Future Generation Computer Systems, Volume 100, 2019, Pages 656-673. https://doi.org/10.1016/j.future.2019.05.060

Neural network architecture based on gradient boosting for IoT traffic prediction

Manuel Lopez-Martin, Belen Carro and Antonio Sanchez-Esguevillas

Abstract— Network traffic forecasting is an operational and management function that is critical for any data network. It is even more important for IoT networks given the number of connected elements and the real-time nature of many connections. This work presents a novel deep learning architecture applicable to this supervised regression problem. It is based on an additive network model formed by 'learning blocks' that are stacked iteratively following, in part, the principles of gradient boosting models. The resulting architecture is trained end-to-end using stochastic gradient descent. This new architecture has connections with residual, stacked and boosted networks, being different from any of them. Like residual networks, it shows excellent convergence behavior during training and allows for deeper models. It has a regularization effect similar to stacked models and presents excellent prediction results as gradient boosting models do. The building elements of the architecture are neural network blocks or learning blocks, that can be constituted by a sequence of simple fully connected layers or by more elaborate dispositions of recurrent and convolutional layers. The resulting architecture is a generic additive network (gaNet) applicable to any supervised regression problem.

To obtain experimental results on a hard prediction problem, the model is applied to the forecasting of network traffic using IoT traffic volume real data from a mobile operator. The paper presents a comprehensive comparison of results between the proposed new model and many alternative algorithms, showing important improvements in terms of prediction performance metrics and training/prediction processing times.

Index Terms— neural network; gradient boosting; additive model; IoT traffic prediction

* Correspondence: mlopezm@acm.org; Tel.: +34-983-423-980, Fax: +34-983-423-667
The authors declare that there is no conflict of interest regarding the publication of this paper

1. INTRODUCTION

Network traffic prediction is one of the main application areas of Machine Learning (ML) to data networking [1,2,3]. Traffic volume prediction can be defined as the forecasting of incoming and outgoing bytes-count at different connections levels in the network hierarchy (device, link, routing…). Traffic prediction is a critical element in network operations and management: congestion control, routing, resource allocation and management of service level agreements (SLA), among many other network responsibilities and functions [1,2].

Traffic prediction can be performed with statistical analysis methods e.g. Auto Regressive Moving Average (ARMA), Auto Regressive Integrated Moving Average (ARIMA)...[4,5,6] or Machine Learning (ML) methods e.g. Neural Networks, Random Forest... [6,7]. Additionally, the prediction process can distinguish whether the data used for prediction consist exclusively on past traffic volumes (time-series forecasting) or some external (exogenous) information is used to help with the prediction, assuming that there is a correlation between this external information and future traffic volumes. For example, we can employ only the past *n* traffic volumes to predict the next one or, in addition, we can use other information related to traffic, such as: device IDs, time/date, type of customer, geographical area, etc... In this latter case, the ML methods are more appropriate, since only a few and complex time-series forecasting methods use exogenous information e.g. Auto Regressive Integrated Moving Average with eXogenous variables (ARIMAX) [8].

In order to perform the prediction, it is necessary to perform a time discretization, with pre-defined time windows (time-slots) used to aggregate traffic. Once the traffic is aggregated and discretized in time-slots, the prediction consists in forecasting the traffic volume for the next time-slot (simple forecasting) or the future k nearest time-slots (k-ahead forecasting). Depending on the application the time-slots can be seconds, minutes, hours,…

M. Lopez, B. Carro and A. Sanchez are with Dpto. TSyCeIT, ETSIT, Universidad de Valladolid, Paseo de Belén 15, Valladolid 47011, Spain; (mlopezm@acm.org; belcar@tel.uva.es; antoniojavier.sanchez@uva.es).

Application of statistical analysis methods for traffic prediction has been the usual direction, but the trend has recently moved to ML methods [2]. From the point of view of an ML model, traffic volume forecasting can be seen as a regression problem where a regression function performs a simple forecasting (one-output regression) or a k-ahead forecasting (multi-output regression), using as predictors the past traffic volumes or exogenous variables, all of them considered as model features.

Therefore, in the more complex setting, an ML method applicable to traffic volume prediction for networking should be a multi-output regression model able to incorporate all kind of features as predictors. Neural networks (NN) are particularly good candidates for this task, considering also the possibility to use recent NN architectures specifically suited for sequence data (Recurrent Neural Networks (RNN), i.e. Long Short Time Memory (LSTM) networks)[7]. NN models can support multi-output regression and can be deployed in high-performance execution environments using current available platforms for deep learning (e.g. Tensorflow). Other extremely successful models for regression problems are based on bagging [9] (e.g. Random Forest) and boosting [9] (e.g. AdaBoost) algorithms, and particularly gradient boosting (GB)[10,11]. These models present more difficulty to address a multi-output regression [12], being the easier solution to train a specific model for each output value. Therefore, considering the benefits provided by the different algorithms it seems interesting to explore the possibilities of combining the ideas and principles of GB and NN in a single model. This paper shows that, with an adequate combination of GB and NN models, we are able to propose a novel architecture that provides better prediction metrics than other models with limited computation times. We call this architecture generic additive network (gaNet), being applicable to any supervised regression problem. gaNet is a stagewise additive model trained with the residuals between the ground-truth and the predicted outcomes, as gradient boosting models do.

