

# Autoregressive Models for Sequences of Graphs

Daniele Zambon\*

Università della Svizzera italiana, Lugano, Switzerland  
daniele.zambon@usi.ch

Lorenzo Livi

University of Manitoba, Winnipeg, Canada  
University of Exeter, Exeter, United Kingdom

Daniele Grattarola\*

Università della Svizzera italiana, Lugano, Switzerland  
daniele.grattarola@usi.ch

Cesare Alippi

Università della Svizzera italiana, Lugano, Switzerland  
Politecnico di Milano, Milano, Italy

**Abstract**—This paper proposes an autoregressive (AR) model for sequences of graphs, which generalises traditional AR models. A first novelty consists in formalising the AR model for a very general family of graphs, characterised by a variable topology, and attributes associated with nodes and edges. A graph neural network (GNN) is also proposed to learn the AR function associated with the graph-generating process (GGP), and subsequently predict the next graph in a sequence. The proposed method is compared with four baselines on synthetic GGPs, denoting a significantly better performance on all considered problems.

**Index Terms**—graph, structured data, stochastic process, recurrent neural network, graph neural network, autoregressive model, prediction.

## I. INTRODUCTION

Several physical systems can be described by means of autoregressive (AR) models and their variants. In that case, a system is represented as a discrete-time signal in which, at every time step, the observation is modelled as a realisation of a random variable that depends on preceding observations. In this paper, we consider the problem of designing AR predictive models where the observed entity is a graph.

In the traditional setting for AR predictive models, each observation generated by the process is modelled as a vector, intended as a realisation of random variable  $x_{t+1} \in \mathbb{R}^d$ , so that

$$\begin{cases} x_{t+1} = f(x_t, x_{t-1}, \dots, x_{t-p+1}) + \epsilon, \\ \mathbb{E}[\epsilon] = 0, \\ \text{Var}[\epsilon] = \sigma^2 < \infty. \end{cases} \quad (1)$$

In other terms, each observation  $x_{t+1}$  is obtained from the regressor

$$\mathbf{x}_t^p = [x_t, x_{t-1}, \dots, x_{t-p+1}]$$

through an AR function  $f(\cdot)$  of order  $p$  affected by an additive stationary random noise  $\epsilon$ . Given (1), the prediction of  $x_{t+1}$  is often taken as

$$\hat{x}_{t+1} = \mathbb{E}_\epsilon[x_{t+1}] = f(x_t, x_{t-1}, \dots, x_{t-p+1}), \quad (2)$$

This research is funded by the Swiss National Science Foundation project 200021\_172671. We gratefully acknowledge partial support of the Canada Research Chairs program. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

\*Equal contribution.

where the expectation  $\mathbb{E}_\epsilon[\cdot]$  is with respect to noise at time  $t+1$ . The predictor from (2) is optimal when considering the  $L^2$ -norm loss between  $\hat{x}_{t+1}$  and  $x_{t+1}$ .

Predicting the temporal evolution of a multivariate stochastic process is a widely explored problem in system identification and machine learning. However, several recent works have focused on problems where the vector representation of a system can be augmented by also considering the relations existing among variables [1], ending up with a structured representation that can be naturally described by a graph. Such structured information provides an important inductive bias that the learning algorithms can take advantage of. For instance, it is difficult to predict the chemical properties of a molecule by only looking at its atoms; on the other hand, by explicitly representing the chemical bonds, the description of the molecule becomes more complete, and the learning algorithm can take advantage of that information. Similarly, several other machine learning problems benefit from a graph-based representation, e.g., the understanding of visual scenes [2], the modelling of interactions in physical and multi-agent systems [3], [4], or the prediction of traffic flow [5]. In all these problems (and many others [1]), the dependencies among variables provide a strong prior that has been successfully leveraged to significantly surpass the previous state of the art.

In this paper, we consider attributed graphs where each node and edge can be associated with a feature vector. As such, a graph with  $N$  nodes can be seen as a tuple  $(\mathcal{V}, \mathcal{E})$ , where

$$\mathcal{V} = \{v_i \in \mathbb{R}^F\}_{i=1, \dots, N}$$

represents the set of nodes with  $F$ -dimensional attributes, and

$$\mathcal{E} = \{e_{ij} \in \mathbb{R}^S\}_{v_i, v_j \in \mathcal{V}}$$

models the set of  $S$ -dimensional edge attributes [6]. By this formulation also categorical attributes can be represented, via vector encoding. We denote with  $\mathcal{G}$  the set of all possible graphs with vector attributes. Graphs in  $\mathcal{G}$  can have different order and topology, as well as variable node and edge attributes. Moreover, a correspondence among the nodes of different graphs might not be present, or can be unknown; in this case, we say that the nodes are *non-identified*.

Here we build on the classic AR model (1) operating on discrete-time signals, and propose a generalised formulation

that allows us to consider sequences of generic attributed graphs. The formulation is thus adapted by modelling a graph-generating process (GGP) that produces a sequence  $\{g_1, g_2, \dots, g_t, \dots\}$  of graphs in  $\mathcal{G}$ , where each graph  $g_{t+1}$  is obtained through an AR function

$$\phi(\cdot) : \mathcal{G}^p \rightarrow \mathcal{G}. \quad (3)$$

on the space  $\mathcal{G}$  of graphs. In order to formulate an AR model equivalent to (1) for graphs, the AR function  $f(\cdot)$  must be defined on the space of graphs, and the concept of additive noise must also be suitably adapted to the graph representation.

Previous art on predictive autoregression in the space of graphs is scarce, with [7] being the most notable contribution to the best of our knowledge. However, the work in [7] only considers binary graphs governed by a vector AR process of order  $p = 1$  on some graph features, and does not allow to consider attributes or non-identified nodes.

The novel contribution of this work is therefore two-fold. First, we formulate an AR system model generating graph-valued signals as a generalisation of the numerical case. In this formulation, we consider graphs in a very general form. In particular, our model deals with graphs characterised by:

- directed and undirected edges;
- identified and non-identified nodes;
- variable topology (i.e., connectivity);
- variable order;
- node and edge attributes (not necessarily numerical vectors, but also categorical data is possible).

