise search algorithm that combines AICc minimization and repeated KPSS tests to obtain the optimal p, d, q automatically [29].

4 Time Series Imaging

Imaging time series by encoding the raw time series into image representations is a useful tool in the analysis of temporal patterns and dominant characteristics. For instance, one may particularly be interested in characterizing these features in order to identify properties that might explain the behavior of a nonlinear deterministic system with respect to time. In seismological, neural, speech processing and related fields, spectrograms are used in to analyze and identify and characterize the the relationship between measurements and events in a time series using a frequency-domain representation.

Similarly, a recurrence plot (RP) is a 2-dimensional visualization tool introduced by Eckmann et al. [1] for dynamical systems and non-stationary time series that highlights the state recurrence in a multi-dimensional embedding called the phase space trajectory. It is primarily used for visual inspection of a system's topological and textural properties and its characterization using Recurrence Quantification Analysis (RQA). Figure 1 illustrates example recurrence plot representations generated from three arbitrary time series data.

The recurrence plot $\mathbf{R}$ of a univariate time series $\{x_t\}$ is generated from the delay vectors $\mathbf{x_i}$

$$\mathbf{x_i} = (x_i, x_{i+\tau}, .., x_{i+(m-1)\tau}),\ i = 1, \ldots, n-(m-1)\tau$$
$$\mathbf{R}_{i,j} = \mathrm{H}(\varepsilon - \|\mathbf{x}_i - \mathbf{x}_j\|), i,\ j = 1, \ldots, n-(m-1)\tau$$

where m is the embedding dimension, τ is the time delay, n is the length of the time series, ε is the threshold value, and H is the heaviside function.

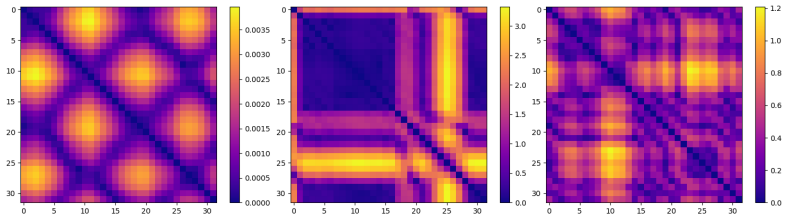

Fig. 1: Recurrence plots generated from time series data.

Time series imaging must be performed from time series data to produce representations that are suitable as input to a convolutional neural network. Recurrence plots can be used to generate such image representations. To motivate the use of recurrence plots over imaging techniques based on spectral analysis, any strong limitation in the alternative method (i.e. spectrogram) must be discussed.

Spectrograms assume signal stationarity. Intuitively, a stationary time series is a time series wherein its statistical characteristics do not vary over time (time-invariance). A less strict formal definition of stationarity (weak stationarity) is a time series $\{x_t\}$ that satisfies constant mean in all time steps $\mu = \mu_0,\ t = 1, \ldots, T$ and $\exists$f: $\mathbb{Z} \rightarrow \mathbb{R}$ such that $\mathrm{f}(\ell) = \mathrm{cov}(x_t, x_{t+\ell}), \forall t, \ell$. In other words, the covariance between x_t and $x_{t+\ell}$ depends only on lag difference ℓ for all t and ℓ [21]. Time-dependent variance and asymmetric cycles exist in nonlinear time series which violates the stationarity assumption. In many real-world time series, stationarity is not met, especially when the process exhibits nonlinearity. Pesaran [22] suggested that weakening the stationarity assumption to short-time stationarity is a reasonable procedure. Short-time stationarity assumes that stationarity exists within short time frames, and changes in longer time scales. This inevitably leads to a modelling implementation that requires manual analysis of the time series considered.

The addition of a manual procedure is generally undesirable since it unnecessarily complicates the process by requiring supervision which defeats the purpose of the proposed approach. Thus, the use of spectrograms is inappropriate for the purpose of time series modelling since it will, as a consequence of violating the stationarity assumption, be prone to information loss. On the other hand, the use of recurrence plots will allow the convolutional neural network to extract discriminative features based on its local and global spatial properties and structure arising from the representation, since it is designed to encode temporal information from a nonlinear time series.