gaNet has also connections with residual networks [13] and stacked models [14,9]. A residual network is formed by a sequence of layers where the input to the internal layers is built by adding the outputs of more than one previous layer. The architecture of gaNet resembles a residual network with different network topology and connections, and shares with residual networks its robust convergence behavior during training, even with an extreme number of layers added to the network. It also has a configuration similar to a stacked model, which is formed by a succession of stages where the output of a previous stage becomes the input for the next stage, and each stage is trained separately, sharing the same ground-truth outcome for all stages. The regularization effect observed in stacked models (a type of ensemble models) is also observed in gaNet models.

To evaluate gaNet and compare it with other models, we apply it to the difficult and important problem of k-ahead network traffic forecasting at the device level, using only recent and short traffic values. This is a practical problem considering current highly demanding networks with a large number of devices and volatile network conditions (e.g. IoT networks), where acquiring and storing longer series of traffic volumes per device is costly, and, where the changing conditions of the network make information that goes far back in time less useful. Besides, the trend to network services personalization makes it necessary to manage traffic at the device level as part of SLA contracts. In our case, we compare all models considering the traffic prediction for the next 6 time-slots using the values of the previous 24 time-slots with a time-slot duration of 1-hour. The experiments have been performed with real IoT traffic data obtained from a mobile operator [15] (generated specifically for device traffic forecast). The data included time-stamped traffic records with dummy (obfuscated) mobile identifiers. Considering the application in an operational environment where privacy is paramount, the objective has been to not to use any information other than the time-stamped traffic volumes.

In order to evaluate the performance of gaNet, we compare it with alternative ML models applicable to this problem, such as: recurrent neural networks in isolation [16] or together with convolutional neural networks[17,18], fully connected neural networks with different number of hidden layers, seq2seq models [19] with and without soft attention [20,21] and classic ML models as random forest and linear regression.

The reasons for the good performance of gaNet models are related to their similarities to residual networks, ensemble (stacked) models and gradient boosting methods. Each of these models provides different properties to the algorithm. Ensemble models are perfect candidates for variance reduction, while gradient boosting models address the bias problem, and residual networks have been shown to solve the convergence and training problems associated with deep neural architectures. The combination of these different concepts provides the necessary balance to tackle both the problem of underfitting and overfitting, which are present in any machine learning algorithm.

The capacity to predict network traffic is crucial in many networking domains: traffic control, network adaptation, congestion and admission control, and network management [2,22,23,24]. All of these areas require a network traffic forecasting service that is transversal and invoked by many other services as presented in Fig 1. Depending on the time-scale used for the prediction (milliseconds, seconds, hours, days, ...), the use of the prediction may be different, being generally the case that the larger time-scales are associated with information administered by humans while the smaller ones are handled by automated processes. In our case, the time-scale used is 1-hour, which is in the middle of the separation of human vs. automatic management and can be useful for both purposes [15].

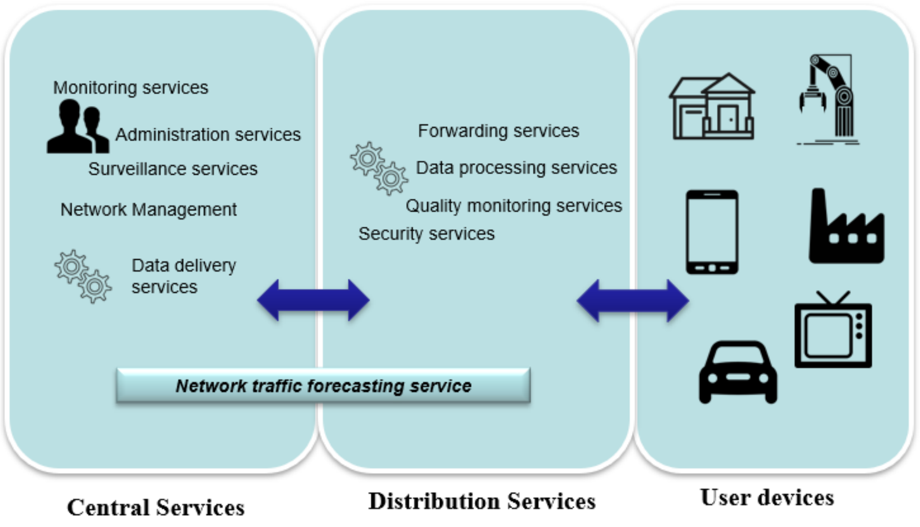


Figure 1. Services framework for an IOT application and position of the traffic forecasting service

As a summary, we consider that this work achieves several objectives with the following contributions: (1) To present a novel NN architecture for multi-output regression problems based on GB principles, that is easy and fast to train, also fast with prediction times and providing excellent prediction results. (2) To show that a lego-like assembly of similar structures which follows a stagewise additive model shares many of the good properties observed in residual networks, ensemble models and gradient boosting methods. (3) To analyze different architectural alternatives for the new model. (4) To propose a model that can be deployed in modern high-performance platforms (e.g. Tensorflow). (5) To provide an extensive comparison of our proposed model with alternative ML models for the prediction of volume of traffic for a real IoT network.

