Generative Representation Learning in Recurrent Neural Networks for Causal Timeseries Forecasting

Georgios Chatziparaskevas, Ioannis Mademlis, *Senior Member, IEEE*, and Ioannis Pitas, *Fellow, IEEE*

***Abstract*—Feed-forward Deep Neural Networks (DNNs) are the state-of-the-art in timeseries forecasting. A particularly significant scenario is the causal one: when an arbitrary subset of variables of a given multivariate timeseries is specified as forecasting target, with the remaining ones (exogenous variables) *causing* the target at each time instance. Then, the goal is to predict a temporal window of future target values, given a window of historical exogenous values. To this end, this paper proposes a novel deep recurrent neural architecture, called Generative-Regressing Recurrent Neural Network (GRRNN), which surpasses competing ones in causal forecasting evaluation metrics, by smartly combining generative learning and regression. During training, the generative module learns to synthesize historical target timeseries from historical exogenous inputs via conditional adversarial learning, thus internally encoding the input timeseries into semantically meaningful features. During a forward pass, these features are passed over as input to the regression module, which outputs the actual future target forecasts in a sequence-to-sequence fashion. Thus, the task of timeseries generation is synergistically combined with the task of timeseries forecasting, under an end-to-end multitask training setting. Methodologically, GRRNN contributes a novel augmentation of pure supervised learning, tailored to causal timeseries forecasting, which essentially forces the generative module to transform the historical exogenous timeseries to a more appropriate representation, before feeding it as input to the actual forecasting regressor. Extensive experimental evaluation on relevant public datasets obtained from disparate fields, ranging from air pollution data to sentiment analysis of social media posts, confirms that GRRNN achieves top performance in multi-step long-term forecasting.**

***Impact Statement*—Timeseries forecasting is essential in several real-world application domains, ranging from environmental monitoring to network load prediction and financial assets management. Causal forecasting is the specific scenario where (part of) the input comes from a different timeseries than the output, with a causal relationship present between the input and the output series. Causal forecasting with a multi-step output horizon is a difficult problem and current solutions based on Recurrent Neural Networks (RNNs) lag in performance, which potentially renders them too unreliable. The algorithm proposed in this paper (GRRNN) offers lower error rates across various application domains, thus increasing the real-world applicability of RNNs for automated causal forecasting. GRRNN demonstrates that integrated generative representation learning can usefully amplify the performance of relevant RNNs, by enriching the features learned via regular supervised training.**

This work was supported by European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101093003.

Georgios Chatziparaskevas was with the Aristotle University of Thessaloniki, Greece (e-mail: gehatzip@yahoo.gr).

Ioannis Mademlis was with the Aristotle University of Thessaloniki, Greece (e-mail: imademlis@csd.auth.gr).

Ioannis Pitas is with the Aristotle University of Thessaloniki, Greece (e-mail: pitas@csd.auth.gr).

The source code is publicly available at: https://github.com/gehatzip/GRRNN.

***Index Terms*—Generative Adversarial Network, Multitask learning, Recurrent Neural Network, Representation Learning, Timeseries forecasting.**

I. Introduction

THE explosion of storage space and computational power, combined with ubiquitous connectivity, has allowed ever-increasing volumes of data to be harvested and analyzed at higher frequencies with each passing year. The substantial data demands of Deep Neural Networks (DNNs), applicable in a wide variety of areas such as computer vision [1] or Natural Language Processing [2], has exacerbated this trend. Timeseries data have also come into focus in this context, as any quantifiable phenomenon exhibiting variability over time can be studied as a timeseries; from sensor readings, like climatic factors, to digital videos or behavioural data, like electronic shop sales. Modelling and forecasting timeseries is essential for effective management, automation, prevention, decision making and leadership. Most useful timeseries consist of a large number of variables with interdependencies that are often not easily explainable. As a result, traditional approaches to modeling and forecasting have become inadequate.

The majority of real-world quantitative phenomena varying over time depend on multiple parameters, that may be interdependent. To effectively identify such dependencies, modeling through multivariate timeseries is the best option: they consist of an M-dimensional vector at each time instance, with the M different components of this time-varying vector called *channels*. Essentially, a timeseries is a temporally ordered set of such M-channel vectors and each element of this set, i.e., a specific timestamped M-dimensional vector, is called an *observation*. Practical advances in sensor and computer technology have made it possible to capture, store and analyze timeseries with a huge number of variables/parameters/channels.

In many problems, the ability to predict future values of a timeseries given past ones is a major advantage in countless application domains. Examples include financial forecasting [3], electric load forecasting [4], etc. This type of forecasting can be either *single-step* or *multi-step*: single-step forecasting methods predict the timeseries only at the next timestep, while multi-step predict the timeseries multiple timesteps ahead. The ability to forecast deeper in time is obviously more valuable in every application, rendering multi-step forecasting desirable, nonetheless it is more difficult. As a result, special methods are typically developed for multi-step forecasting, which are more involved and complex that single-step algorithms [5]. The process can be either *direct*, generating forecasts for a number

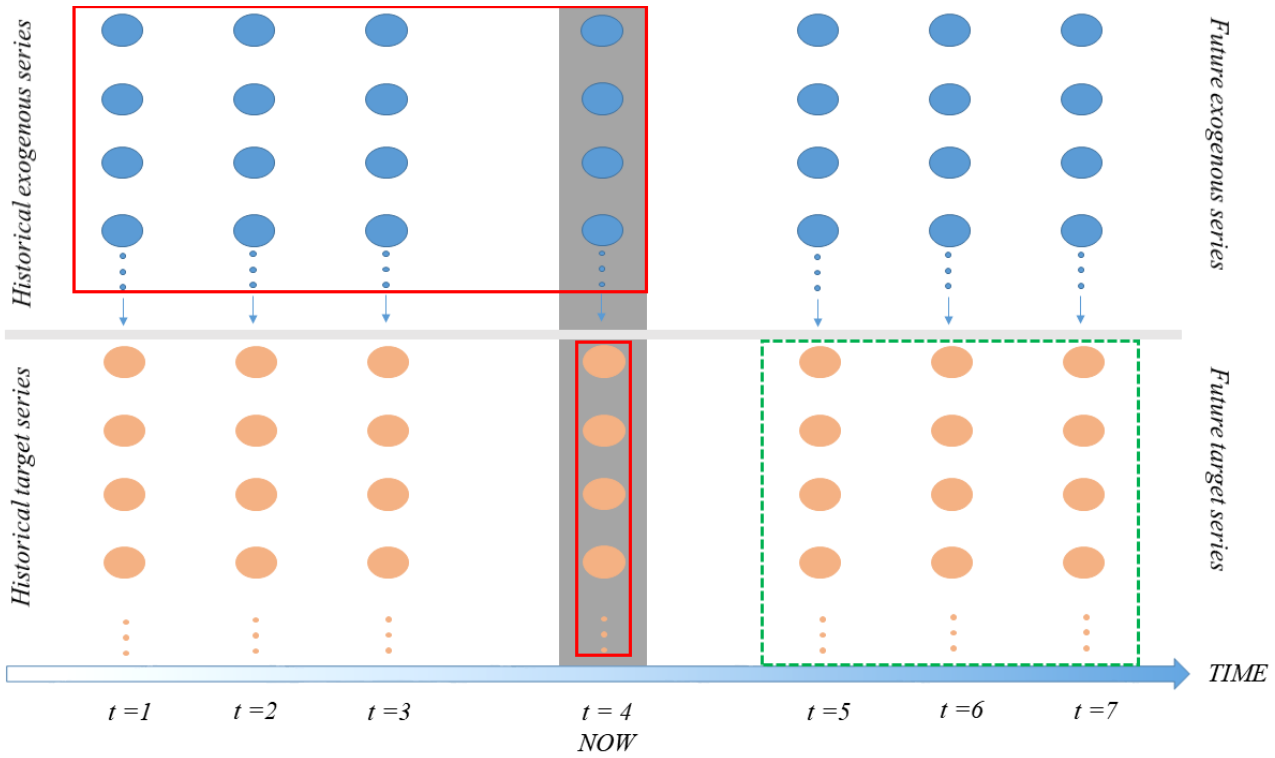


Fig. 1: The causal timeseries forecasting setting of GRRNN. At each timestep, each ellipse depicts a different channel. The solid-lined red rectangle encloses the inference-stage inputs of GRRNN, while the dotted-lined green rectangle encloses its inference-stage outputs. For simplicity, this example is limited to 7 timesteps overall.

of timesteps ahead at once, or *recursive*, with multiple single-step-ahead forecasts [6]. The latter method has the drawback of propagating the forecasting error of each single-step forecast to the next one.

A particularly significant scenario is the *causal* one: when an arbitrary subset of variables/channels of a given multivariate timeseries is specified as forecasting *target*, with the remaining ones (*exogenous* variables) "causing" the target at each time instance. Then, the goal is to predict a temporal window of *future target* values, given a window of *historical exogenous* values and, optionally, *historical target* values.