4.1 Dimensionality Reduction

The size of a recurrence plot is proportional to the length of a given time series, which translates to an $O(n^2)$ memory complexity. Reducing a time series $\{x_t\}$ of arbitrary length n to a fixed length $\bar{n}$ such that $\bar{n} \ll n$ will drastically reduce the size of the recurrence plot. Piecewise Aggregate Approximation is a deterministic algorithm used for dimensionality reduction. In the context of time series, the dimensionality refers to the length of the time series. Piecewise Aggregate Approximation (PAA) [32] is applied to $\{x_t\}$ which generates the approximate time series $\{\overline{x}_t\}$ by

$$\bar{x}_i = \frac{\bar{n}}{n} \sum_{j=n/\bar{n}(i-1)+1}^{(n/\bar{n})i} x_j$$

The recurrence plot $\mathbf{R}$ is instead generated from $\{\bar{x}_t\}$. To this end, the chosen $\bar{n}$ for the Piecewise Aggregate Approximation algorithm can be thought of as a hyperparameter that controls the image size (dimensions) of a recurrence plot $\mathbf{R} \in \mathbb{R}^{\bar{n}\times\bar{n}}$. It is clear that as $\bar{n} \to n$, the greater amount of information is encoded to $\mathbf{R}$, which may improve the model's performance. Thus, $\bar{n}$ must be chosen carefully according to the hardware (memory) budget available and the problem's inherent complexity. If the nature of $\{x_t\}$ can be characterized as one that contains finer subtleties (local features), then $\bar{n}$ must be relatively large to effectively encode these information. Similarly, if $\{x_t\}$ contains larger and more global features, $\bar{n}$ can be smaller, as these features can be better captured in the approximation $\{\overline{x}_t\}$.

4.2 Optimal Parameters

In order to maintain a faithful image representation of a time series $\{x_t\}$, optimal recurrence plot parameters threshold value ε, embedding dimension m, and time delay τ specific to $\{x_t\}$ must be set to ensure that its respective recurrence plot reflects its dynamics appropriately. The threshold value ε dictates the amount of recurrences and recurrent structures that will be occur in the plot. For all variables, we set this to $\varepsilon = 0$, since determining the threshold value is a non-trivial task that heavily depends on the objective of the researcher [33]. To the best of our knowledge, literature proposing optimal threshold values only provides recommendations for (manual) recurrence analysis, not when recurrence plots are used in automated modelling approach that utilizes deep learning. Moreover, appropriate embedding dimension m and time delay τ must be set so as to avoid interruptions and artificial increase in diagonal lines and small blocks [33]. Non-optimal m and τ values also result in phase space (delay) vectors that are dominated by noise [31].

For time series from discrete-time maps, Cao recommends a time delay $\tau = 1$ [30]. Since the approximate time series $\{x_t\}$ generated by the Piecewise Aggregate Approximation algorithm can be interpreted as the mapping $\mathbb{R}^n \to \mathbb{R}^{\bar{n}}$, $\{x_t\}$ ap-

proximated as $\{\bar{x}_t\}$ qualifies as an output of a discrete-time map. Notice that setting $\tau = 1$ avoids large skips between elements x_i and $x_{i+\tau}$ in the delay vector $\mathbf{x}_i$ as these skips in x_t affect the granularity of information in $\mathbf{x}_i$. Relatively large skips as a consequence of setting $\tau > 1$ over $\{\bar{x}_t\}$ in generating $\mathbf{R}$ is problematic, since each element in $\{\bar{x}_t\}$ is a representative of a particular segment in $\{x_t\}$. In other words, setting $\tau = \tau_0 > 1$ translates to skipping $\tau_0 - 1$ points for every $\bar{x}_i$ in $\{\bar{x}_t\}$, generally leading to information loss.

Finally, following the findings of Iwanski and Bradley [7] that recurrence plots of experimental data appear to be independent of the embedding dimension m, it is set to $m = 1$. Iwanski and Bradley demonstrated that increasing m does not affect the recurrence plot both qualitatively and quantitatively. This is an important finding because pattern recognition is at the core of using deep learning. If topological structures remain consistent or very similar across varied m, then the process of identifying an appropriate value for m using various heuristics can be avoided. This significantly simplifies the overall modelling process since the optimal embedding dimension and time delay values $m = 1, \tau = 1$ are identified with a theoretical and experimental basis without the need for any further estimation.