The paper is organized as follows: Section 2. identifies related works. Section 3 presents the work performed and the models analyzed in the paper. Section 4 shows the results obtained and finally, Section 5 provides discussion and conclusions.

2. RELATED WORKS

As mentioned in the previous section, the model presented in this work (gaNet) has connections with several other models: gradient boosting [10,11], residual networks [13] and stacked models [14,9]. gaNet incorporates ideas from GB models but is different from boosting models [9]. Boosting models consist of an additive boosting combination of weak leaners. They assign a weight to each sample based on the error obtained by that particular sample. Following the paradigm of boosting models, there is a series of works that apply NN architectures as weak learners [25,26].

The combination of gradient boosting with trees and neural networks is explored by recent research. In line with the work proposed here, [27] proposes an implementation of gradient boosting using Tensorflow, which is a popular deep learning platform. They implement a traditional gradient boosting algorithm based on trees as weak learners. Authors in [28] propose a model based on gradient boosting with neural decision forest as 'weak learners'.

There are several works that establish the connection of residual networks with boosting and gradient boosting models. In [29] is presented a theoretical connection between residual networks and boosting theory, where the final layer of a residual network can be assimilated to a layer-by-layer boosting method. Similarly, [30] provides a link between gradient boosting and residual networks proposing a new gradient boosting algorithm based on the notion of functional gradients.

Considering that gaNet is an algorithm for multi-output regression, [12] presents a detailed review of the difficulties inherent in multi-output regression algorithms.

Even when it is not the main topic of this work, we present the results of applying several advanced algorithms for network traffic prediction such as: sequence to sequence [19], attention [20,21], convolutional networks (CNN)[31] and recurrent networks (LSTM)[32] and, for which there is a good number of previous works on network traffic forecasting [7].

The literature presenting advancements on network traffic forecasting is extensive, studying many different types of algorithms: (1) statistical analysis methods, either linear (ARMA, ARIMA,..) , non-linear (GARCH,..), or based on probabilistic models

(Hidden Markov Models), and (2) ML methods, either based on neural networks(CNN,) or not (Random Forest, …)

Regarding statistical/probabilistic methods, in [33], an ARIMA model is applied to predict load demands and adjust the provisioning of resources for applications in a cloud data center. Authors in [34] study a combination of ARIMA and Generalized Auto Regressive Conditional Heteroskedasticity (GARCH) showing that it provides better prediction accuracy than a fractional auto-regressive integrated moving average (FARIMA). In this work the goal is to obtain good prediction accuracy under short-range dependence (SRD) and long-range dependence (LRD) characteristics in the time-series. They propose a model that is able to incorporate the LRD in the model by combining ARIMA (linear model) with GARCH (non-linear model), resulting in a model with better LRD properties than FARIMA. In [35] a study is made on the deterioration of the prediction accuracy in long-range predictions using ARMA and the Markov Modulated Poisson process (MMPP). A hidden Markov model is used in [36] to predict traffic volume based on flow statistics.

All statistical/probabilistic models depend to a large extent on the stationarity of the data used for prediction and the prediction time range. They adjust/train the model to a particular time-series, and they are not applicable to models that need to predict a set of time-series with a single model training.

Regarding ML methods, the research presented in [37] uses convolutional neural networks (CNN) to predict short-term changes in the amount of traffic that crosses a data center network, the non-linearity provided by CNN surpasses the ARIMA results. A recurrent neural network is applied in [38] to the prediction of multivariate time-series when the time-series have missing values. In [39] the authors propose an ensemble of a convolutional and a recurrent network, which are trained together end-to-end, and where the outputs of the two networks are concatenated and given to a final softmax layer to achieve time-series classification. They compare the final model with and without an attention mechanism for the recurrent network. Similarly, [40] presents a complex two-stage attention mechanism for the prediction of multivariate time-series with a recurrent neural network. The first stage is applied to the signals dimension and the second to the time dimension. They achieve better prediction metrics than with simple recurrent networks, other non-linear models and ARIMA. Authors in [15] present a study on time-series classification for IoT traffic using several ML methods e.g. Logistic Regression, Random Forest, Gradient Boosting Method (GBM) and Bayesian Logistic Regression, comparing them with the results of classical time-series methods e.g. Hidden Markov Model (HMM), Exponential Smoothing, ARIMA and ARIMAX. ARIMAX presents the best prediction results but it is extremely expensive in terms of training and prediction times. The work shows that the ML methods have better operational results, since their prediction accuracy is close to the ARIMAX model with much smaller requirements for training and prediction times.

The application of ML methods to time-series forecasting is a complex issue, where there is an active research activity [1,2,41] in which the work presented here is framed.

Time-series forecasting in highly demanding environments is a challenging problem, since it is not only necessary to deliver an accurate future traffic forecast but also to deliver it in time. In this line, it is important that the data acquisition [42] and forecasting algorithms are efficient [22], but also that they can be distributed and deployed in modern computing platforms [43], with the ultimate challenge of accommodating network services for real-time analytics of massive IoT data [41].