Our model also accounts for the presence of an arbitrary stationary noise, which is formulated as a graph-valued random variable that can act on the topology, node attributes, and edge attributes of the graphs.

Second, we propose to learn the AR function (3) using the recently proposed framework of graph neural networks (GNNs) [1], a family of learnable functions that are designed to operate directly on arbitrary graphs. This paper represents a first step towards modelling graph-valued stochastic processes, and introduces a possible way of applying existing tools for deep learning to this new class of prediction problems.

The rest of the paper is structured as follows: Section II-A formulates the problem of defining AR models in the space of graphs and Section II-B introduces the proposed architecture for the GNN; Section III reports the experimental analysis performed to validate the methodology.

## II. NEURAL GRAPH AUTOREGRESSION

### A. Autoregressive Model for a GGP

Due to the lack of basic mathematical operators in the vast family of graphs we consider here, the generalisation of model (1) to account for graph data is non-trivial. For instance, we have to deal with the fact that the sum between two graphs is not defined, although it can be formulated in some particular cases, e.g., when two graphs have the same order, identified nodes and numerical attributes [8].

Let  $\{g_1, g_2, \dots, g_t, \dots\}$  be a discrete signal where each sample is described by a graph data structure. As done for

the numerical case, we model each observation of the process as a realisation of a random variable  $g_{t+1} \in \mathcal{G}$ , dependent on a graph-valued regressor

$$\mathbf{g}_t^p = [g_t, g_{t-1}, \dots, g_{t-p+1}],$$

through an AR function  $\phi(\cdot)$  on the space of graphs. Similarly to (1),  $g_{t+1}$  is modelled as

$$g_{t+1} = H(\phi(\mathbf{g}_t^p), \eta), \quad (4)$$

where

$$H : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G}.$$

is a function that models the effects of noise graph<sup>1</sup>  $\eta \in \mathcal{G}$  on graph  $\phi(\mathbf{g}_t^p)$ . Function  $H(\cdot, \cdot)$  in (4) is necessary because, as mentioned above, the sum between graphs  $\phi(\mathbf{g}_t^p)$  and  $\eta$  is not generally defined.

Assumptions made on the noise in (1) have to be adapted as well. In the classic formulation (1), the condition of unbiased noise is formulated as  $\mathbb{E}[\epsilon] = 0$  or, equivalently, as  $f(x_t, \dots, x_{t-p-1}) = \mathbb{E}_\epsilon[f(x_t, \dots, x_{t-p-1}) + \epsilon]$ . In the space of graphs the assumption of unbiased noise can be extended as

$$\phi(\mathbf{g}_t^p) \in \mathbb{E}_\eta^f[H(\phi(\mathbf{g}_t^p), \eta)], \quad (5)$$

where  $\mathbb{E}^f[\cdot] \in \mathcal{G}$  is the set of mean graphs according to Fréchet [10], defined as the set of graphs minimising:

$$\mathbb{E}^f[g] = \arg \min_{g' \in \mathcal{G}} \int_{\mathcal{G}} d(g, g')^2 dQ_g(g). \quad (6)$$

Function  $d(\cdot, \cdot)$  in (6) is a pre-metric distance between two graphs, and  $Q_g$  is a graph distribution defined on the Borel sets of space  $(\mathcal{G}, d)$ . Examples of graph distances  $d(\cdot, \cdot)$  that can be adopted are the graph edit distances [11]–[13], or any distance derived by positive semi-definite kernel functions [14]. Depending on the graph distribution  $Q_g$ , there can be more than one graph minimising (6), hence ending up with  $\mathbb{E}^f[\cdot]$  as a set. We stress that for a sufficiently small Fréchet variation of the noise graph  $\eta$  (see Eq. (7)) and a metric  $d(\cdot, \cdot)$ , the Fréchet mean graph exists and is unique.

Equation (6) holds only when

$$\int_{\mathcal{G}} d(g, g')^2 dQ_g(g) < \infty,$$

which can be interpreted as the graph counterpart of  $\text{Var}[\epsilon] < \infty$  in (1). The variance in the graph domain can be expressed in terms of the Fréchet variation:

$$\text{Var}^f[g] := \min_{g' \in \mathcal{G}} \int_{\mathcal{G}} d(g, g')^2 dQ_g(g) < \infty. \quad (7)$$

The final AR system model in the graph domain becomes

$$\begin{cases} g_{t+1} = H(\phi(\mathbf{g}_t^p), \eta), \\ \phi(\mathbf{g}_t^p) \in \mathbb{E}_\eta^f[H(\phi(\mathbf{g}_t^p), \eta)], \\ \text{Var}^f[\eta] < \infty. \end{cases} \quad (8)$$

<sup>1</sup>In the present paper, with noise graph we mean a graph-valued random variable distributed according to a stationary graph distribution [9].

Notice that the proposed graph autoregressive model (8) is a proper generalisation of model (1). In fact, it can be shown that (8) reduces to (1), when considering  $(\mathbb{R}, |\cdot|)$  – or more generally,  $(\mathbb{R}^d, |\cdot|_2)$  – in place of  $(\mathcal{G}, d)$ , and choosing  $H(a, b)$  as the sum  $a + b$  (see Appendix A for a proof).