5 Convolutional Neural Networks

Motivated by the need to adapt traditional neural networks to image classification and regression tasks, convolutional neural networks were developed. Consider a classification task wherein the input is an $n \times n$ image flattened to form a vector of length n^2. Let $\mathrm{F}_{\mathrm{ANN}}$ be a vanilla neural network with $n_0 = n^2$ input units. For each layer in a forward pass, the matrix multiplication $\mathbf{\Theta}^\top \mathbf{x}$ of layer's weights $\mathbf{\Theta} \in \mathbb{R}^{n \times m}$ and input from the previous layer $\mathbf{x} \in \mathbb{R}^{n \times 1}$ is an $O(mn)$ operation. When n_0 is very large, the matrix multiplication in the immediate hidden layer becomes an issue in terms of both memory and time complexity. In addition, suppose that a binary image I represented by a matrix contains a local structure represented by a group of 1's centered at (x_0, y_0) and a similar image I' containing the same local structure translated to a position centered at $(x_0 - c, y_0 - c)$. $\mathrm{F}_{\mathrm{ANN}}$, due to its lack of tolerance to translation, would treat I and I' as images containing their own unique local structure. Thus, computing for $\mathbf{y} = h(\mathbf{\Theta}^\top \mathbf{x} + \mathbf{b})$ using I versus computing $\mathbf{y}$ using I' would yield different outputs. As a consequence, $\mathrm{F}_{\mathrm{ANN}}$ will not efficiently recognize similar and redundant patterns across instances in the data resulting to the need to increase the model's complexity by means of increasing the number of hidden units and layer depth.

These issues can be resolved with the use of a convolutional neural network due to its properties that are useful for learning spatial data: *sparse interactions*, *weight sharing*, and *equivariance*. These are made possible by introducing *convolutional* and *pooling* layers.

A convolutional layer is based on the discrete *convolution* operation. A discrete convolution denoted by $\circledast$ is defined as

$$[f \circledast g](i) = \sum_a f(a)g(i-a)$$

For 2-dimensional inputs such as images, we define the discrete convolution over the two axes of an image $\mathbf{I}$ [34, 35]

$$[\mathbf{I} \circledast \mathbf{K}](i,j) = \sum_p \sum_q \mathbf{I}(i,j)\mathbf{K}(u-p, v-q)$$

where $\mathbf{K} \in \mathbb{R}^{k \times k}$ is called the *kernel*, a matrix of learnable weights. Convolutions are commutative, so the above equation is equivalent to

$$[\mathbf{I} \circledast \mathbf{K}](i,j) = \sum_p \sum_q \mathbf{I}(i-p, j-q)\mathbf{K}(u,v)$$

By making kernel $\mathbf{K}$ smaller than $\mathbf{I}$, sparse connectivity is achieved, which reduces the time complexity to $O(kn)$ versus $O(mn)$, when $k \ll m$. This is also an improvement in memory complexity, since it drastically reduces the number of parameters that are needed to be stored. Another advantage of using convolutions is parameter sharing, which allows for the use of the same parameters for different inputs. As a consequence, parameter sharing in a convolution causes the convolutional layer to be *equivariant*. Mathematically, a function f is equivariant to a function g if $f(g(x)) = g(f(x))$. This makes the convolutional neural network tolerant to translations in $\mathbf{I}$, in contrast to a vanilla neural network that do not reuse weights, which also is a major advantage in memory complexity.

Furthermore, as the depth of a convolutional neural network increases, it is desirable to reduce the sizes of outputs per layer in order to reduce the computational and statistical burden of succeeding layers. The reduction is a downsampling operation accomplished using a pooling layer. A pooling operation essentially takes the maximum (max pooling) or the average (average pooling) of a submatrix $\mathbf{x}_{\subseteq} \in \mathbb{R}^{s \times s}$ of an input matrix $\mathbf{x} \in \mathbb{R}^{H \times W}$ and placing it as a cell in an output matrix $\mathbf{z} \in \mathbb{R}^{H' \times W'}$ collecting maximum or average values per submatrix. Let the submatrix positioned topmost left at (w,h) be denoted by $\mathbf{x}_{w:w+s,h:h+s}$. Max and average pooling are defined as the following, respectively

$$\mathbf{z}_{i',j'} = \max_{0 \le i,j < s} \{\mathbf{x}_{i':H+i,j':W+j}\}$$

$$\mathbf{z}_{i',j'} = \frac{1}{s^2} \sum_{0 \le i,j < s} \{\mathbf{x}_{i'+i,j'+j}\}$$