From an algorithmic perspective, the family of autoregressive (AR) and moving average (MA) methods form the basis of traditional forecasting. AR algorithms model the timeseries as autoregressive processes with future observations depending on previous ones [7]. MA methods on the other hand model the timeseries as moving average processes with each next observation depending on previous error terms. ARMA and its generalization ARIMA result from these basic regression methods [8], with Vector Autoregression (VAR) being a generalization of AR for multivariate timeseries forecasting. ARMA and ARIMA have also been generalized to multiple variables, while autoregression approaches have also evolved to non-linear ones [9]. Finally, the NARX (Non-linear AutoRegressive eXogenous) family of models has been instrumental in the popularization of causal forecasting [10].

Such traditional approaches have recently been surpassed in evaluation metrics by DNNs. As universal function approximators, DNNs can achieve better forecasting performance under certain conditions [11]. Multilayer Perceptrons (MLPs) [12] and Convolutional Neural Networks (CNNs) [13], Recurrent Neural Networks (RNNs) [14], autonecoders [15], sequence-to-sequence architectures [16], [17] and neural attention mechanisms [18] [19] have all been successfully adapted to timeseries forecasting under regression settings [20]. More recently, in an alternative approach, Generative Adversarial Networks (GANs) [21] have been employed for addressing timeseries forecasting as a timeseries generation task [22].

Most neural architectures initially handled forecasting under a specific learning paradigm. Nevertheless, in recent years, attempts have been made to deploy combinations of the above approaches, or hybridize deep learning with traditional timeseries forecasting methods. One such example is LSTNet [23], which embeds autoregression into a deep learning regression pipeline. However, existing multivariate forecasting DNNs, such as these ones, typically struggle to identify and utilize interdependencies between the different input channels, in order to come up with more accurate future predictions. Various approaches have been tested to overcome this limitation, but without any spectacular success up to now. Examples include clustering of similar timeseries [24], spatial attention mechanisms [25], [26], [27], as well as Convolutional [28] and Graph-Convolutional Neural Networks [29] capturing both spatial and temporal dynamics simultaneously. The limited ability of similar existing methods to identify and exploit

interdependencies between input channels have motivated us to formulate, implement and evaluate a novel alternative, relying on *combining attention with generative representation learning*. Partially relevant combinations, although attempted for other domains and tasks [30], [31], [32], [33], have not been designed for causal timeseries forecasting up to now.

Thus, this paper presents Generative-Regressing Recurrent Neural Network (GRRNN): a novel deep neural architecture for *direct multi-step causal forecasting*. It fills an important gap in existing literature, by exploring synergistic joint exploitation of regression and generative learning. To the best of our knowledge, no previous DNN has investigated this avenue for reducing causal timeseries forecasting error rates.

Essentially, GRRNN is a pipeline of two consecutive neural modules, i.e., a generative one (a Wasserstein Conditional Recurrent Generative Adversarial Network) and a regression one (a sequence-to-sequence Long Short-Term Memory-based Encoder-Decoder network with spatial and temporal attention). GRRNN appropriately exploits our knowledge of historical target data during training, so as to better forecast future target data during inference. Thus, the main novel contributions of the proposed method are two-fold:

- Joint, end-to-end regression and generative training is proposed for the first time in timeseries forecasting, leading to improved generalization ability concerning the prediction of future target data, due to a cross-regularization effect between the two modules. This effect likely stems from two sources: i) the inherent regularizing influence of multitask learning [34], [35], and ii) the inherent resistance of GANs to overfitting [21].
- The regression module does not receive raw historical exogenous input timeseries observations, but semantically rich internal features computed by the auxiliary generative module[1]. Thus, the latter one essentially learns effective latent representations, as it is trained to synthesize historical target data given historical exogenous ones, and transforms the input exogenous data accordingly, before feeding them to the regressor. These transformed representations natively capture interdependencies between the exogenous and the target timeseries channels. To the best of our knowledge, nothing similar has been proposed before for causal timeseries forecasting.

The causal timeseries forecasting setting in which GRRNN operates is depicted in Figure 1. Extensive experimental evaluation on 5 relevant public datasets obtained from disparate fields, ranging from air pollution data to sentiment analysis of social media posts, confirms that GRRNN indeed achieves top performance in multi-step long-term multivariate timeseries forecasting. The source code is publicly available at https://github.com/gehatzip/GRRNN.

The remainder of this article is structured in the following manner. Section II details previous recent literature in deep neural timeseries forecasting, emphasizing the causal scenario. Section III formulates the problem at hand, overviews required algorithmic preliminaries and presents the proposed method. Section IV overviews the experimental evaluation process and discusses the obtained results. Finally, Section V draws relevant conclusions from the preceding presentation.

[1]Along with the raw last historical target observation.

II. Related Work

Deep neural architectures for causal timeseries forecasting analyze historical exogenous values of an input timeseries in order to predict its future target values, for a horizon of fixed timesteps. Causal forecasting follows the NARX paradigm: assuming that the multivariate target/exogenous timeseries is denoted by $\mathbf{y}_t/\mathbf{x}_t$, respectively, the goal is to predict *future target* timeseries values, for a horizon of N_p timesteps, given an input window of *historical target* values and *historical exogenous* values, across N_h timesteps:

$$[\mathbf{y}_{t+1}, \ldots, \mathbf{y}_{t+Np}] = F(\mathbf{y}_t, \mathbf{y}_{t-1}, \ldots, \mathbf{y}_{t-N_h+1}, \mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-N_h+1}). \quad (1)$$

A notable DNN improving upon the NARX concept is DARNN [26]: a sequence-to-sequence Long Short-Term Memory (LSTM) RNN network architecture for single-step ahead forecasting. DARNN receives as its input a window of historical exogenous values and forecasts the respective future target values for the next timestep, following immediately after the input time window.

DARNN consists of two LSTMs, i.e., an Encoder and Decoder, and is equipped with two neural attention mechanisms. The first one assigns attention weights to the individual channels (spatial attention) and the second applies weights to different observations of each channel (temporal attention). A similar approach, combining spatial and temporal attention, is also followed in DSTP-RNN [27], where spatial attention consists of two parts: the first one weighs only the exogenous channels, while the second one weighs the weighed exogenous channels along with the historical channels.

The idea of capturing both the spatial (inter-timeseries) and temporal relations, which is central to DARNN, lies also behind neural architectures such as LCRM [36], DA-SKIP [37] or CNN-LSTM-NARX [38]. LCRM uses convolutional layers to extract the spatial and short-term temporal relations, along with an LSTM recurrent layer to capture longer-term temporal relations. DA-SKIP uses a DARNN to capture dependencies and a GRU-SKIP component that captures periodicity, followed by an autoregressive component that captures linearity. The final forecasts are the sum of DARNN and GRU-SKIP/autogressive combo. CNN-LSTM-NARX uses a 1D convolutional layer to capture correlations between the exogenous and target series. Subsequently, the computed feature maps are forwarded to a LSTM that captures the temporal correlations and predicts the final future target values for one time-step ahead.

Moving beyond the causal setup, recent neural architectures which have utilized GANs for timeseries forecasting include CWGAN-TS [39] and [40]. These exploit a GAN and a GAN-MLP combination, respectively, to generate future observations of the target timeseries as a conditional data synthesis task. However, GANs have not been previously utilized in the causal forecasting setup.

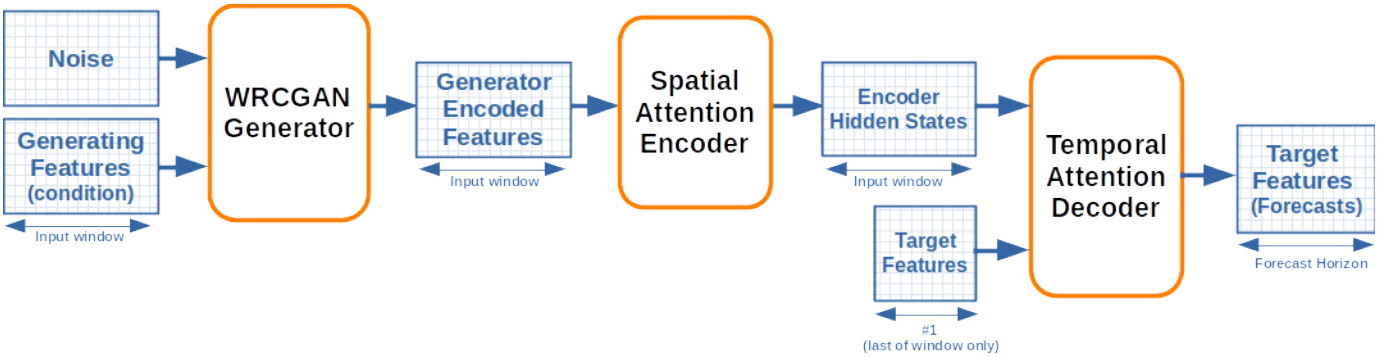


Fig. 2: GRRNN block diagram during the inference stage (no Critic present).

GRRNN, i.e., the deep neural architecture proposed in this paper, augments the input channels selection and weighting of the spatiotemporal attention mechanism with generative representation learning. Unlike [39] and [40], which use a GAN to conduct probabilistic forecasting, GRRNN internally exploits the Generator of a GAN during inference not to generate the final/future target forecasts, but to synthesize the historical target timeseries that corresponds to the input historical exogenous timeseries. To achieve this, it generates semantically rich latent features which encode efficiently the historical exogenous inputs. These are being fed as input to the main sequence-to-sequence regression module, where the final future target timeseries is forecast. Therefore, the proposed method combines a regression module (trained in a supervised manner) with a generative synthesis module (trained in an adversarial manner), into an end-to-end pipeline that takes advantage of both.