Given a GGP modelled by (8), we can predict graph at time  $t + 1$  as that graph  $\hat{g}_{t+1}$  minimising quantity

$$\mathbb{E} \left[ d(g, g_{t+1})^2 \right]$$

where the expectation is taken with respect to  $g_{t+1}$ . Therefore, we obtain that the optimal prediction is attained at graph

$$\begin{aligned} \hat{g}_{t+1} &= \arg \min_{g \in \mathcal{G}} \mathbb{E} \left[ d(g, g_{t+1})^2 \right] \\ &= \mathbb{E}_\eta^f [H(\phi(\mathbf{g}_t^p), \eta)] = \phi(\mathbf{g}_t^p). \end{aligned} \quad (9)$$

### B. Learning the AR function with a Graph Neural Network

Given a GGP described by an AR model of order  $p$ , the task of predicting the next graph in the sequence can be formulated as that of approximating  $\phi(\cdot)$  in (8), as the optimal prediction is given by (9). In order to approximate  $\phi(\cdot)$  we propose to use a GNN, which can be seen as a family of models

$$\phi_{nn}(\cdot; \theta) : \mathcal{G}^k \rightarrow \mathcal{G}$$

parametrised in vector  $\theta$ . The family of models receives regressor  $\mathbf{g}_t^k = [g_t, \dots, g_{t-k+1}]$  and outputs the predicted graph  $\hat{g}_{t+1}$

$$\hat{g}_{t+1} = \phi_{nn}(\mathbf{g}_t^k; \theta). \quad (10)$$

As the order  $p$  is usually unknown, the value  $k$  is a hyperparameter that must be appropriately chosen.

We propose a possible architecture for  $\phi_{nn}$ , composed of three main blocks:

- 1) each graph is mapped to an abstract vector representation, using graph convolutions [16]–[18] and graph pooling [15], [19] layers;
- 2) a predictive model is applied to the resulting vector sequence;
- 3) the predicted graph is obtained by mapping the predicted vector back to the graph domain.

The full GNN  $\phi_{nn}$  is therefore obtained by the composition of three blocks, denoted  $\phi_{conv}$ ,  $\phi_{rnn}$ , and  $\phi_{dense}$ . Although each block has its own parameters  $\theta_{conv}$ ,  $\theta_{rnn}$  and  $\theta_{dense}$ , the network can be trained end-to-end as a single family of models with  $\theta = \{\theta_{conv}, \theta_{rnn}, \theta_{dense}\}$ . The three blocks are defined as follows (see also Figure 1 for a schematic view of the architecture).

The first block of the network converts the input sequence  $\mathbf{g}_t^k$  to a sequence of  $l$ -dimensional vectors. This operation can be described by map:

$$\phi_{conv}(\cdot; \theta_{conv}) : \mathcal{G} \rightarrow \mathbb{R}^l,$$

which is implemented as a GNN alternating graph convolutional layers for extracting local patterns, and graph pooling layers to eventually compress the representation down to a single vector. By mapping graphs to a vector space, we go back to the numerical setting of (1). Moreover, the vector

representation of the graphs takes into account the relational information that characterises the GGP.

By applying  $\phi_{conv}$  to each graph in the regressor  $\mathbf{g}_t^k$  (with the same  $\theta_{conv}$ ), we obtain a sequence

$$[\phi_{conv}(g_t; \theta_{conv}), \dots, \phi_{conv}(g_{t-k+1}; \theta_{conv})]$$

of  $l$ -dimensional vectors, which is then processed by block

$$\phi_{rnn}(\cdot; \theta_{rnn}) : \mathbb{R}^{k \times l} \rightarrow \mathbb{R}^l.$$

The role of this block is to produce the vector representation of the predicted graph, while also capturing the temporal dependencies in the input sequences. Here, we formulate the block as a recurrent network, but any method to map the sequence to a prediction is suitable (e.g., fully connected networks).

Finally, we convert the vector representation to the actual prediction in the space of graphs, using a multi-head dense network similar to the ones proposed in [20], [21]:

$$\phi_{dense}(\cdot; \theta_{dense}) : \mathbb{R}^l \rightarrow \mathcal{G}.$$

Note that generating a graph by sampling its topology and attributes with a dense network has known limitations and implicitly assumes node identity and a maximum order. While this solution is more than sufficient for the experiments conducted in this paper, and greatly simplifies the implementation of the GNN, other approaches can be used when dealing with more complex graphs, like the GraphRNN decoder proposed in [22].

## III. EXPERIMENTS

The experimental section aims at showing that AR models for graphs are effective. In particular, we show that the proposed neural graph autoregressive (NGAR) model can be effectively trained, and that it provides graph predictions that significantly improve over simpler baselines.

The experiments are performed on sequences of attributed graphs with  $N = 5$  identified nodes and a variable topology. Each node is associated with a vector attribute of dimension  $F = 2$ , and no edge attributes; however, the extension to graphs with edge attributes, as well as graphs of variable order and non-identified nodes, is straightforward (e.g., see [20] for an example). The sequences are produced by two synthetic GGPs that we generate with a controlled memory order  $p$  (details follow in Section III-B), allowing us to have a ground truth for the analysis.

The comparative analysis of the tested methods is performed by considering a graph edit distance (GED) [11]

$$d(g_{t+1}, \hat{g}_{t+1}) \quad (11)$$

between the ground truth and the prediction made by the models<sup>2</sup>. We also analyse the performance of NGAR in terms of prediction loss and accuracy, in order to show the relative

<sup>2</sup>Although the NGAR method was trained with a specific loss function (see Sec.III-C), here we considered the GED measure instead, in order to provide a fair comparison of the methods.