In the same way that the layers in a vanilla neural network are stacked, a convolutional neural network is stacked from left to right, incorporating the convolutional and pooling layers

$$\mathbf{x} \to \boxed{\circledast} \underset{\phi}{\to} \boxed{\odot} \to \ldots \to \boxed{\circledast} \underset{\phi}{\to} \boxed{\odot} \underset{\text{flat}}{\longrightarrow} \boxed{\oplus} \underset{\phi}{\to} \ldots \underset{\phi}{\to} \boxed{\oplus} \underset{\text{lin}}{\longrightarrow} \mathbf{y}$$

In the example above, input $\mathbf{x}$ is transformed by a series of convolutional ($\circledast$) and pooling ($\odot$) layers mediated by a nonlinear activation function ϕ and flattened (*flat*) and connected to a typical fully-connected layer ($\oplus$) where the final layer outputs $\mathbf{y}$ with a linear activation function (*lin*). For the sake of simplicity in presentation, $\mathbf{x}$ is a *single-channel* input. For multivariate inputs, $\mathbf{x} \in \mathbb{R}^{H \times W}$ is replaced with an tensor $\mathbf{X} \in \mathbb{R}^{H \times W \times D}$, where the last axis represents the number of variables D (channels). The process essentially remains the same despite the inclusion of the channel axis to handle the multivariate case.

6 Model Pipeline and Architecture

For convenience, we describe the pipeline using a training instance from the dataset $\mathcal{D} = \{(\mathbf{X}_{t-k:t}, \mathbf{x}_{t+1})\}$. The procedure begins with the transformation of a multivariate time series window $\{\mathbf{X}_i\}_{t-k:t}$ into a tensor $\mathbf{X}_{t-k:t} \in \mathbb{R}^{R \times R \times D}$ containing recurrence plots of each variable where R is the image size and D is the number of variables. This is done by applying Piecewise Aggregate Approximation and time series imaging using recurrence plots to each univariate time series $\{\mathbf{X}_{it}\}_{t-k:t}$ representing variable i. Next, the model takes $\mathbf{X}_{t-k:t}$ as input and is forward-propagated to obtain the output $\hat{\mathbf{x}}_{t+1}$, which is the model's prediction of the label vector containing one-step ahead values for each variable $\mathbf{x}_{t+1} = \{\mathbf{X}_t\}_{t:t+1} \in \mathbb{R}^{D \times 1}$.

6.1 Architecture

The proposed model architecture is outlined in table 1. The model consists of two convolutional and fully-connected (FC) layers. Batch normalization [36] is used to accelerate model training and acts as an additional regularization component by normalizing layer inputs. Dropouts [37] are also included to reduce the risk of overfitting by randomly dropping nodes along with their connections during training.

Type	Size/rate	# filters/nodes	Activation
Conv + Batch norm	(3×3)	32	relu
Max pooling	(2×2)	-	-
Conv + Batch norm	(3×3)	32	relu
Max pooling	(2×2)	-	-
Dropout	0.25	-	-
Flatten	-	-	-
FC	-	256	relu
Dropout	0.25	-	-
FC	-	D	linear

Table 1: Proposed model architecture

7 Experimental Setup

The model is compared against the following state-of-the-art models: ResNet [38], Fully Convolutional Networks [39], and InceptionTime [40]. We trained three models, CNN-16, CNN-32, CNN-64, each corresponds to the same model architecture trained with different image size configurations $[16, 32, 64]$. The following models are used to test if there is an improvement in performance if the same generic model architecture is used We used the following datasets from `timeseriesregression.org` [41]:

Dataset	No. of variables with $p < 0.05$
BIDMC32HR	2
BenzeneConcentration	5
HouseholdPowerConsumption1	5
IEEEPPG	2
LiveFuelMoistureContent	7
NewsHeadlineSentiment	2
PPGDalia	4

Table 2: Datasets

The criteria for dataset selection is based on the following:

- Dimensions (no. of variables) must at least be greater than 1.
- For convenience, time series length must at least be greater than 64. This allows for the use of the CNN-64 model without any extrapolation procedures on datasets with a length less than 32.
- Training size must at least contain 1000 instances.

- Datasets containing essentially the same training samples are discarded. For instance, BIDMC32HR, BIDMC32RR and BIDMC32SpO2 only differs in their labels. Hence, only BIDMC32HR is retained.
- At least one variable exhibits nonlinearity.