3. WORK DESCRIPTION

In this Section we provide details of the dataset used to carry on the experiments and the novel architecture that is proposed to perform volume of traffic predictions for data networks.

The dataset employed and the proposed model are presented on sections 3.1 and 3.2 respectively.

3.1 Selected dataset

The selected dataset is presented in [15] and corresponds to real IoT traffic from a mobile operator. The data consists of separate activity records (bytes transmitted and received) for 6214 mobile devices during a consecutive period of approximately 26 days (632 hours). In addition to the traffic volumes, the data included the obfuscated mobile identification (SIM) and the time stamp of traffic records.

The traffic per device has been aggregated in time slots of 1-hour with a total of 4076384 values of traffic in a time-series structure per device. This data has been further separated into two datasets: training and test, with no data sharing between both datasets. The training dataset corresponds to the activity of all devices for the first 512 hours (21.3 days) and the test dataset for the following 120 hours (5 days).

The objective of the experiment carried out with the datasets is to forecast the volume of network traffic of a device (aggregated input and output traffic) for the next 6 hours in time slots of 1-hour, having as predictors the network traffic of the previous 24 hours, in time slots of 1-hour also. Therefore, the final arrangement of the datasets for this supervised regression problem is a training dataset of 105638 samples with 24 real values as predictors and 6 real outcome values (multivariate multiple regression), and, a test datset of 24856 samples with the same number of predictors (24) and predicted values (6).

It is interesting to mention two important challenges of the dataset: a) the unbalanced nature of the distribution of traffic, and,

b) the wide value ranges for the traffic volumes, with values extending over several orders of magnitude. The second challenge is addressed by applying a log1p transform to the traffic volumes. The log1p transform is based on the following expression: $\log(x+1)$ where x is the transformed variable and log is the natural logarithm. The first challenge is more difficult and poses a difficulty to the prediction algorithm.

Fig 2. presents graphically the unbalanced distribution of network traffic for all devices. The chart provides a histogram of traffic (log1p transformed) transmitted and/or received by any device in a 1-hour aggregation time-slot. We can observe that most of the traffic is associated to devices with zero aggregated activity in 1-hour time slots, with some important peaks of traffic for the ranges of a few tens to thousands of bytes and a long tail for extremely large and rare traffic volumes. Fig 2 shows also the mean, median and standard deviation for the traffic volume. The transformed median value (4.1109) corresponds to a real traffic volume of 60 bytes aggregated in 1-hour time-slot.

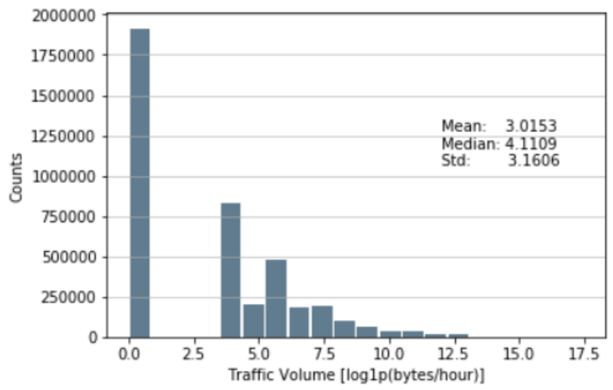


Fig 2. Distribution of network traffic volumes (log1p of the aggregated bytes transmitted in 1-hour time slots).

Considering the high number of idle connections in the 1-hour time slot aggregation period, it is also interesting to show the distribution of active connections (connections with a non-zero traffic volume in the 1-hour period) considering the day of week (Monday to Sunday) and the hour of day (0 to 23 hours). Fig 3 provides these distributions of active connections. We observe the expected hour/day cyclic activity corresponding to the distribution of working hour/days.

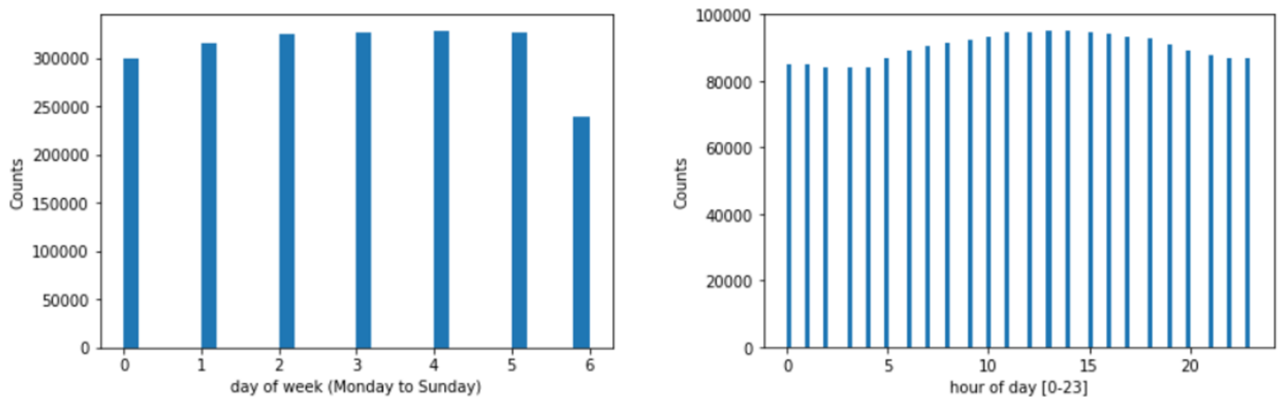


Fig 3. Distribution of network traffic active connections per day of week and hour of day.