III. Timeseries forecasting with Generative-Regressing Recurrent Neural Network

This Section first formulates the causal timeseries forecasting task. Subsequently, it briefly presents necessary preliminaries that are being exploited by the proposed method. Finally, it details the proposed deep neural architecture (GRRNN) and highlights its novel contributions.

A. Problem formulation

In this paper, the goal of multivariate causal timeseries forecasting is to predict a future *target series* multiple timesteps ahead, given as input a window of historical exogenous observations that form a *generating series*. Both the generating and the target series are sets of temporally ordered multichannel observations (vectors), with the latter one depending on the former one.

Formally, the generating series consists of N_h M_G-dimensional vectors (multichannel observations), while the target series may consist of N_p M_T-dimensional observations. In the general case, it may hold that $M_G \neq M_T$ and $N_h \neq N_p$. Obviously, N_p is the number of timesteps ahead the forecasting is desired to "see". Thus, the input to a machine learning model for timeseries forecasting is a temporal window of historical exogenous values in the form of the generating timeseries: $\mathbf{X} \in \mathbb{R}^{M_G \times N_h}$, with each column of matrix $\mathbf{X}$ being an observation. Similarly, the output of the model is the temporally ordered set of future values of the target series: $\mathbf{X} \in \mathbb{R}^{M_T \times N_p}$. The goal of the forecasting model is to predict $\mathbf{Y}$ given $\mathbf{X}$. Depending on the employed algorithm, the input and/or the output can be fed/generated either all at once, or sequentially (one observation/column at a time, in consecutive steps).

Moreover, during training, a set of N_h M_T-dimensional vectors of historical target values may be given. These are temporally coupled to the N_h corresponding exogenous observations in a one-to-one fashion. They can be considered a separate matrix $\mathbf{H} \in \mathbb{R}^{M_T \times N_h}$.

B. Preliminaries

1. Wasserstein Recurrent Conditional Generative Adversarial Network, or WRCGAN [41]. A simple Generative Adversarial Network (GAN) [21] consists of two neural networks trying to outwit one another during training: the Generator and the Discriminator. The first one receives as input a random vector and generates as output a vector that should be drawn from the generating distribution of a given dataset. The Discriminator is a classifier attempting to discriminate between input vectors that have been fabricated by the Generator ("fake") and ones that have been drawn from the actual training dataset ("real"). In Conditional GANs [42], the Generator receives an additional input vector, besides the random one, upon which the generation process is conditioned. After training is completed, the Discriminator can be discarded.

In vanilla GANs, the Discriminator tries to maximize the log probability of real input and the inverted probability of fake input, while the Generator tries to minimize the latter. This introduces lack of stability to learning and decelerates convergence. The so-called Wasserstein GAN [41] addressed this issue by essentially replacing the Discriminator, which classifies the input as real or fake, with a so-called Critic, which assigns a higher score to real inputs than fake ones. The training objective is to minimize the Wasserstein distance between the distributions P_r, P_g of the real samples $\mathbf{x}$ and generated ones $\hat{\mathbf{x}}$:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(\mathbf{x}, \hat{\mathbf{x}}) \sim \gamma}[\|\mathbf{x} - \hat{\mathbf{x}}\|], \qquad (2)$$

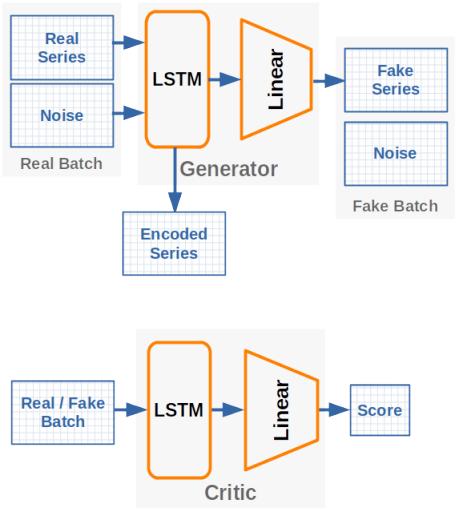


Fig. 3: WRCGAN block diagram.

where $\Pi(P_r, P_g)$ is the set of all joint distributions $\gamma(\mathbf{x}, \hat{\mathbf{x}})$, $x \sim$ marginal P_r, $\hat{x} \sim$ marginal P_g and inf denotes the greatest lower bound.

The Wasserstein distance is approximated by:

$$W(P_r, P_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{\mathbf{x} \sim P_r}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim P_g}[f(\mathbf{x})], \quad (3)$$

where sup denotes the least upper bound and f is a function bounded by the 1-Lipschitz constraint:

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|. \quad (4)$$

The WGAN Discriminator, referred to as "Critic", is trained to approximate f with an objective to maximize the Wasserstein distance: the discrepancy between the distribution of the real and generated samples. Its loss function is the negative of that distance, adjusted by a Gradient Penalty [43] that enforces the Lipschitz constraint on function f:

$$\begin{aligned} \mathcal{L}_D = & \underbrace{-\mathbb{E}_{\mathbf{x} \sim P_r}[D(\mathbf{x})] + \mathbb{E}_{\hat{\mathbf{x}} \sim P_g}[D(\hat{\mathbf{x}})]}_{Original\ Critic\ Loss} \\ & + \underbrace{\lambda \mathbb{E}_{\hat{\mathbf{x}} \sim P_G}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]}_{Gradient\ Penalty}. \end{aligned} \quad (5)$$

The Generator tries to minimize the Wasserstein distance, so its loss function is the negative score assigned by the Critic to the generated samples:

$$\mathcal{L}_G = -\mathbb{E}_{\hat{\mathbf{x}} \sim P_g}[D(\hat{\mathbf{x}})], \quad (6)$$

where

$$D \sim f, \hat{x} = G(x). \quad (7)$$

WRCGAN is a Wasserstein Conditional GAN where both the Generator and the Discriminator/Critic are LSTM RNNs [44], as they are more suitable for sequential data (e.g., timeseries). The structure of WRCGAN is depicted in Figure 3.

2. Sequence-to-Sequence modelling. Tasks involving the mapping of input sequences to output sequences, where the lengths of the input and the output are not necessarily equal, are typically addressed by Encoder-Decoder neural architectures [45]. Using LSTMs for timeseries forecasting, the context of the Encoder LSTM's hidden states along with the last known observation of the timeseries is given as input to the Decoder. During training, this last known observation is known and used in a scheme called *teacher forcing* [46]. During inference, it may simply be the last observation forecast as the Decoder's output in the previous timestep.

3. Spatio-Temporal Attention. The proposed method adopts from [26] a dual-stage attention mechanism acting on LSTM-based Encoder-Decoder architectures. This consists of a spatial attention module and a temporal attention module. The first one applies a weight to each individual current input channel/variable at each timestep. These weights are outputted by a MultiLayer Perceptron (MLP), given all the values of this variable throughout the input window, along with the hidden state of the Encoder during the previous timestep. The weighted channels are subsequently passed on to the Encoder afterwards. Figure 4 depicts the overall process, which can be formally expressed as:

$$\begin{aligned} e_t^k &= \mathbf{W}_e \cdot [\mathbf{h}_{t-1}; \mathbf{c}_{t-1}; \mathbf{x}_k], \\ a_t^k &= \frac{exp(e_t^k)}{\sum_{i=1}^{M_{gen}} exp(e_t^i)}, \\ \tilde{\mathbf{x}}_t &= (a_t^1 x_t^1, a_t^2 x_t^2, \ldots, a_t^{M_{gen}} x_t^{M_{gen}})^\top, \\ & 1 \leq k \leq M_{gen} \end{aligned} \quad (8)$$

where $\mathbf{h}_{t-1}$ and $\mathbf{c}_{t-1}$ are the hidden and cell states of the encoder LSTM at the previous timestep $t-1$, $\mathbf{W}_e$ are the trainable parameters of the spatial attention MLP, a_t^k is the weight of the k-th feature at time t. Finally $\tilde{\mathbf{x}}_t$ is the weighted feature vector at time t and M_{gen} is the number of the rich generated features.