Minimal data preprocessing has been performed in the datasets. In the case of missing values, linear interpolation was performed. The BDS test (discussed in section 3) was applied to each dataset to determine the number of variables that exhibit nonlinearity.

Each model is trained with a batch size of 1024 for 1000 epochs. Early stopping and plateau-triggered learning rate reduction is performed to prevent overfitting. Once the model is trained, it is validated against the testing set with Root Mean Squared Error (RMSE) as the metric averaged over d variables

$$\text{RMSE}_{ave} = \frac{1}{d}\sqrt{\frac{\sum(\hat{\mathbf{x}}_{t+1} - \mathbf{x}_{t+1})^2}{n}}$$

To ensure consistency, for each dataset we trained each model for 5 runs and averaged the results. To compare the models, we performed Friedman's test [42], which is used to detect any differences in average ranks between models. In case of ties, fractional ranking is used. Given k models and b different datasets, it tests for the null hypothesis $H_0 : M_1 = M_2 = \ldots = M_k$ and the alternative hypothesis $H_a : M_i' \neq M_i$ for at least one $i' \neq i$ where M_i's are the median responses associated with the k models. Let $R_i j$ be the rank of the i-th model within the j-th dataset and let R_i be the sum of the ranks for the i-th model for all datasets. The Friedman test statistic is computed by

$$\chi_F^2 = \frac{12}{bk(k+1)}\sum_{i=1}^{k}\left[R_i - \frac{b(k+1)}{2}\right]^2$$

where $\frac{b(k+1)}{2} = E_0(R_i)$ is the mean of the R_i's when H_0 is true. If H_0 is false, then a post-hoc test called the Nemenyi test is conducted to identify which model(s) perform significantly different among others. The Nemenyi test [43] computes for the critical difference

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}}$$

Any model that differs by at least CD with $\alpha = 0.05$ outperforms other models based on the average ranks shown in a critical difference diagram.

`Tensorflow` was used to implement the models. ResNet, FCN, and InceptionTime implementations are adapted from [41]. We also used the `pyts` package to implement recurrence plots and the Piecewise Aggregate Approximation algorithm. The BDS test is implemented using the `statsmodels` package and the automatic ARIMA hyperparameter optimization using the `pmdarima` package. All of these were implemented in Python 3.8. The implementation is available at `https://github.com/saojeda/rcnn-experiments`.

8 Results

Table 3 shows the results of the experiments. Based on this table, we computed the Friedman test statistic and corresponding p-value. Friedman's test show that H_0 cannot be rejected ($p = 0.69220 > 0.05$), so the post-hoc Nemenyi test used to visualize the critical difference diagram is not conducted.

	CNN-16	CNN-32	CNN-64	FCN	InceptionTime	ResNet
BIDMC32HR	0.35622	0.34296	0.36488	0.10863	0.13123	**0.10863**
BConcentration	292.79553	258.91063	**241.59186**	600.77448	659.98979	780.95314
HPConsumption1	4.66115	**4.08298**	4.12137	101.34325	95.23767	103.55190
IEEEPPG	42.15604	43.63521	41.84182	63.89978	**26.56151**	45.42202
LFMContent	1575.85710	1414.00837	**1223.33757**	1835.93391	1927.07244	1835.93391
NHSentiment	**14.98553**	15.79095	16.04000	15.01115	14.98857	14.99808
PPGDalia	55.00934	51.68706	43.66140	**1.61548**	1.62459	**1.61548**

Table 3: Model training results showing the RMSE averaged over 5 runs of a model in a particular dataset. Long dataset names are shortened.

The models CNN-16, CNN-32, CNN-64 do not outperform FCN, InceptionTime, and ResNet. However, the CNN models' performance are still on par with these state-of-the-art time series models in one-step ahead forecasting tasks. It is worth noting that the CNN models appears to perform better in datasets resembling discrete-time maps. It is evident that the FCN, InceptionTime, and ResNet models perform better compared the CNN models in the datasets BIDMC32HR, IEEEPPG, and PPGDalia which are sampled in an interval resembling continuous-time. It is reasonable to think that these models may have local structures that are not sufficiently captured by the Piecewise Aggregate Approximation algorithm given length $\bar{n}$. The CNN models may perform better with a larger $\bar{n}$, but available memory resource can be a constraint. In these cases, it is desirable to use the alternative SoTA models. The CNN models were able to provide significant boosts in performance particularly in datasets that are by nature sampled in discrete-time, with several orders of magnitude of improvement versus SoTA time series models.