3.2 Model description

This section presents the gaNet model at different levels of detail.

3.2.1 gaNet model: introduction

As mentioned in the Introduction, the initial intention of this work is to implement the ideas proposed by the GB algorithms using neural networks (NN) as substitutes to the one-step decision trees (stumps) used in GB. As opposed to decision stumps, an NN is not a weak learner implying that some of the properties of GB are expected to be modified. Nevertheless, using NN as the learning building blocks of GB may provide many advantages if we could leverage the end-to-end training capabilities given by stochastic gradient descent (SGD) that is applied for the most successful deep learning architectures. A GB model is an iterative stagewise additive model formed by pretrained learning blocks. The learning blocks ('learners') try to approximate (regression) an output made by the differences between the ground-truth output and the output from the model built so-far, these differences are called residuals. An additional multiplicative parameter is optimized in each step of the algorithm. Both the regression learners and the multiplicative parameters are fitted once and added to the model which is not fitted end-to-end, requiring two optimizations per step. These optimizations can be implemented with two separate NNs per step, one to implement the 'learner' and another to optimize the multiplicative parameter as a simple linear regression problem. We have implemented this approach in two different variants in what we call gaNet Type I and II models (Section 3.2.2). However, this approach is not ideal for many reasons: large number of NN models to administer, slow training of separate NNs, no end-to-end training and difficulty in deploying the model in high performance environments.

Therefore, the next objective has been: (a) how to transform the initial step-by-step training into an end-to-end training, with the aim of converting an original GB model implemented with NNs into a pure deep learning model that supports end-to-end training and prediction, and (b) to study how this important transformation of the original ideas of GB affects the resulting model in terms of prediction capabilities and computation times. The resulting architecture following these objectives is called gaNet Type III model.

The last gaNet architecture is the Type IV, which is a Type III generalization where the learning blocks are not all similar, as in Type III, but each one can follow a different model. This provides an additional level of flexibility. In Section 4 we show that a Type IV model presents the best results considering the performance metrics: MSE, R^2 and NRMSD. The different variants of the gaNet architecture obtained by successive refinements are presented in detail in Section 3.2.2.

In order to facilitate the maintenance and management of the generated models, our intention has been to create a single model to carry out the forecasting for all devices (incoming and outgoing traffic) and all output values (multi-output prediction). As mentioned before, considering the usual methods for time-series forecasting: a) time-series models based on statistical analysis methods (e.g. ARIMA) that adjust the parameters of a lineal model to the next value of the time-series based on the previous n inputs, and, b) machine learning (ML) models (e.g. Random Forest, Linear Regression...), that treat the forecasting problem as a multi-output supervised regression problem; in both cases, it is difficult to apply a single model to a multi-output prediction from traffic generated by a number of different devices. In the case of ARIMA, we would need to fit the parameters of an ARIMA model per device or to build a VARIMA model that is too complex. Moreover, the parameters of an ARIMA model are adjusted to the n previous values of the time-series taken as predictors; in our case n is a small value compared with the total length of the time-series implying the necessity to adjust several ARIMA models for the time-series generated by each device. Similarly, in the case of ML solutions, with classical models which produce a single predicted value, it would be necessary to construct a model for each predicted value, this being the simplest solution to the difficult problem of multi-output regression in machine learning [12]. To solve these difficulties, our proposed model is able to produce a single model that performs k-ahead forecasting with time-series generated by different devices.

3.2.2 gaNet model: details

This section presents the proposed architecture in detail. In order to arrive to the final model, we provide a description of the successive models that were considered from the original ideas of a gradient boosting (GB) model to the different variants of our proposed model (gaNet). In this section we will identify all these models as gaNet models, giving them an additional type identifier to help in their subsequent reference.

As mentioned in the introduction, the different gaNet models are formed by the composition of building blocks where all the building blocks are made of NN architectures. The main advantage of using NNs is the possibility to use stochastic gradient descent to allow an end-to-end training of a complete model with the additional benefits of current research advances in deep learning architectures and their implementation platforms (e.g. Tensorflow, Theano…).