The temporal attention mechanism weighs the Encoder's hidden states at each timestep using the Decoder's hidden state. A context $\mathbf{c_t}$ is computed at each timestep t as the weighted sum of the Encoder's hidden states. This context is concatenated with the previous forecast produced or the last historical target timeseries observation at the initial step. The temporal attention weights are outputted by a deep neural network consisting of a linear layer, followed by a $tanh$ non-linearity and another linear layer. Figure 5 depicts the process, which can be formally expressed as:

$$\begin{aligned} l_t^i &= \mathbf{v}_d^\top \cdot tanh(\mathbf{W}_d \cdot [\mathbf{d}_{t-1}; \mathbf{s}_{t-1}; \mathbf{h}_i]), \\ \beta_t^i &= \frac{exp(l_t^i)}{\sum_{j=1}^{N_p} exp(l_t^j)}, \\ \mathbf{c}_t &= \sum_{i=1}^{T} \beta_t^i \mathbf{h}_i, \\ \tilde{\mathbf{y}}_{t-1} &= \tilde{\mathbf{w}}^\top [\mathbf{y}_{t-1}; \mathbf{c}_{t-1}], \\ & 1 \leq i \leq N_h, \end{aligned} \quad (9)$$

where $\mathbf{d}_{t-1}$ and $\mathbf{s}_{t-1}$ are the previous hidden and cell states of the decoder LSTM, $\mathbf{h}_i$ is the hidden state of the encoder at the i-th timestep of the input window, $\mathbf{v}_d^\top$, $\mathbf{W}_d$, $\tilde{\mathbf{w}}^\top$ are

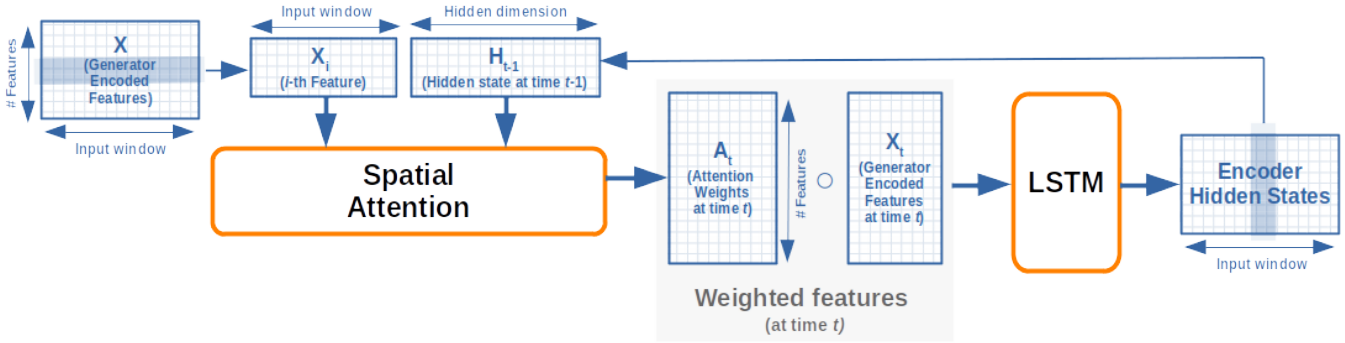


Fig. 4: GRRNN Encoder block diagram.

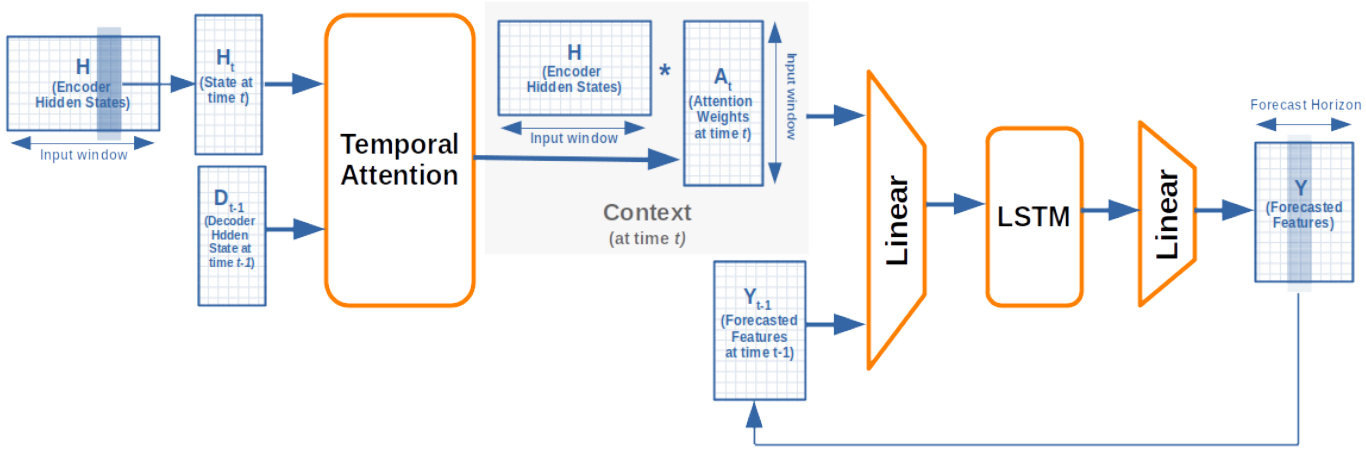


Fig. 5: GRRNN Decoder block diagram.

trainable parameters, $\mathbf{c}_t$ is the context at time t, and N_h is the input window size. The decoder LSTM is given as input $\tilde{\mathbf{y}}_{t-1}$ and produces the forecast $\mathbf{y}_t$ at each timestep t.

C. GRRNN: The proposed architecture for causal timeseries forecasting

GRRNN is a deep neural architecture for direct multi-step timeseries forecasting. It consists of two separate LSTM modules, namely the generative module G and the regression module R. From an architectural perspective these are placed consecutively, thus forming a pipeline that is being trained end-to-end using a multitask loss function. Figure 2 depicts the block diagram of GRRNN's inference stage.

G is tasked to sequentially synthesize a timeseries, conditioned on the input generating series. This is a secondary, auxiliary task meant to generate semantically rich internal representations of the input that are forwarded to R during inference. G is structured as a Wasserstein Conditional Recurrent GAN (WRCGAN) [41]. Therefore, it is composed of a Generator and a Critic (Discriminator). Both the Generator and the Critic are single LSTM layers unfolding for N_h timesteps. During the forward pass of both the training and the inference stage, at the i-th timestep, the *hidden state* vector $\mathbf{f}_i$ of the Generator layer is the i-th feature vector forwarded as input to R. On the other hand, during the forward pass of the training stage only, the respective *output* vector $\mathbf{o}_i$ of this layer is forwarded as the i-th input observation to the Critic. With this structure, G is adversarially trained to synthesize historical target observations $\mathbf{H}$ conditioned on historical exogenous observations $\mathbf{X}$. After training of the overall GRRNN model has been completed, the Critic can be discarded.

The regression module R consists of an Encoder-Decoder neural architecture, which implements sequence-to-sequence modelling. Both the Encoder and the Decoder consist of a single LSTM layer each. The Encoder unfolds for N_h timesteps, while the Decoder unfolds for N_p times steps. Additionally, R is equipped with a dual-stage attention mechanism adopted from [26]. The first-stage spatial attention mechanism weighs the individual current input channels/variables at each timestep, using the previous state of the Encoder. The second-stage temporal attention mechanism weighs the hidden states of the Encoder, each time using the previous hidden state of the Decoder. The sum of the weighted hidden states of the Encoder is passed as context to the Decoder.

Training of the overall pipeline-structured architecture proceeds end-to-end, by means of multitask learning [47]. In short, G is tasked to synthesize the historical target observations given the historical exogenous observations, across the input time window. The generated internal representations of the latter ones are fed as input to the Encoder of R.

Additionally, the raw last historical target observation is fed as an extra input to the Decoder of R, which is tasked to forecast future target observations across the output time window. Thus, during training, the overall loss function is a sum of:

- a regression term $\mathcal{L}_R$, acting on the output of the Decoder of R and employing ground-truth future target observations for loss computations, and
- an adversarial term $\mathcal{L}_G$, acting on the output of the Generator of G thanks to an independent Critic network which is being trained concurrently.

Optimizing the model in these two tasks simultaneously, using any variant of Stochastic Gradient Descent (SGD) and error back-propagation, has the advantage of cross-regularizing the two modules to achieve better generalization. Obviously, during back-propagation, the gradient signal caused by $\mathcal{L}_R$ is propagated backwards until reaching R, where the respective neuronal errors are summed with errors caused by $\mathcal{L}_G$.

Formally, the GRRNN loss function is $\mathcal{L}_{GRRNN} = \frac{1}{2}\mathcal{L}_G + \frac{1}{2}\mathcal{L}_R$, with both terms equally weighted. $\mathcal{L}_G$ is given by Eq. (6) and $\mathcal{L}_R$ is given by:

$$\mathcal{L}_R = \sum_{i=1}^{N_p} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2, \tag{10}$$

where $\mathbf{y}_i$ is the ground-truth value of the future target series at the i-th timestep of the forecast horizon, while $\hat{\mathbf{y}}_i$ is the predicted one at the i-th timestep.

As it can be seen, $\mathcal{L}_R$ is the sum of Mean Square Errors (MSE) of the forecasts throughout the forecast horizon N_p. $\mathcal{L}_G$ is the negative mean score computed by the Critic D for the fake samples $G(\mathbf{z})$ produced by the Generator. Concurrently, D is being trained with the loss function $\mathcal{L}_D$ from Eq. (5).

The proposed architecture is visualized for the inference stage (no Critic present) in Figure 2. The advantage of GRRNN is that the regressor receives as input learnt, semantically rich timeseries representations, which inherently capture the interdependencies between the historical target and the historical exogenous timeseries. Yet, during inference, only historical exogenous observations are required to be known and provided as input (not the historical target channels, except for the last timestep of the input window). Additionally, the multitask nature of $\mathcal{L}_{GRRNN}$ and the involvement of a GAN jointly result in heightened resistance to overfitting [34], [35], [21]. As shown in Section IV, these improvements jointly lead to significant, measurable performance gains in the most difficult scenario of long-term, multi-step forecasting.