9 Conclusion

In this work, we presented a framework that makes use of recurrence plots in order to perform multivariate nonlinear one-step ahead forecasts that can be applied in real-world use cases. Raw time series data is first transformed into recurrence plot representations, then these image representations are learned by a generic convolutional neural network model to recognize discriminative features based from the topological structure of the input recurrence plots. We have shown that the convo-

lutional neural network is able to learn these patterns and consequently is able to produce forecasts in an automated manner that does not require manual data analysis and considerable preprocessing. In the future, the authors may work on extending this approach to more specialized convolutional neural network architectures that can exploit the topological properties of recurrence plots to further improve its performance.

References

1. Eckmann JP, Oliffson Kamphorst S, Ruelle D (1987) Recurrence Plots of Dynamical Systems. Europhys. Lett. 4(9):973–977 doi: 10.1209/0295-5075/4/9/004
2. Casdagli MC (1997) Recurrence Plots Revisited. Physica D 108(1-2):12–44 doi: 10.1016/S0167-2789(97)82003-9
3. Thiel M, Romano MC, Kurths J (2004) How much information is contained in a recurrence plot? Physics Letters A 330(2004):343–349 doi: 10.1016/j.physleta.2004.07.050
4. Liebert W, Schuster HG (1989) Proper Choice of the Time Delay for the Analysis of Chaotic Time Series. Physics Letters A 142(2,3):107–111 doi: 10.1016/0375-9601(89)90169-2
5. Fraser AM, Swinney HG (1986) Independent coordinates for strange attractors from mutual information. Phys Rev A 33(2):1134 doi: 10.1103/PhysRevA.33.1134
6. Kennel MB, Brown R, Abarbanel HDI (1992) Determining embedding dimension for phase-space reconstruction using a geometrical construction. Phys. Rev. A 45, 3403 (1992). doi: 10.1103/PhysRevA.45.3403
7. Iwanski JS, Bradley E (1998) Recurrence Plots of Experimental Data: To Embed Or Not to Embed. Chaos, 8:861–871 doi: 10.1063/1.166372
8. Gao J, Cai H (2000) On the structures and quantification of recurrence plots. Physics Letters A 270(1–2):75–87 doi: 10.1016/S0375-9601(00)00304-2
9. Gautama T, Mandic DP, Van Hulle MM (2003) A differential entropy based method for determining the optimal embedding parameters of a signal. In: 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)., Hong Kong, 2003, pp. VI-29. doi: 10.1109/ICASSP.2003.1201610
10. Marwan N, Kurths J, Saparin P (2007) Generalised recurrence plot analysis for spatial data. Physics Letters A 360 (2007):545–551 doi: 10.1209/0295-5075/4/9/004
11. Sipers A, Borm P, Peeters R (2011) On the unique reconstruction of a signal from its unthresholded recurrence plot. Physics Letters A 375 (2011):2309–2321 doi: 10.1016/j.physleta.2011.04.040
12. Penatti OAB, Santos MFS (2017) Human activity recognition from mobile inertial sensors using recurrence plots. ArXiv. Available via arXiv. `https://arxiv.org/pdf/2004.13408.pdf`. Accessed 31 Aug 2020
13. Marwan N (2008) A historical review of recurrence plots. The European Physical Journal Special Topics 164(2008):3–12 doi: 10.1209/0295-5075/4/9/004
14. Lim B, Zohren S (2020) Time series forecasting with deep learning: A survey. ArXiv. Available via arXiv. `https://arxiv.org/pdf/2004.13408.pdf`. Accessed 24 Aug 2020
15. Sotoca JM, Sánchez JS, Mollineda RA (2005) A review of data complexity measures and their applicability to pattern classification problems. In: Actas del III Taller Nacional de Mineríade Datos y Aprendizaje, TAMIDA 2005. p 77–83
16. Li L, Abu-Mostafa YS (2006) Data Complexity in Machine Learning. Caltech Computer Science Technical Reports. doi: 10.7907/Z9319SW2.
17. Brock WA, Scheinkman JA, Dechert WD, LeBaron B (1996) A test for independence based on the correlation dimension. Econometric Reviews 15(3):197–235
18. Belaire-Franch J, Contreras D (2002) How to compute the BDS test: A software comparison. J. Appl. Econ. 17:691–699. doi: 0.1002/jae.679
19. Zivot E (n.d.) Nonlinear Time Series Models. `https://faculty.washington.edu/ezivot/econ584/notes/nonlinear.pdf`. Accessed 24 Aug 2020
20. Hyndman RJ, Athanasopoulos G (2018) ARIMA Models. In: Forecasting: principles and practice, 2nd edn. OTexts, Melbourne, Australia. p 221–274
21. Rao SS (2020) A course in time series analysis. `https://www.stat.tamu.edu/~suhasini/teaching673/time_series.pdf`. Accessed 29 Aug 2020
22. Pesaran B (2020) Spectral Analysis for Neural Signals. `https://neurophysics.ucsd.edu/courses/physics_173_273/SfN_Short_Course_c1.pdf`. Accessed 29 Aug 2020
23. Wang Z, Oates T (2015) Visual Interpretation of Hand Gestures for Human Computer Interaction: A Review. Paper presented at the Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 25-31 July 2015