We will present the different gaNet models in increasing divergence from the initial GB ideas to architectures more similar to stacked or residual networks. The ideas behind a GB model [10,11] are presented in Fig 4. The gaNet architectures that follow more closely this model will be referenced as Type I models. A GB model is a supervised machine learning model having as inputs a set of samples $\{(x_i, y_i)\}$, where x_i is a vector of features used as predictors for training and y_i is a vector of expected results (ground-truth values). We assume to have a number N of such samples. In Fig 4. we call the vector of features x_i as X and the vector of results y_i as Y. The algorithm for GB is obtained by repeating iteratively the sequence of 3 steps presented in Fig 4. It starts by selecting an initial fixed output value (F0) that we choose as the value that minimize the quadratic error between F_0 and our expected values Y. This value corresponds to the mean value of Y. Starting from this initial value: F_0, we obtain the first residual: $r_1 = Y - F_0$ and we train a learner f_1, with X as predictors and r_1 as our ground-truth value. In Fig 4 we identify the learner as a learning block (building block of the model). To train the learner f_1 we use the quadratic error loss between r_1 and h_1, that is defined as: $L_2(r_1, h_1) = \sum_{i=0}^{N}(r_{1,i} - h_{1,i})^2$, where h_1 is the output of f_1. Once the learner is trained, we calculate the value of a multiplicative parameter: γ_1, that minimizes the quadratic error loss: $L_2(Y, F_0 + \gamma_1 * h_1) = \sum_N (Y - (F_0 + \gamma_1 * h_1))^2$, where again h_1 is the output of f_1 when is trained with the residuals r_1 using X as predictors, and the sum is extended to all the samples of the dataset. Once the optimum value of γ_1 is obtained, we proceed to add the value of $\gamma_1 * h_1$ to F_0, calling the resulting value as F_1. This new value F_1 will be used in place of F_0 in the next iteration of the algorithm until a predefined number m of building blocks (f_m) are added to the model. In each iteration of the algorithm the successive values of F_m, r_m and γ_m are updated following the steps shown in Fig 4.

The algorithm for the Type I model consists of a forward stagewise additive model of independently trained building blocks (f_m), and independently optimized parameters (γ_m). This imply the need to perform two optimization steps for each building block that is added to the model. We use a different NN to train/optimize f_m and γ_m since we perform all optimizations based on SGD. Fig. 4 outlines how is implemented the NN to train f_m, using the vector of predictors X as the input and the residual r_m as the ground-truth output, but it does not describe the NN that is used to optimize γ. Fig 5 describes this latter NN. As we can see in Fig 5, the NN used to optimize the parameter γ is a simple NN with no hidden layers and a linear output layer. Once this NN is trained, its weights correspond to the optimized values of the multiplicative parameter γ. As expected, the loss function for this NN is $L_2(Y, F_m) = L_2(Y, F_{m-1} + \gamma_m * h_m)$, where γ_m are the weights of the NN. Therefore, the weights of the NN in Fig. 5 adjust their values during training to perform the optimization sought: $\gamma_m = \underset{\gamma}{\operatorname{argmin}} L_2(Y, F_{m-1} + \gamma * h_m)$. It is important to mention that the NN used for the optimization of the γ parameter substitutes the line search methods [10] normally used for this aim by the classic GB models

Just to clarify the connection between the different elements that participate in the gaNet Type I architecture, in Fig. 6 is presented an expansion of a generic block of the architecture presented in Fig 4. In this last figure it is easier to differentiate the two NNs used and their different purpose.

An important consequence of the implementation of the GB methods, as the gaNet Type I model does, is that it is very easy to extend the model to multi-output regression problems. In a multi-output problem, the outcome Y is not a scalar but a vector of values (in our case is a vector of 6 values, but could be any number of values). To perform this extension, it is simply required to extend the different loss functions with a sum taken over all the values of the expected result, that is: $L_2(Y, \hat{Y}) = \sum_{j=0}^{d} \sum_{i=0}^{N} (Y_{j,i} - \hat{Y}_{j,i})^2$, expanding the addition to all dataset samples (N) and output dimension values (d). This is the approach we have for all gaNet models presented here. However, with the intention of simplifying formulas and presentation, we obviate the use of the additional sum taken over the output values.

The possibility to use gaNet for multi-output regression problems is an important property of these models, since the original implementation of GB does not support it, at least in its original form, and, in general multi-output regression is a challenging problem [12].