As a final methodological note, G can be slightly modified to perform *timeseries-to-timeseries translation* instead of conditional timeseries generation, similarly to how GANs are employed for image-to-image translation in computer vision [48]. In practical terms, the only difference is that an additional $\mathcal{L}_T$ supervised loss term is added to the overall pool of loss functions during training, which penalizes the deviation of the Generator's output from the respective historical target timeseries:

$$\mathcal{L}_T = \sum_{i=1}^{N_h} \|\mathbf{o}_i - \mathbf{h}_i\|_1, \tag{11}$$

TABLE I: Target channels of each dataset used.

Dataset	Target
SML2010	Living room Temperature, Bedroom Temperature
Air Quality	CO, NMHC, C_6H_6, NO_x, NO_2
ETD	Oil Temperature
Energy Consumption	Appliances, Lights
Twitter Public Opinion	Republicans Poll Result, Democrats Poll Result

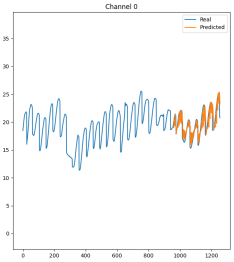
where $\mathbf{h}_i$, i.e., the i-th column of matrix $\mathbf{H}$, is the ground-truth value of the historical target series at the i-th timestep, while $\mathbf{o}_i$ is the one generated by G at the i-th timestep. In this case, the overall loss function is $\mathcal{L}_{GRRNN-T} = \frac{1}{2}\mathcal{L}_G + \frac{1}{2}\mathcal{L}_R + \lambda_T\mathcal{L}_T$, where λ_T is a non-negative real hyperparameter. As training converges, this setup leads G to synthesize a historical target series that is not only realistic enough to fool the Critic, given the condition, but also as close to the specific ground-truth series as possible. However, the GAN dynamics become more complex with the inclusion of $\mathcal{L}_T$ and there is a greater danger of unstable training for G. This second variant of the proposed method is separately evaluated in Section IV, under the name of GRRNN-T.

IV. Experimental Evaluation

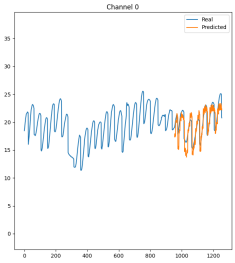
The proposed DNN architecture was evaluated and compared against competing methods on 5 public datasets. The experimental protocol and the corresponding results are presented below.

A. Datasets

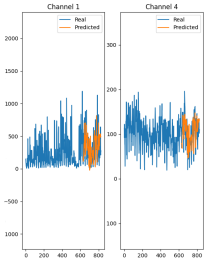
Five different timeseries datasets have been employed for evaluation purposes. The first one is "SML2010" [49], which consists of house sensor outputs (exogenous series) and the living room and bedroom temperatures (target series). Second, the "Air Quality" dataset [50] has been used, which includes measurements of various air pollutants obtained from sensors located in an Italian city over a period of several months. These measurements are the exogenous series, while the target is the concentration of non-methane hydrocarbons (NMHC), benzene (C_6H_6), carbon monoxide (CO), and nitric oxides (NO_x, NO_2). The "Energy Consumption" dataset [51] includes temperature and humidity measurements in various parts inside and outside a household for a period of multiple months (exogenous timeseries). The target series consists of energy consumption measurements for its appliances and lights. Four, the "ETD" dataset [52] contains data related to electricity grid transformers having as exogenous series certain load types and as target the oil temperature of the transformer. Finally, the "Twitter Public Opinion" dataset [53] contains as exogenous series the daily, nationwide average (for the United States) of values quantifying sentiment, polarity, political bias and profanity in Twitter posts about the two main political parties. Its target series is the daily average of US nationwide opinion polls for the two parties. All measurements were obtained during the six-month period before the 2016 US presidential elections. The exact target channels of each dataset can be seen in Table I; all other available channels have been utilized as exogenous ones.



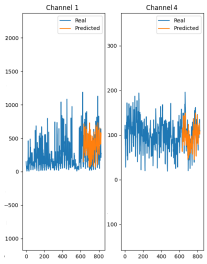
(a) Model: GRRNN-T, Dataset: SML2010.



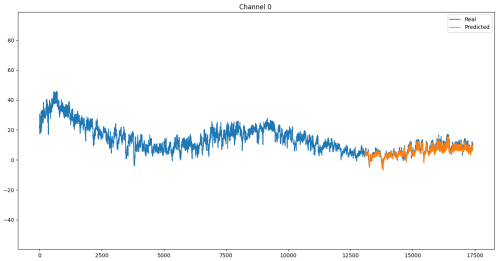
(b) Model: Stem-GNN, Dataset: SML2010.



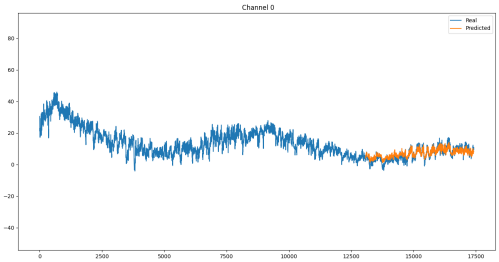
(c) Model: GRRNN-T, Dataset: Air Quality.



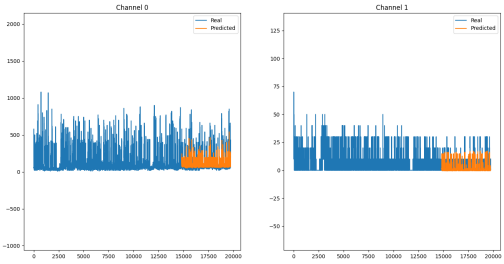
(d) Model: DARNN, Dataset: Air Quality.



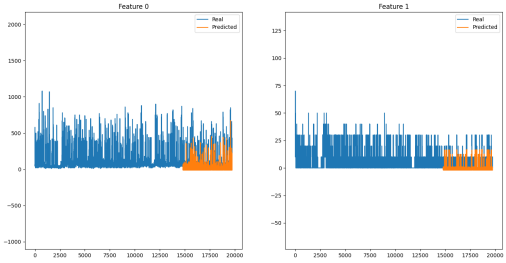
(e) Model: GRRNN-T, Dataset: ETD.



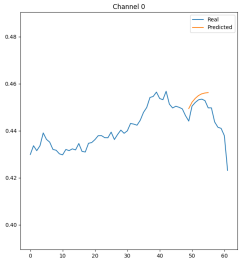
(f) Model: Stem-GNN, Dataset: ETD.



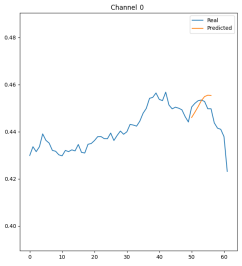
(g) Model: GRRNN-T, Dataset: Energy Consumption.



(h) Model: DARNN, Dataset: Energy Consumption.



(i) Model: GRRNN-T, Dataset: Twitter Public Opinion.



(j) Model: DSTP-RNN, Dataset: Twitter Public Opinion.

Fig. 6: Visual examples of forecasting 7 timesteps ahead with GRRNN-T vs with the best performing competitor (per dataset). Specific channels have been chosen for visualization purposes.

TABLE II: Optimal hyperparameters for the examined datasets. Forecasting horizon: 1 timestep (short/single-step).

	SML2010					Air Quality					ETD				
	Batch size	Learning Rate	Epochs	Window Size	Hidden Dim.	Batch size	Learning Rate	Epochs	Window Size	Hidden Dim.	Batch size	Learning Rate	Epochs	Window Size	Hidden Dim.
NARX	28	0.011	65	7	29	25	0.015	40	7	62	31	0.012	10	9	47
LSTM	22	0.043	66	7	29	16	0.05	70	13	24	16	0.039	15	8	45
Enc-Dec	17	0.05	41	8	30	31	0.011	17	8	60	18	0.05	69	9	45
LSTNET	20	0.046	65	10	63	18	0.023	32	9	27	31	0.017	62	8	64
DARNN	18	0.048	12	11	57	29	0.027	11	10	35	27	0.029	28	13	28
DSTP-RNN	28	0.03	68	14	54	30	0.034	67	13	44	21	0.032	67	7	26
Stem-GNN	17	0.05	56	10	-	31	0.01	15	7	-	20	0.03	62	11	-
GRRNN (proposed)	25	0.043	33	13	62	28	0.016	64	13	36	21	0.018	43	8	48

	Energy Consumption					Twitter Public Opinion				
	Batch size	Learning Rate	Epochs	Window Size	Hidden Dim.	Batch size	Learning Rate	Epochs	Window Size	Hidden Dim.
NARX	19	0.010	25	13	55	17	0.046	15	8	29
LSTM	19	0.016	70	12	37	24	0.025	19	7	28
Enc-Dec	32	0.01	29	14	32	24	0.045	11	7	45
LSTNET	28	0.01	67	14	61	23	0.036	65	7	61
DARNN	26	0.014	11	7	24	18	0.02	52	9	64
DSTP-RNN	24	0.012	42	7	56	20	0.029	55	5	50
Stem-GNN	28	0.018	16	9	-	29	0.011	14	7	-
GRRNN (proposed)	31	0.023	23	10	42	22	0.03	59	5	24

TABLE III: Optimal hyperparameters for the examined datasets. Forecasting horizon: 7 timesteps (long/multi-step).