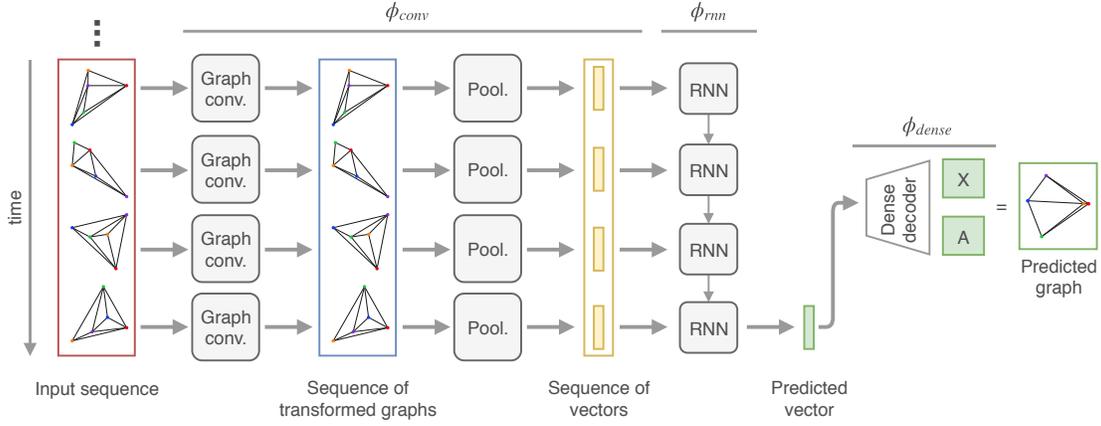


Figure 1. Schematic view of the proposed GNN. The network takes a sequence of  $k$  graphs up to time  $t$  as input (red box), and is trained to predict the graph at time  $t + 1$ . The input sequence is processed by a graph convolutional network, which is applied in parallel to all graphs, followed by a global pooling layer [15] that compresses the graphs down to vectors. The resulting sequence of vectors (yellow box) is processed by a recurrent neural network, which outputs a vector representing the predicted graph. Finally, the predicted vector is transformed in a graph by a multi-layer perceptron with two parallel outputs: one for producing the node features  $X_{t+1}$ , and one for the adjacency matrix  $A_{t+1}$ .

performance of the model on problems of different complexity. Finally, we report a qualitative assessment of the predictions of NGAR, by visualising the graphs predicted by the GNN and the true observations from the GGP.

The rest of this section introduces the baselines (Section III-A), the synthetic GGPs (Section III-B) and the implementation details for the NGAR architecture (Section III-C). Finally, we discuss the results of the experimental analysis in Section III-D.

#### A. Baseline methods

We consider four baselines commonly applied in the numerical case, that can be easily adapted to our setting. We denote our proposed method as *NGAR*, while the four baselines as *Mean*, *Mart*, *Move*, and *VAR*, respectively.

*a) Mean:* the first baseline assumes that the GGP is stationary, with independent and identically distributed graphs. In this case, the optimal prediction  $\hat{g}_{t+1}$  is the mean graph:

$$\hat{g}_{t+1} = \mathbb{E}^f[g],$$

where  $g \sim Q$ , and  $Q$  indicates here the distribution (supposed stationary) of the graphs.

*b) Mart:* the second baseline assumes that the GGP is a martingale, s.t.  $\mathbb{E}^f[g_{t+1}] = g_t$ , and predicts  $g_{t+1}$  as:

$$\hat{g}_{t+1} = g_t,$$

i.e., the graph at the previous time step.

*c) Move:* the third baseline considers the  $k$  preceding graphs  $\mathbf{g}_t^k$ , and predicts  $g_{t+1}$  to be the Fréchet sample mean graph of  $\mathbf{g}_t^k$

$$\hat{g}_{t+1} = \arg \min_{g'} \sum_{g_i \in \mathbf{g}_t^k} d(g', g_i)^2.$$

*d) VAR:* the fourth baseline is a vector AR model (VAR) of order  $k$ , which treats each graph  $g_t$  as the vectorisation of its node features  $X_t$  and adjacency matrix  $A_t$ , concatenated in a single vector:

$$u_t = [\text{vec}(X_t)^\top, \text{vec}(A_t)^\top]^\top \in \mathbb{R}^{N \cdot F + N^2}.$$

First, we compute a prediction  $\hat{u}_{t+1}$  defined by the linear model

$$\hat{u}_{t+1} = B_0 + \sum_{i=1}^k B_i \cdot u_{t-i+1},$$

from the regressor  $\mathbf{u}_t^k = [u_t, \dots, u_{t-k+1}^\top]^\top$  and, subsequently, the actual graph prediction  $\hat{g}_{t+1}$  is re-assembled from  $\hat{u}_{t+1}$ .

We mention that baseline *VAR* can be adopted in these experiments only because we are considering graphs with numerical attributes and fixed order; in more general settings this would not be possible.

#### B. Graph-sequence generation

We consider two different GGPs, both based on a common framework where a multivariate AR model of the form

$$x_{t+1} = f(x_t^p) + \epsilon, \quad (12)$$

produces a sequence of vectors  $\{x_1, x_2, \dots, x_t, \dots\}$ , which are then used to generate the graph sequence. From each  $x_t$ , a node-feature matrix  $X_t \in \mathbb{R}^{N \times F}$  is built, and the adjacency matrix  $A_t$  is created as the Delaunay triangulation of the rows of  $X_t$ , interpreted as points in  $\mathbb{R}^F$ . Here,  $\epsilon$  is a noise term whose components are randomly drawn from a stationary normal distribution  $\mathcal{N}(0, \sigma^2)$ .

The result of this process is a sequence of graphs  $\{g_1, g_2, \dots, g_t, \dots\}$ , where each graph depends on the previous  $p$  graphs. By choosing different implementations of  $f(x_t^p)$ , we are able to generate different GGPs. Note that the noise  $\eta$  and function  $H(\cdot, \cdot)$  in (8) are never made explicit in this procedure, but instead they are determined by the Gaussian

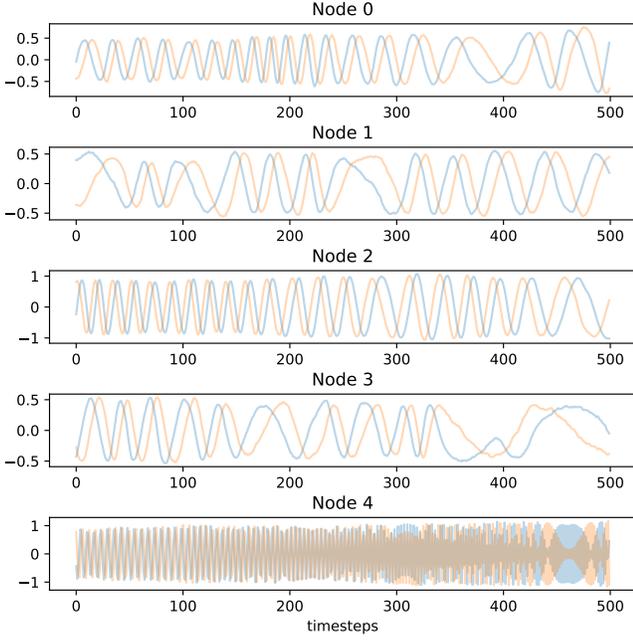


Figure 2. Temporal evolution of the node attributes of a graph in the rotational model of order 10. Each plot shows the dynamics of the  $F = 2$  attributes of each node in the graph.

noise perturbation introduced by  $\epsilon$ , which affects the node attributes and, consequently, causes possible changes in the topology. Section III-B1 and Section III-B2 describe the two procedures.