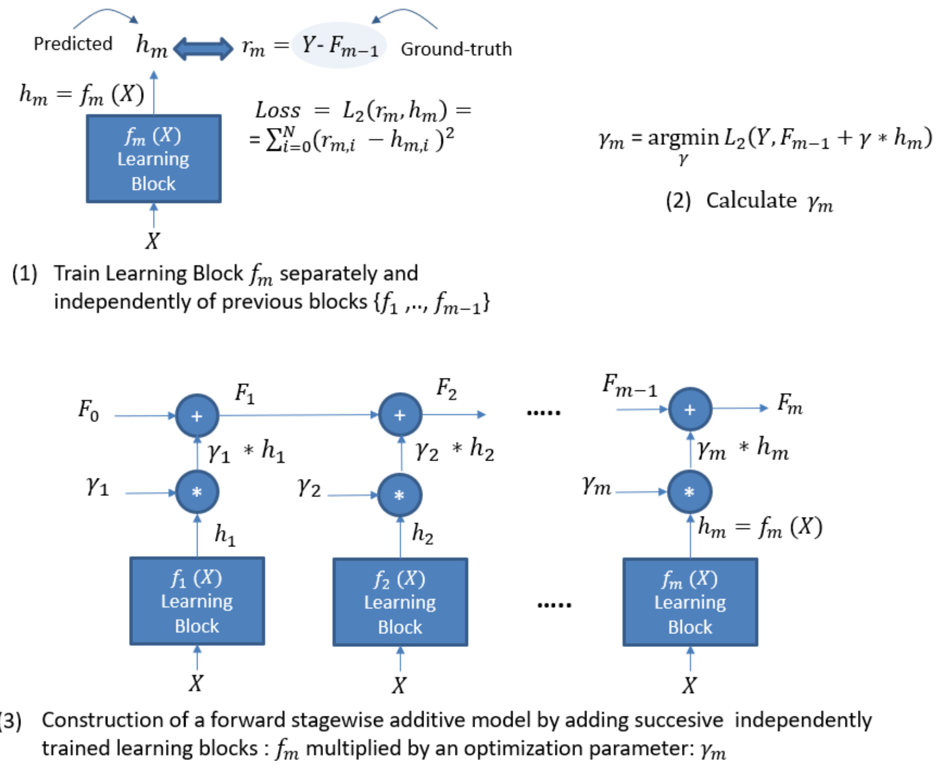

Predicted
h_m
$r_m = Y - F_{m-1}$
Ground-truth
$h_m = f_m(X)$
$f_m(X)$ Learning Block
X
$Loss = L_2(r_m, h_m) = \sum_{i=0}^{N}(r_{m,i} - h_{m,i})^2$
$\gamma_m = \underset{\gamma}{\operatorname{argmin}} L_2(Y, F_{m-1} + \gamma * h_m)$
(2) Calculate γ_m
(1) Train Learning Block f_m separately and independently of previous blocks $\{f_1, .., f_{m-1}\}$
F_0
F_1
F_2
F_{m-1}
F_m
$\gamma_1 * h_1$
$\gamma_2 * h_2$
$\gamma_m * h_m$
γ_1
γ_2
γ_m
h_1
h_2
$h_m = f_m(X)$
$f_1(X)$ Learning Block
$f_2(X)$ Learning Block
$f_m(X)$ Learning Block
(3) Construction of a forward stagewise additive model by adding succesive independently trained learning blocks : f_m multiplied by an optimization parameter: γ_m

Fig 4. gaNet Type I architecture: Training the blocks one by one, optimizing γ separately.

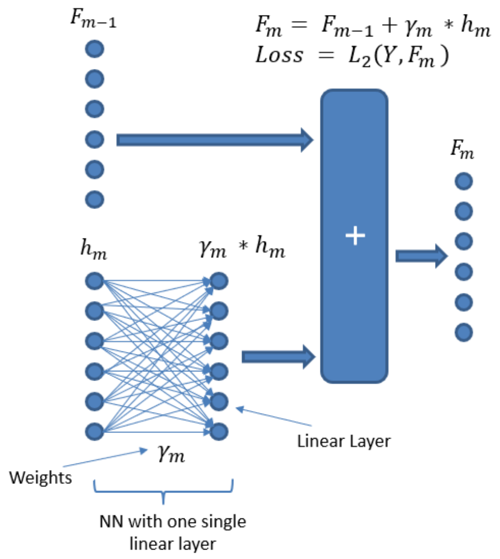


Fig 5. NN used to obtain γ_m

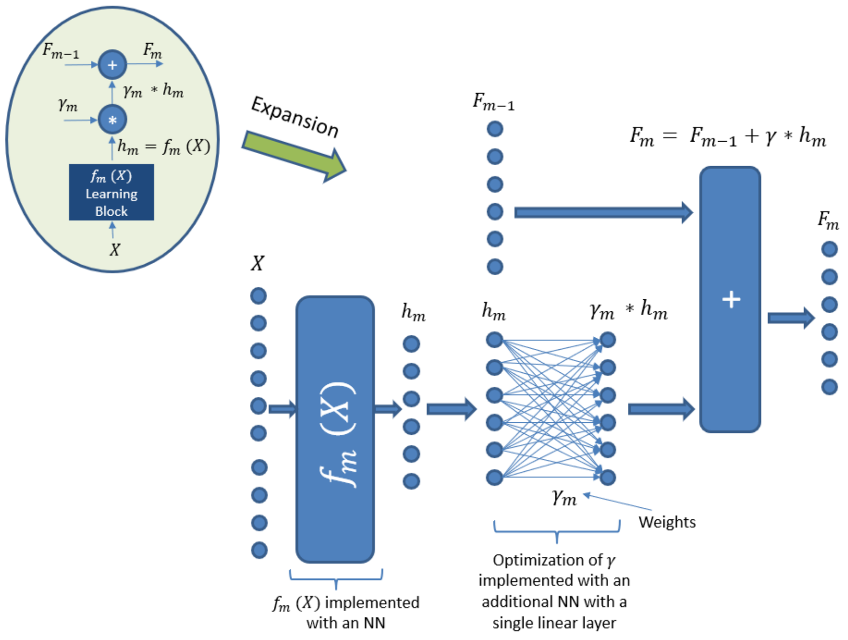


Fig 6. Expansion of one iteration step in gaNet Type I architecture

The two optimization steps (f_m training and $\gamma_m$optimization) per learning block that is carried out in Type I architectures, can be integrated into a single optimization step during the training of each building block, by eliminating the parameter γ , and assuming that the output: h_m can substitute: $\gamma_m * h_m$. The model obtained following this procedure is depicted in Fig. 7, and it is called Type II model. In this case, we end up with a single NN to train, greatly simplifying the entire architecture. The two NNs presented in Fig 6 for the gaNet Type I architecture are reduced to a single NN to train exclusively the function f_m .