	SML2010					Air Quality					ETD				
	Batch size	Learning Rate	Epochs	Window Size	Hidden Dim.	Batch size	Learning Rate	Epochs	Window Size	Hidden Dim.	Batch size	Learning Rate	Epochs	Window Size	Hidden Dim.
Enc-Dec	16	0.049	69	14	55	27	0.01	15	7	64	16	0.049	12	14	29
DARNN	31	0.043	15	13	57	24	0.034	22	10	41	25	0.028	10	13	46
DSTP-RNN	23	0.03	56	13	32	24	0.015	44	9	26	24	0.032	11	12	37
Stem-GNN	31	0.03	29	13	-	31	0.012	34	8	-	23	0.027	28	11	-
GRRNN (proposed)	32	0.022	57	13	31	24	0.02	55	10	61	24	0.02	50	10	48

	Energy Consumption					Twitter Public Opinion				
	Batch size	Learning Rate	Epochs	Window Size	Hidden Dim.	Batch size	Learning Rate	Epochs	Window Size	Hidden Dim.
Enc-Dec	28	0.01	25	14	46	20	0.021	12	3	36
DARNN	16	0.036	19	8	24	22	0.053	40	4	45
DSTP-RNN	27	0.016	48	8	57	21	0.04	45	4	58
Stem-GNN	20	0.01	10	11	-	21	0.039	61	3	-
GRRNN (proposed)	20	0.024	34	14	40	24	0.05	56	3	49

TABLE IV: Optimal values of hyperparameter λ_T for the examined datasets, in the case of GRRNN-T.

Dataset	Single-step-ahead	Multi-step-ahead
SML2010	0.677	0.121
Air Quality	0.133	0.628
ETD	0.101	0.334
Energy Consumption	0.133	0.108
Twitter Public Opinion	0.530	0.994

TABLE V: Hyperparameter value optimization range.

Hyperparameter	Min	Max
Learning rate	0.0001	1.0
Mini-batch size	16	32
Number of epochs	10	70
LSTM hidden dimension	24	64
Input window size	7	14
λ_T	0.1	1.0

The above-mentioned datasets were selected because they are publicly available and widely used, cover diverse application fields and can be inherently partitioned into exogenous and target channels. The only exception is "Twitter Public Opinion", which is not widely used and has been selected due to its unique properties, as it is a comparatively small dataset in terms of total time steps. In each case, the relevant multivariate timeseries was temporally split into a training subset (80% of the overall timesteps) and a test subset (the remaining 20%). The training subset was then internally split into a training set and a validation set, again following the 80%/20% rule. All splits were performed according to the chronological order, as per common protocols in relevant literature (e.g., [29], [27], [26], [54], [23], etc.).

The training set was partitioned into individual data points by employing overlapping temporal sliding windows of observations across all channels, with a size of $N_h + N_p$ timesteps and a shifting step of 1 timestep. The validation set was similarly split into sliding windows of size $N_h + N_p$ and a shifting step equal to the forecast horizon N_p. During validation, the

TABLE VI: Forecasting error rates for the examined datasets. Forecasting horizon: 1 timestep (short/single-step). The best performance per metric is in bold, the second best is underlined. Lower is better in all metrics.

	SML2010			Air Quality			ETD		
	MAE	SMAPE	RMSE	MAE	SMAPE	RMSE	MAE	SMAPE	RMSE
NARX	0.408	0.514	0.457	0.132	0.652	0.191	0.014	0.088	0.018
LSTM	0.087	0.158	0.105	0.103	0.386	0.139	0.065	0.326	0.079
Enc-Dec	0.103	0.173	0.134	0.163	0.592	0.212	0.066	0.337	0.081
LSTNet	0.204	0.276	0.300	0.088	0.350	0.120	0.026	0.137	0.031
DARNN	**0.024**	**0.040**	**0.047**	0.082	0.308	0.118	**0.013**	0.068	**0.016**
DSTP-RNN	0.061	0.107	0.078	**0.077**	**0.294**	**0.110**	0.015	0.093	0.018
Stem-GNN	0.154	0.258	0.186	0.180	0.647	0.230	0.085	0.439	0.105
GRRNN (proposed)	0.045	0.079	0.058	0.079	0.347	**0.110**	**0.013**	**0.066**	0.017
GRRNN-T (proposed)	0.148	0.250	0.178	0.159	0.578	0.207	0.085	0.385	0.106

	Energy Consumption			Twitter Public Opinion		
	MAE	SMAPE	RMSE	MAE	SMAPE	RMSE
NARX	0.582	1.778	0.732	0.294	0.556	0.326
LSTM	0.054	1.229	0.082	0.405	0.799	0.464
Enc-Dec	0.054	1.241	0.082	0.363	0.678	0.427
LSTNet	0.032	1.175	0.069	0.163	**0.317**	0.214
DARNN	**0.028**	**1.100**	**0.056**	0.296	0.609	0.334
DSTP-RNN	0.033	1.151	0.057	**0.150**	0.328	**0.207**
Stem-GNN	0.063	1.176	0.087	0.316	0.630	0.371
GRRNN (proposed)	0.038	1.190	0.060	0.215	0.474	0.269
GRRNN-T (proposed)	0.042	1.227	0.070	0.292	0.507	0.364

TABLE VII: Forecasting error rates for the examined datasets. Forecasting horizon: 7 timesteps (long/multi-step). The best performance per metric is in bold, the second best is underlined. Lower is better in all metrics.

	SML2010			Air Quality			ETD		
	MAE	SMAPE	RMSE	MAE	SMAPE	RMSE	MAE	SMAPE	RMSE
Enc-Dec	0.144	0.246	0.175	0.161	0.587	0.209	0.072	0.359	0.089
DARNN	0.131	0.217	0.171	0.147	0.622	0.205	0.074	0.371	0.091
DSTP-RNN	0.081	0.144	0.111	0.158	0.746	0.224	0.069	0.342	0.084
Stem-GNN	0.068	0.115	**0.093**	0.171	0.603	0.211	0.069	0.339	0.083
GRRNN (proposed)	0.062	**0.101**	0.098	**0.133**	**0.471**	0.174	0.027	0.140	**0.036**
GRRNN-T (proposed)	**0.060**	0.102	0.098	**0.133**	0.491	**0.173**	**0.026**	**0.132**	0.035

	Energy Consumption			Twitter Public Opinion		
	MAE	SMAPE	RMSE	MAE	SMAPE	RMSE
Enc-Dec	0.053	1.235	0.081	0.305	0.555	0.377
DARNN	0.052	1.554	0.083	0.219	0.362	0.278
DSTP-RNN	0.079	1.312	0.100	0.191	0.365	0.298
Stem-GNN	0.069	1.199	0.090	0.287	0.508	0.367
GRRNN (proposed)	**0.035**	**1.131**	0.073	**0.148**	0.420	0.238
GRRNN-T (proposed)	0.041	1.210	**0.071**	0.184	**0.302**	**0.214**

TABLE VIII: Standard deviation of the proposed method's forecasting error rates in the 7-timestep horizon scenario, across all data points in each test set.

	SML2010			Air Quality			ETD		
	MAE	SMAPE	RMSE	MAE	SMAPE	RMSE	MAE	SMAPE	RMSE
GRRNN	0.061	0.256	0.064	0.070	0.264	0.074	0.009	0.076	0.009
GRRNN-T	0.053	0.066	0.058	0.106	0.194	0.117	0.938	0.310	1.333

	Energy Consumption			Twitter Public Opinion		
	MAE	SMAPE	RMSE	MAE	SMAPE	RMSE
GRRNN	0.033	0.233	0.041	0.195	0.573	0.152
GRRNN-T	0.031	0.183	0.034	0.155	0.247	0.080

TABLE IX: Average model ranking across all datasets and metrics, for both forecasting horizon scenarios. The best performance per scenario is in bold, the second best is underlined. Lower is better.

Model	1-step horizon	7-step horizon
NARX	6.53	-
LSTM	6.07	-
Enc-Dec	6.73	4.93
LSTNET	4.33	-
DARNN	**2.33**	4.4
DSTP-RNN	**2.33**	4.47
Stem-GNN	7.53	3.87
GRRNN (proposed)	<u>2.60</u>	<u>1.73</u>
GRRNN-T (proposed)	6.53	**1.6**

first part of each such data point (with temporal length N_h) was utilized as input to the forecasting algorithm/DNN, while the second part (with temporal length N_p) was exploited as ground-truth forecast. The validation set was exploited for selecting the best model for each competing method, during hyperparameter optimization. Finally, the test set was split to data points in a manner identical to how the validation set had been partitioned and was employed for final evaluation of each forecasting DNN/algorithm with its optimal hyperparameters.

At the i-th timestep ($1 \leq i \leq N_h$), the Encoder of R receives as its i-th input observation the hidden state vector $\mathbf{f}_i$ from the Generator of G. The Decoder sequentially forecasts N_p future target observations/vectors, where N_p is the forecasting horizon. Typically, it is selected according to the needs of the user in the context of the application domain. Since $N_p \neq N_h$, an Encoder-Decoder neural architecture is the most suitable [55].