1) *Rotational model*: the function  $f(\mathbf{x}_t^p)$  in (12) is taken of the form

$$f(\mathbf{x}_t^p) = R(\mathbf{x}_t^p) \cdot x_t$$

where  $x_t \in \mathbb{R}^{N \cdot F}$  and  $R(\mathbf{x}_t^p)$  is a rotation matrix depending on the regressor. Matrix  $R$  is block-diagonal with blocks  $R_n$ ,  $n = 1, \dots, N$ , of size  $F \times F$  defined as:

$$R_n(\mathbf{x}_t^p) = \begin{bmatrix} \cos(\omega) & \sin(\omega) \\ -\sin(\omega) & \cos(\omega) \end{bmatrix}$$

$$\omega = c_n + \alpha \cos \left( \sum_{i=0}^{p-1} x_{t-i}(2n-1) + x_{t-i}(2n) \right)$$

The parameters  $\{c_n\}_1^N$  are randomly generated by a uniform distribution in  $(-1, 1]$ , while  $\alpha$  is set to 0.01.

The node attribute matrices are finally obtained by arranging each  $x_t$  in a  $N \times F$  matrix, and the regression function  $f(\cdot)$  can be interpreted as an independent and variable rotation of each node feature (see Figure 2).

2) *Partially Masked Linear Dynamical System*: we consider a discrete-time linear dynamical system

$$x_{t+1} = R x_t, \quad (13)$$

where  $x_t \in \mathbb{R}^c$ , and  $R \in \mathbb{R}^{c \times c}$  is an orthonormal random correlation matrix computed with the method proposed by Davies and Higham in [23].

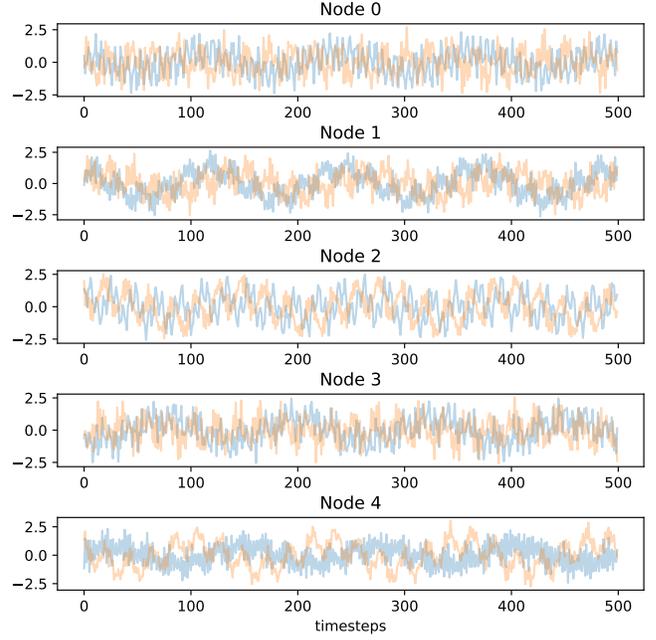


Figure 3. Temporal evolution of the node attributes of a graph in PMLDS(10). Each plot shows the dynamics of the  $F = 2$  attributes of each node in the graph.

Table I  
HYPERPARAMETER CONFIGURATION FOR THE GNN USED IN THE EXPERIMENTS.

Hyperparameter	Value
Weight for L2	0.0005
Learning rate	0.001
Batch size	256
Early stopping	20 epochs

Although the dynamical system (13) depends on exactly one previous time step, the partial observation of the first  $N F < c$  components of  $x_t$  results in a dynamical system of order  $p \propto (c - N \cdot F)$  [24]. Similarly to the rotational model, then, node attributes  $X_t$  are obtained by reshaping the masked vectors in  $\mathbb{R}^{N F}$  to  $N \times F$  matrices (see Figure 3). We refer to this setting as a *partially masked linear dynamical system* of complexity  $c$  (PMLDS( $c$ )).

The size  $c$  of the original linear system represents an index of complexity of the problem, as the system's memory is dependent on it: given  $N$ , a higher  $c$  will result in more complicated dynamics of  $x_t$ , and *vice versa* (see Figure 4). However, because the evolution of the GGP is controlled by hidden variables that the NGAR model never sees at training time, the problem is closer to real-world scenarios where processes are rarely fully observable.

### C. Graph neural network architecture

We use the same GNN architecture for both settings. Input graphs are represented by two matrices:

- a binary adjacency matrix,  $A \in \{0, 1\}^{N \times N}$ ;

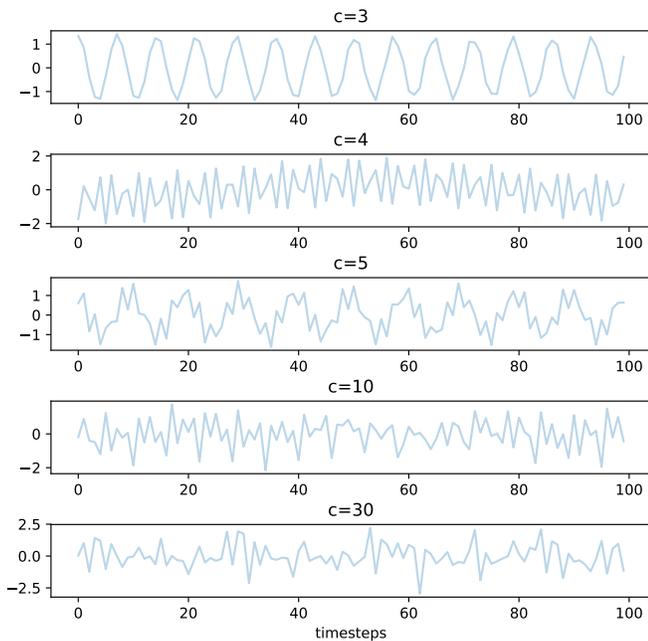


Figure 4. Temporal evolution of the first component of PMLDS, for different values of  $c$ . As  $c$  increases, it becomes more and more difficult to explain the dynamics of the observed coordinate.