24. Hatami N, Gavet Y, Debayle J (2017) Paper presented at the Tenth International Conference on Machine Vision, Vienna, Austria, 13-15 November 2017
25. Hsueh Y, Ittangihala VR, Wu W et al (2019) Condition Monitor System for Rotation Machine by CNN with Recurrence Plot. Energies 12:3221. doi: 10.3390/en12173221
26. Zhao Z, Zhang Y, Comert Z, Deng Y (2019) Computer-Aided Diagnosis System of Fetal Hypoxia Incorporating Recurrence Plot With Convolutional Neural Network. Frontiers in Physiology. doi: 10.3389/fphys.2019.00255
27. Garcia-Ceja E, Uddin MZ, Torresen J (2018) Classification of Recurrence Plots' Distance Matrices with a Convolutional Neural Network for Activity Recognition. Procedia Computer Science 130:157–163. doi: 10.1016/j.procs.2018.04.025
28. Ojeda SAA, Solano GA, Peramo EC (2020) Multivariate Time Series Imaging for Short-Term Precipitation Forecasting Using Convolutional Neural Networks. 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). doi: 10.1109/icaiic48513.2020.9065238
29. Hyndman RJ, Khandakar Y (2008) Automatic Time Series Forecasting: TheforecastPackage forR. Journal of Statistical Software. doi: 10.18637/jss.v027.i03
30. Cao L (1997) Practical method for determining the minimum embedding dimension of a scalar time series. Physica D: Nonlinear Phenomena 110:43–50. doi: 10.1016/s0167-2789(97)00118-8
31. Gautama T, Mandic D, Hulle MV (2003) A differential entropy based method for determining the optimal embedding parameters of a signal. 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003 Proceedings (ICASSP '03). doi: 10.1109/icassp.2003.1201610
32. Keogh E, Chakrabarti K, Pazzani M, Mehrotra S (2001) Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Knowledge and Information Systems 3:263–286. doi: 10.1007/pl00011669
33. Marwan N (2011) How To Avoid Potential Pitfalls In Recurrence Plot Based Data Analysis. International Journal of Bifurcation and Chaos 21:1003–1017. doi: 10.1142/s0218127411029008
34. Zhang A, Lipton Z, Li M et al (n.d.) 6. Convolutional Neural Networks. In: 6. Convolutional Neural Networks - Dive into Deep Learning 0.14.3 documentation. `http://d2l.ai/chapter_convolutional-neural-networks/index.html`. Accessed 31 Aug 2020
35. Goodfellow I, Bengio Y, Courville A (2017) Convolutional Networks. In: Deep learning. The MIT Press, Cambridge, MA, pp 330–371
36. Ioffe S, Szegedy C (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift 37:448–456
37. Srivastava N, Hinton G, Krizhevsky A (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting 15(56):1929–1958
38. He K, Zhang X, Ren S, Sun J (2016) Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi: 10.1109/cvpr.2016.90
39. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi: 10.1109/cvpr.2015.7298965
40. Hassan IF, Lucas B, Forestier G, Pelletier C et al (2019) InceptionTime: Finding AlexNet for Time Series Classification. arXiv preprint arXiv:1909.04939
41. Tan CW, Bergmeir C, Petitjean F and Webb GI (2020) Time Series Regression. arXiv preprint arXiv:2006.12672
42. Borkowski J (2014) Blocked Designs. `http://www.math.montana.edu/jobo/thainp/ksampa.pdf` Accessed 31 August 2020
43. Demsar J (2006) Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7(Jan):1–30