B. Preprocessing

All datasets were preprocessed according to standard procedures before being utilized. Initially, any invalid channels (e.g., with constant value over time) were removed. Afterwards, min-max scaling per channel was applied to the remaining data: each value x was transformed to $\tilde{x}$ as follows:

$$\tilde{x} = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{12}$$

C. Hyperparameter optimization

The hyperparameter space consists of the batch size, the learning rate, the number of epochs, the hidden state dimension of the RNN units and the input window size. Additionally, $\mathcal{L}_T$ loss scaling factor λ_T is employed for GRRNN-T training. The validation set of each dataset has been exploited for hyperparameter optimization before evaluating the proposed architecture on the corresponding test set, separately for each forecast horizon. The optimization algorithm used is SMAC (Sequential Model-based Algorithm Configuration) [56], i.e., a stochastic algorithm that applies Bayesian optimization in order to find an optimal hyperparameter combination. A cost function is defined and the algorithm navigates the hyperparameter space with the objective to minimize it. In this paper, the average of all error metrics computed on the validation set was used, while the number of optimization cycles was set to 100.

Tables II and III contain the optimal hyperparameter values for the examined forecast horizons of 1 and 7 timesteps, respectively. Table IV contains the optimal values of the hyperparameter λ_T across all datasets and for both forecast horizons, in the case of GRRNN-T. Table V defines the value range utilized for hyperparameter optimization.

D. Evaluation

The proposed and the other competing methods were evaluated for one short-term ($N_p = 1$) and one long-term ($N_p = 7$) forecasting horizon, separately for each dataset. Besides the proposed GRRNN architecture, four competing, recent, sophisticated DNNs were evaluated on causal timeseries forecasting in the 5 employed datasets for comparison purposes. The first one is LSTNet [23] which combines a CNN and an RNN, along with autoregression, and normally conducts only single-step forecasting. Thus, LSTNet is examined only in the single-step-ahead forecast setting. The second is DSTP-RNN [27] which implements two spatial and one temporal attention mechanisms. The third one is Stem-GNN [54] which uses spectral representation and convolutions to extract inter-channel correlations and patterns. The fourth one is a multi-step ahead version of DARNN [26]. DSTP-RNN, Stem-GNN and DARNN are evaluated for both forecasting horizons.

Besides the above-described recent DNN competitors, three traditional baselines were additionally evaluated: a) NARX, implemented as a MultiLayer Perceptron, b) a single vanilla LSTM, and c) a double vanilla LSTM in an Encoder-Decoder (Enc-Dec) formulation. NARX and the single vanilla LSTM can only be utilized in the single-step forecasting scenario, while Enc-Dec is also applicable in the multi-step forecasting case. Finally, a second variant of the proposed architecture was evaluated, which also contains $\mathcal{L}_T$ from Eq. (11) as an additional loss term during training. It is denoted below as GRRNN-T.

The employed evaluation metrics were the following ones:

- Mean Absolute Error:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|x_i - \hat{x_i}|\,.$$

- Symmetric Mean Absolute Percentage Error:

$$SMAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|x_i - \hat{x_i}|}{(|x_i| + |\hat{x_i}|)/2}.$$

- Root Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x_i})^2}.$$

In the above, x_i is the real value of a timeseries at timestep i, $\hat{x_i}$ is the forecasted value of that timeseries at timestep i and N is the number of timesteps. In all three cases, a lower metric value implies better forecasting. The final forecast errors per dataset were produced after averaging over all target channels, since the employed evaluation metrics generate one value per target channel.

TABLE X: Dunn-Bonferroni post-hoc test results for the 7-step forecasting horizon.

	DARNN	DSTP-RNN	GRRNN	GRRNN-T	Enc-Dec	Stem-GNN
DARNN	1.0	1.0	0.000317	0.000120	1.0	1.0
DSTP-RNN	1.0	1.0	0.000196	0.000073	1.0	1.0
GRRNN	0.000317	0.000196	1.0	1.0	0.000005	0.010038
GRRNN-T	0.000120	0.000073	1.0	1.0	0.000002	0.004514
Enc-Dec	1.0	1.0	0.000005	0.000002	1.0	1.0
Stem-GNN	1.0	1.0	0.010038	0.004514	1.0	1.0

E. Results

The forecasting error rates for the test sets of all employed datasets are presented in Tables VI and VII (for all metrics, lower is better). Overall, the proposed GRRNN architecture is typically either the best one or the second best one across almost all evaluation metrics and for all datasets. It is intended as a multiple-step-ahead forecasting method and, indeed, it performs best in terms of error rates in the relevant, more challenging experimental setting that has a long forecast horizon of 7 timesteps. Table VIII depicts the standard deviation of the proposed method results per error metric, across all test data points, for this multi-step horizon scenario. For evaluation completeness reasons, the method has also been tested on the single-step-ahead forecasting horizon scenario.

In the difficult 7-step horizon scenario, GRRNN achieves the lowest error rates in all error metrics, except in "SML2010" RMSE where it comes second behind Stem-GNN. When distinguishing between GRRNN and GRRNN-T, although there is no clear winner between them, it seems that GRRNN-T performs far better on two out of three error metrics in the "Twitter Public Opinion" dataset. This difference can be attributed to the small size of this dataset, which consists of less than 100 timesteps. Indeed, without the $\mathcal{L}_T$ loss term of GRRNN-T during training, the generative module simply learns the conditional distribution of the historical target data given historical exogenous ones; thus the quality of the features it computes is highly sensitive to the sample (training set) size. The result is a poor representation learning capability in the low-data regime for GRRNN. This behaviour is fully in-line with the intuition behind the functionality and modus operandi of GRRNN, as well as with the well-known data inefficiency of GANs.

In single-step-ahead forecasting the proposed GRRNN architecture performs slightly worse, although it still remains the best or second best performer in 3 out of 5 datasets. It comes second for all metrics in the "SML2010" dataset, after DARNN. It also has the lowest RMSE and second lowest MAE of the "Air Quality" dataset. As for the "ETD" dataset, GRRNN excels along with DARNN: both of them have the lowest MAE, with GRRNN having lower SMAPE and DARNN lower RMSE. In the "Twitter Public Opinion" dataset, LSTNet and DSTP-RNN architectures are more accurate than the proposed method, again probably due to small dataset size.

Regarding the relative performance of the proposed GRRNN compared to the baseline DARNN architecture, the former scores higher in all instances of the multi-step-ahead forecasting setting. However it does not fare equally well in the single-step-ahead setting, meaning that the benefits of generative representation learning become apparent only in the harder scenario (long forecasting horizon). In the less challenging single-step-ahead configuration, the stochasticity of the generated features during inference (caused by the random vector input to the Generator) tends to drown out the advantages of the richer representations computed by the auxiliary generative module. This effect seems to get exacerbated for GRRNN-T, which fails utterly in the single-step-ahead forecast scenario compared to GRRNN. A potential reason is the more unstable GAN training in such a setup, where Eq. (10) offers less supervision in comparison to the multi-step-ahead configuration.

The relative performance of all competing models is summarized in Table IX, which depicts the average ranking of each model over all error rates and datasets, for each of the two examined forecast horizons. As it can be seen, GRRNN is the second best approach in both forecasting scenarios, while GRRNN-T is the best approach in the multi-step forecasting scenario. To validate the significance of the ranking, Friedman test was conducted for a typical significance level of 0.05. The null hypothesis proposes that no statistical significance exists in the rankings and that the forecasting accuracy is similar for all competing models. The resulting p-value is $1.67E-15$ and $2.22E-11$ for the single-step and the multi-step forecasting scenario, respectively; both vales are many orders of magnitude lower than 0.05. Thus, the null hypothesis is rejected and the models' performance difference is shown to be statistically significant. Post-hoc test results for the multi-step horizon, which is the one this article focuses on, are depicted in Table X that contains pair-wise p-values for the Bonferroni-adjusted Dunn test. As it can be seen, GRRNN-T significantly stands out from the competition. Finally, example visualizations of 7-step forecasts using GRRNN-T and the best performing competing method per dataset are depicted in Fig. 6.

Finally, it can be noted that the errors for one dataset may diverge across different metrics. For instance, this is the case in the "Air Quality" dataset for the single-step-ahead forecast configuration, where the proposed GRRNN came first per RMSE (along with DSTP-RNN), second per MAE and third per SMAPE. This behaviour can be attributed to the statistical profile of the dataset and the nature of the metrics themselves.

V. Conclusions

This article presents a novel deep neural architecture, called GRRNN, that conjoins generative learning and regression to attack the challenging problem of direct multi-step causal forecasting. Essentially, a generative module transforms the input historical exogenous timeseries to a more appropriate and richer representation, before feeding it as input to the

actual forecasting regressor. Thus, the task of timeseries generation is synergistically combined with the task of timeseries forecasting, with the overall architecture trained end-to-end in a multitask manner. Extensive experimental evaluation on publicly available datasets from diverse fields has shown that GRRNN is very effective and outperforms its competitors in the multi-step-ahead forecasting setting, which is the most challenging one, if large enough training datasets are available.