- a matrix representing the attributes of each node,  $X \in \mathbb{R}^{N \times F}$ .

All hyperparameters of  $\phi_{nn}$  were found through a grid search on the values commonly found in the literature, using the validation loss as a metric, and are summarised in Table I. All sub-networks were structured with two layers to provide sufficient nonlinearity in the computation, and in particular  $\phi_{conv}$  and  $\phi_{dense}$  have been shown in the literature to be effective architectures for processing small graphs [19], [20]. Our network consists of two graph convolutional layers with 128 channels, ReLU activations, and L2 regularization. Here, we use convolutions based on a first-order polynomial filter, as proposed in [25], but any other method is suitable (e.g., we could use the edge-conditioned convolutions proposed in [26] in order to consider edge attributes). The graph convolutions are followed by a gated global pooling layer [15], with soft attention and 128 channels. The  $\phi_{conv}$  block is applied in parallel (i.e., with shared weights) to all  $k$  graphs in the input sequence, and the resulting vector sequence is fed to a 2-layer LSTM block with 256 units and hyperbolic tangent activations. Note that we keep a fixed  $k = 20$  for NGAR, Move, and VAR. The output of the LSTM block is fed to a fully connected network of two layers with 256 and 512 units, with ReLU activations, and L2 regularisation. Finally, the network has two parallel output layers with  $N \cdot N$  and  $N \cdot F$  units respectively, to produce the adjacency matrix and node attributes of the predicted graph. The output layer for  $A$  has sigmoid activations, and the one for  $X$  is a linear layer. The network is trained until convergence using Adam

Table II  
TEST PERFORMANCE OF NGAR ON THE ROTATIONAL MODEL OF ORDER  $p$ . WE REPORT THE FULL PREDICTION LOSS (INCLUDING THE REGULARISATION TERMS), AS WELL AS THE MSE AND LOG-LOSS ON THE NODE FEATURES  $X$  AND ADJACENCY MATRICES  $A$ , RESPECTIVELY. FOR  $A$ , WE ALSO REPORT THE ACCURACY ACHIEVED BY THE MODEL.

$p$	Loss	MSE ( $X$ )	log-loss ( $A$ )	Accuracy ( $A$ )
1	1.108	0.715	0.334	0.86
5	0.341	0.076	0.227	0.92
10	0.326	0.090	0.197	0.92
20	0.336	0.105	0.194	0.92
50	0.479	0.193	0.244	0.90
100	0.541	0.250	0.246	0.90

Table III  
TEST PERFORMANCE OF THE NGAR MODEL ON PMLDS( $c$ ). WE REPORT THE FULL PREDICTION LOSS (INCLUDING THE REGULARISATION TERMS), AS WELL AS THE MSE AND LOG-LOSS ON THE NODE FEATURES  $X$  AND ADJACENCY MATRICES  $A$ , RESPECTIVELY. FOR  $A$ , WE ALSO REPORT THE ACCURACY ACHIEVED BY THE MODEL.

$c$	Loss	MSE ( $X$ )	log-loss ( $A$ )	Accuracy ( $A$ )
11	0.200	0.018	0.154	0.95
15	0.278	0.041	0.195	0.92
20	0.283	0.038	0.204	0.92
30	0.341	0.081	0.222	0.90
60	1.366	0.983	0.345	0.85
110	4.432	3.950	0.418	0.82

[27], monitoring the validation loss on a held-out 10% of the training data for early stopping. We jointly minimise the mean squared error for the predicted node attributes, and the log-loss for the adjacency matrix. For each problem, we evolve the GGPs for  $10^5$  steps and test the model on 10% of the data.

#### D. Results

Tables II and III report the test performance of NGAR, in terms of prediction loss and accuracy. We can see that as  $p$  and  $c$  increase, the test performance gets worse, which is in agreement with the fact that  $p$  and  $c$  can be seen as indexes of complexity of the problem. In Table II, for the rotational GGP with  $p = 1$  we observe an unexpected high test loss, which might be associated with overfitting. We also see from Figure 5 that the test performance seems to be consistently better when the complexity of the problem is lower than the

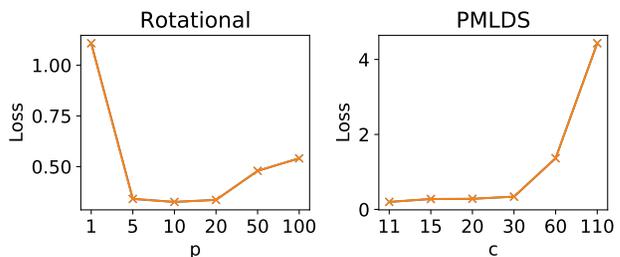


Figure 5. Test loss of NGAR for different levels of complexity on Rotational and PMLDS.

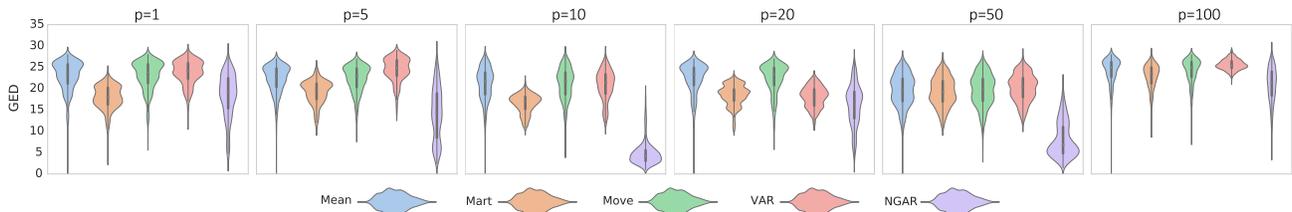


Figure 6. Comparison of *Mean* (blue), *Mart* (orange), *Move* (green), *VAR* (red), and *NGAR* (purple) on different rotational GGPs. Each plot shows the distribution of the residual GED (11) between the targets in the test set and the graphs predicted by each model respectively (best viewed in colour).

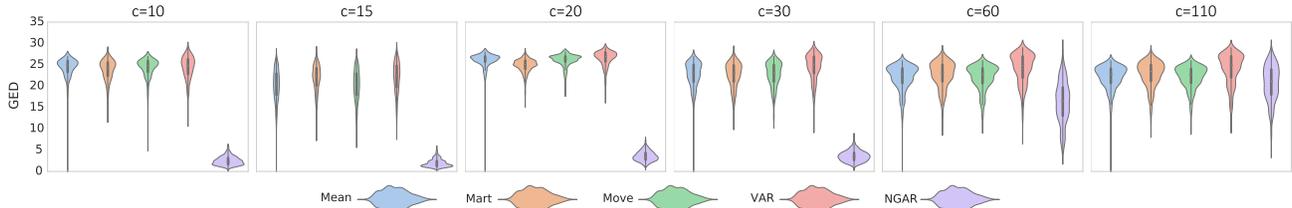


Figure 7. Comparison of *Mean* (blue), *Mart* (orange), *Move* (green), *VAR* (red), and *NGAR* (purple) on different PMLDS( $c$ ) GGPs. Each plot shows the distribution of the residual GED (11) between the targets in the test set and the graphs predicted by each model respectively (best viewed in colour).

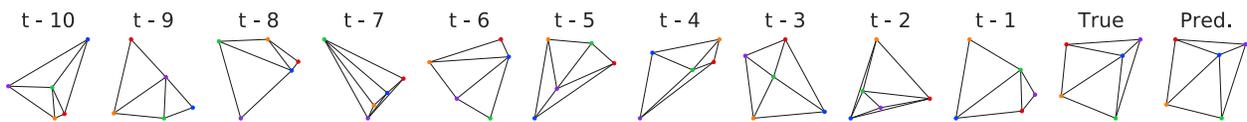


Figure 8. Comparison of a graph predicted by the *NGAR* model with the ground truth, on PMLDS(30) (best viewed in colour). Given a sequence of graphs observed up to time  $t - 1$ , it is difficult to intuitively predict the following graph in the sequence (marked “True” in the figure). However, the model is able to predict the graph almost exactly (marked “Pred.” in the figure).

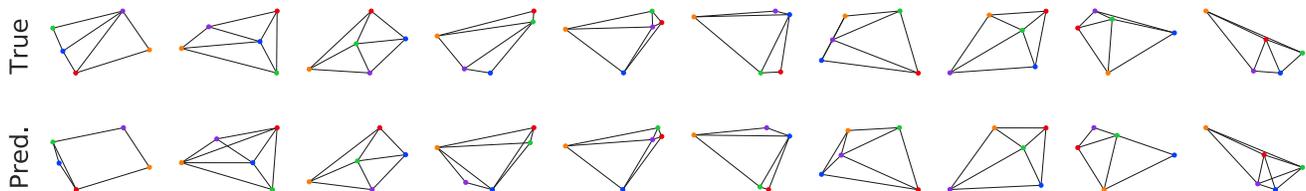


Figure 9. Comparison between randomly selected graphs in the test set of the PMLDS(30) GGP and the corresponding graphs predicted by the *NGAR* model (best viewed in colour). Upper row: samples from the ground truth. Bottom row: graphs predicted by the GNN.

order of *NGAR*, i.e.,  $p \leq k$  and  $c - N \cdot F \leq k$  (c.f. Sec. III-B1 and Sec. III-B2).

The second part of the experimental analysis aims at comparing the *NGAR* method with the baselines in terms of prediction error, assessed as the GED between predicted and ground-truth graphs; Figures 6 and 7 show a comparison of the baselines on different levels of complexity ( $p$  and  $c$ ). Among the baselines, we cannot clearly identify one with significantly better performance. Given the performance of *Mean* and *Mart* for all  $p$  and  $c$ , we see that neither GGP can be modelled as stationary or a martingale. *Mean* and *Move* performed almost the same in all experiments, while and *Mart* in some cases performed significantly better than the other baselines, e.g., see the rotational GGP with  $p = 10$ . Despite *VAR* being a method adapted from a widely used model in multivariate AR problems, here it was not able to result in relevant graph predictions. On the other hand, *NGAR*

consistently outperformed the baselines on almost all setting.

Finally, a qualitative assessment of the predictions is presented in Figures 8 and 9, which highlights a good performance of *NGAR* even on sequences of graphs that cannot be intuitively predicted.

#### IV. CONCLUSION AND FUTURE WORK

In this paper, we formalised an autoregressive framework for a stochastic process in which the observations are graphs. The framework considers a generic family of attributed graphs, in which nodes and edges can be associated with numerical and non-numerical attributes, the topology is allowed to change over time, and non-identified nodes can be considered, as well. We show that the proposed model for graphs is a non trivial generalisation of the classic multivariate autoregressive models, by introducing the Fréchet statistics. We proposed also to address the task of predicting the next graph in a

graph sequence with a GNN, hence leveraging deep learning methods for processing structured data. The GNN implementation proposed here is based on graph convolutional layers and recurrent connections, however when requested by the application, different architectures can be adopted.

Finally, we performed an experimental campaign in which we demonstrate the applicability of the proposed graph AR model, as well as that the proposed GNN can be trained to address prediction tasks. The proposed method is compared with four baselines on synthetic GGPs, to rely on a ground-truth for the analysis. The obtained results are promising, showing that the GNN was able to learn a non-trivial AR function, especially when compared to simpler, statistically motivated baselines. Possible application scenarios to be explored in future works include the prediction of the dynamics in social networks, load forecasting in power distribution grids, and modelling the behaviour of brain networks.