Future research may focus on overcoming the identified limitations of the proposed architecture, for the low-data regime and the single-step-ahead forecasting cases, by enhancing the generative module to achieve improved conditional density estimation during training and reduced stochasticity during inference. Alternatively, it might be fruitful to rethink how the generative module assists the forecasting process. In GRRNN, it is exploited to generate richer features with greater predictive capacity. It would be interesting for it to also assist the regressor's decoder in the generation of the final forecasts.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Commission's Horizon Europe Research and Innovation Actions under grant agreement 101093003 (HORIZON-CL4-2022-DATA-01-01, TEMA). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] C. Symeonidis, I. Mademlis, I. Pitas, and N. Nikolaidis, "Neural attention-driven Non-Maximum Suppression for person detection," *IEEE Transactions on Image Processing*, vol. 32, pp. 2454–2467, 2023.

[2] D. Tsirmpas, I. Gkionis, G. T. Papadopoulos, and I. Mademlis, "Neural Natural Language Processing for long texts: A survey on classification and summarization," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108231, 2024.

[3] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data," *Pattern Recognition Letters*, vol. 136, pp. 183–189, 2020.

[4] ——, "Deep adaptive input normalization for time series forecasting," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3760–3765, 2019.

[5] R. Chandra, S. Goyal, and R. Gupta, "Evaluation of deep learning models for multi-step ahead time series prediction," *IEEE Access*, vol. 9, pp. 83 105–83 123, 2021.

[6] S. B. Taieb, G. Bontempi, A. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition," 2011.

[7] R. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. Australia: OTexts, 2018.

[8] G. E. P. Box and G. M. Jenkins, "Some Recent Advances in Forecasting and Control," *Journal of the Royal Statistical Society Series C*, vol. 17, no. 2, pp. 91–109, 1968.

[9] ——, *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[10] I. J. Leontaritis and S. A. Billings, "Input-output parametric models for non-linear systems Part I: deterministic non-linear systems," *International Journal of Control*, vol. 41, no. 2, pp. 303–328, 1985.

[11] Z. Tang, C. de Almeida, and P. A. Fishwick, "Time series forecasting using neural networks vs. Box-Jenkins methodology," *Simulation*, vol. 57, no. 5, pp. 303–310, 1991.

[12] Z. Tang and P. Fishwick, "Feedforward neural nets as models for time series forecasting," *Informs Journal on Computing*, vol. 5, pp. 374–385, 11 1993.

[13] S. Barra, S. M. Carta, A. Corriga, A. S. Podda, and D. R. Recupero, "Deep learning and time series-to-image encoding for financial forecasting," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 683–692, 2020.

[14] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021.

[15] N. Nguyen and B. Quanz, "Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, vol. 1409, 2014.

[17] S. Du, T. Li, and S.-J. Horng, "Time series forecasting using sequence-to-sequence deep learning framework," in *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, 2018.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, R. Vishwanathan, and R. Garnett, Eds., 2017.

[19] T. Gangopadhyay, S. Y. Tan, Z. Jiang, R. Meng, and S. Sarkar, "Spatiotemporal attention for multivariate time series prediction and interpretation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[20] A. Koochali, A. Dengel, and S. Ahmed, "If you like it, GAN it—probabilistic multivariate times series forecast with GAN," *Engineering Proceedings*, vol. 5, no. 1, 2021.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2014.

[22] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," *arXiv preprint arXiv:1706.02633*, 2017.

[23] G. Lai, W.-C. Chang, T. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with Deep Neural Networks," in *Proceedings of the International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018.

[24] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach," *Expert Systems with Applications*, vol. 140, p. 112896, 2020.

[25] Y. Tao, L. Ma, W. Zhang, J. Liu, W. Liu, and Q. Du, "Hierarchical attention-based recurrent highway networks for time series prediction," *arXiv preprint arXiv:1806.00685*, 2018.

[26] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[27] Y. Liu, C. Gong, L. Yang, and Y. Chen, "DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction," *Expert Systems with Applications*, vol. 143, p. 113082, 2020.

[28] L. Bai, L. Yao, S. S. Kanhere, Z. Yang, J. Chu, and X. Wang, "Passenger demand forecasting with multi-task convolutional recurrent neural networks," in *Poceedings of the Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. Springer, 2019.

[29] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," 2020.

[30] C. Papaioannidis, I. Mademlis, and I. Pitas, "Autonomous UAV safety by visual human crowd detection using multi-task deep neural networks," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[31] ——, "Fast CNN-based single-person 2D human pose estimation for autonomous systems," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[32] ——, "Fast single-person 2D human pose estimation using multi-task Convolutional Neural Networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[33] ——, "Fast semantic image segmentation for autonomous systems," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2022.

[34] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[35] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2004.

[36] X. Yu and D. Li, "A long-term recurrent convolutional network for stock index prediction," *Journal of Physics: Conference Series*, vol. 1914, no. 1, p. 012049, 2021.

[37] B. Huang, H. Zheng, X. Guo, Y. Yang, and X. Liu, "A novel model based on DA-RNN network and skip gated recurrent neural network for periodic time series forecasting," *Sustainability*, vol. 14, no. 1, 2022.

[38] A. S. A. Moursi, N. El-Fishawy, S. Djahel, and M. A. Shouman, "Enhancing PM2.5 prediction using NARX-based combined CNN and LSTM hybrid model," *Sensors*, vol. 22, no. 12, 2022.

[39] S. Liu and M. Motani, "Towards better long-range time series forecasting using generative adversarial networks," *arXiv preprint arXiv:2110.08770*, 2021.

[40] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, "Stock market prediction based on generative adversarial network," *Procedia Computer Science*, vol. 147, pp. 400–406, 2019.

[41] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2017.

[42] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, vol. abs/1411.1784, 2014.

[43] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2017.

[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[45] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[46] A. Goyal, A. Lamb, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2016.

[47] Y. Zhang and Q. Yang, "An overview of multi-task learning," *National Science Review*, vol. 5, no. 1, pp. 30–43, 2017.

[48] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[49] F. Zamora-Martínez, P. Romeu, P. Botella-Rocamora, and J. Pardo, "On-line learning of indoor temperature forecasting models towards energy efficiency," *Energy and Buildings*, vol. 83, pp. 162–172, 2014.

[50] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario," *Sensors and Actuators B: Chemical*, vol. 129, no. 2, pp. 750–757, 2008.

[51] L. M. Candanedo, V. Feldheim, and D. Deramaix, "Data driven prediction models of energy use of appliances in a low-energy house," *Energy and Buildings*, vol. 140, pp. 81–97, 2017.

[52] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the International Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2021.

[53] D. Karamouzas, I. Mademlis, and I. Pitas, "Public opinion monitoring through collective semantic analysis of tweets," *Social Network Analysis and Mining*, vol. 12, no. 1, p. 91, 2022.

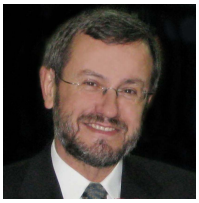
[54] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Spectral temporal graph neural network for multivariate time-series forecasting," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2020.

[55] S. Barra, S. M. Carta, A. Corriga, A. S. Podda, and D. R. Recupero, "Deep learning and time series-to-image encoding for financial forecasting," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 683–692, 2020.

[56] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proceedings of the International Conference on Learning and Intelligent Optimization (LION)*. Springer, 2011.

Georgios Chatziparaskevas is a software engineer. He received his B.Sc. in Computer and Telecommunication Engineering in 2007. Currently he is working as software engineer in the field of CAE.

Dr. Ioannis Mademlis (S'17-M'18-SM'22) is a computer scientist, specialized in artificial intelligence. He received a M.Sc. degree in intelligent/cognitive systems (2014) and a Ph.D. in machine learning and computer vision (2018) from the Aristotle University of Thessaloniki, Greece (AUTH). He has participated in 6 European Union-funded R&D projects, having co-authored approximately 70 publications in academic journals and international conferences. He is an IEEE Senior Member and a committee member of the International Artificial Intelligence Doctoral Academy (IAIDA). His current research interests include machine learning, computer vision, autonomous robotics and human-computer interaction.



Prof. Ioannis Pitas (SM'94-F'07, IEEE Fellow, IEEE Distinguished Lecturer, EURASIP Fellow) received the Diploma and PhD degree in Electrical Engineering, both from the Aristotle University of Thessaloniki (AUTH), Greece. Since 1994, he has been a Professor at the Department of Informatics of AUTH and Director of the Artificial Intelligence and Information Analysis (AIIA) lab. He served as a Visiting Professor at several Universities. His current interests are in the areas of computer vision, machine learning, autonomous systems, intelligent digital media, image/video processing, human-centred interfaces, affective computing, 3D imaging and biomedical imaging. He has published over 906 papers, contributed in 47 books in his areas of interest and edited or (co-)authored another 11 books. He has also been member of the program committee of many scientific conferences and workshops. In the past he served as Associate Editor or co-Editor of 9 international journals and General or Technical Chair of 4 international conferences. He participated in 70 R&D projects, primarily funded by the European Union and is/was principal investigator/researcher in 42 such projects. He has 31600+ citations to his work and h-index 87+ (Google Scholar). Prof. Pitas leads the International AI Doctoral Academy (IAIDA) of the European H2020 R&D project AI4Media https://ai4media.eu/. He coordinates the HE project "TEMA" (g.a.n. 101093003).