As a possible extension of this work, we intend to formulate the GNN architecture to work entirely on the space of graphs, without mapping the representation to a vector space. This would require a method for aggregating the graphs of a sequence (e.g., via weighted sum). However, such an operation is currently missing in the literature.

## REFERENCES

- [1] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, “Relational inductive biases, deep learning, and graph networks,” *arXiv preprint arXiv:1806.01261*, 2018.
- [2] D. Raposo, A. Santoro, D. Barrett, R. Pascanu, T. Lillicrap, and P. Battaglia, “Discovering objects and their relations from entangled scene representations,” *arXiv preprint arXiv:1702.05068*, 2017.
- [3] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende *et al.*, “Interaction networks for learning about objects, relations and physics,” in *Advances in neural information processing systems*, 2016, pp. 4502–4510.
- [4] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, “Neural relational inference for interacting systems,” *arXiv preprint arXiv:1802.04687*, 2018.
- [5] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, “High-order graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting,” *arXiv preprint arXiv:1802.07007*, 2018.
- [6] L. Livi and A. Rizzi, “The graph matching problem,” *Pattern Analysis and Applications*, vol. 16, no. 3, pp. 253–283, 2013.
- [7] E. Richard, S. Gaïffas, and N. Vayatis, “Link prediction in graphs with autoregressive features,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 565–593, 2014.
- [8] L. W. Beineke, R. J. Wilson, and P. J. Cameron, *Topics in algebraic graph theory*. Cambridge University Press, 2004, vol. 102.
- [9] D. Zambon, C. Alippi, and L. Livi, “Concept drift and anomaly detection in graph streams,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2018.
- [10] M. Fréchet, “Les éléments aléatoires de nature quelconque dans un espace distancié,” in *Annales de l’Institut Henri Poincaré*, vol. 10, no. 4, 1948, pp. 215–310.
- [11] Z. Abu-Aisheh, R. Raveaux, J.-Y. Ramel, and P. Martineau, “An exact graph edit distance algorithm for solving pattern recognition problems,” in *4th International Conference on Pattern Recognition Applications and Methods 2015*, 2015.
- [12] S. Fankhauser, K. Riesen, and H. Bunke, “Speeding up graph edit distance computation through fast bipartite matching,” in *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer, 2011, pp. 102–111.
- [13] S. Bogueux, L. Brun, V. Carletti, P. Foggia, B. Gaüzère, and M. Vento, “Graph edit distance as a quadratic assignment problem,” *Pattern Recognition Letters*, vol. 87, pp. 38–46, 2017.

- [14] D. Sejdinovic, B. Sriperumbudur, A. Gretton, and K. Fukumizu, “Equivalence of distance-based and rkhs-based statistics in hypothesis testing,” *The Annals of Statistics*, pp. 2263–2291, 2013.
- [15] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated graph sequence neural networks,” *arXiv preprint arXiv:1511.05493*, 2015.
- [16] M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in *Neural Networks, 2005. IJCNN’05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 2. IEEE, 2005, pp. 729–734.
- [17] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.
- [18] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” *arXiv preprint arXiv:1312.6203*, 2013.
- [19] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.
- [20] M. Simonovsky and N. Komodakis, “Graphvae: Towards generation of small graphs using variational autoencoders,” *arXiv preprint arXiv:1802.03480*, 2018.
- [21] N. De Cao and T. Kipf, “MolGAN: An implicit generative model for small molecular graphs,” *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.
- [22] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, “Graphrnn: Generating realistic graphs with deep auto-regressive models,” in *International Conference on Machine Learning*, 2018, pp. 5694–5703.
- [23] P. I. Davies and N. J. Higham, “Numerically stable generation of correlation matrices and their factors,” *BIT Numerical Mathematics*, vol. 40, no. 4, pp. 640–651, 2000.
- [24] E. Ott, *Chaos in Dynamical Systems*. Cambridge University Press, 2002.
- [25] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations (ICLR)*, 2016.
- [26] M. Simonovsky and N. Komodakis, “Dynamic edge-conditioned filters in convolutional neural networks on graphs,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

## APPENDIX

### A. Equivalence between (1) and (8).

Let  $(\mathcal{G}, d)$  be the Euclidean space  $(\mathbb{R}, |\cdot|)$ , and  $H(a, b) = a + b$ , then

$$g_{t+1} = H(\phi(\mathbf{g}_t^k), \eta) = \phi(\mathbf{g}_t^k) + \eta.$$

Regarding assumption (5), we see that the integral in (6) (known as Fréchet function of  $Q$ ) becomes

$$\int_{\mathcal{G}} d(g, g')^2 dQ(g) = \int_{\mathbb{R}} |g - g'|^2 dQ(g)$$

and

$$\begin{aligned} \int_{\mathcal{G}} d(H(\phi(\mathbf{g}_t^p), \eta), g')^2 dQ(\eta) &= \int_{\mathbb{R}} |\phi(\mathbf{g}_t^p) + \eta - g'|^2 dQ(\eta) \\ &= \int_{\mathbb{R}} \left\{ |\phi(\mathbf{g}_t^p) - g'|^2 + |\eta|^2 - 2\eta(\phi(\mathbf{g}_t^p) - g') \right\} dQ(\eta) \\ &= |\phi(\mathbf{g}_t^p) - g'|^2 + \text{Var}[\eta] - 2\mathbb{E}[\eta](\phi(\mathbf{g}_t^p) - g'). \end{aligned}$$

We conclude that (1) and (8) are equivalent, in fact

$$\phi(\mathbf{g}_t^k) \in \mathbb{E}^f[H(\phi(\mathbf{g}_t^k), \eta)] \Leftrightarrow \mathbb{E}[\eta] = 0.$$

and

$$\text{Var}[\eta] < \infty \Leftrightarrow \text{Var}^f[\eta] < \infty.$$