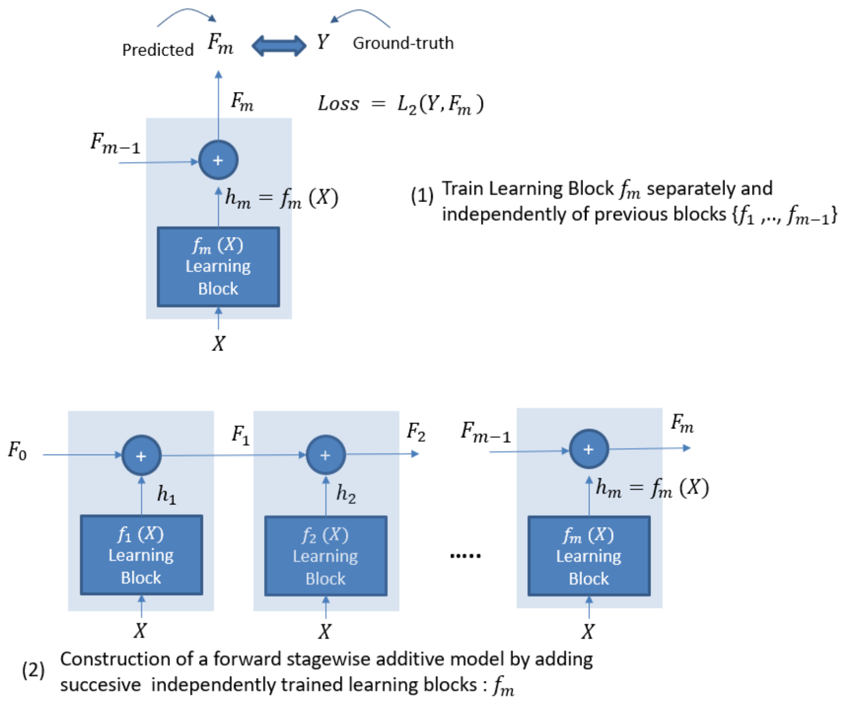


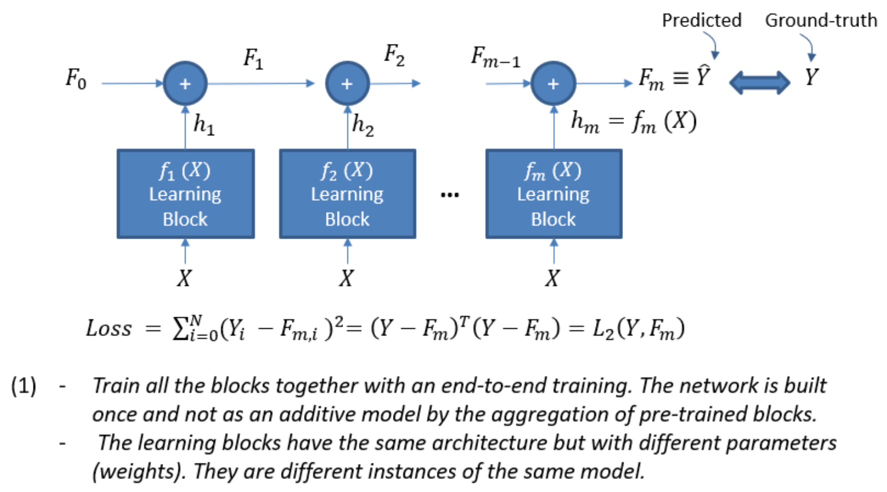
Fig 7. gaNet Type II architecture: Training the blocks one by one, incorporating the optimization of γ into the training of each block.

Both Type I and II models can be considered as variants of GB models where the simple decision stumps of the original GB models are substituted by more complex architectures (learning blocks). In our case, the learning blocks chosen for this work consist on architectures based on deep learning models with different combinations of convolutional, recurrent and fully connected layers.

Type I model requires two optimization steps for each learning block added to the model. Type II reduces this burden to a single optimization step per building block, but our aim would be to integrate all optimizations (training) in a single end-to-end process, not one per build block. Moreover, in Type I and II models, the training of each building block is performed independently, which means that, once a learning block is added to the model, its parameters are frozen and will not be adjusted during the successive additions of new learning blocks. This approach facilitates the optimization problem per building block but does not provide an optimum solution. In Fig. 8 we propose a solution to this problem by performing a single end-to-end training of all building blocks together, with a quadratic loss function on the final residuals formed by the difference between the ground-truth outcome values and the output values of the last block: F_m. The model proposed in Fig. 8 is called Type III model.

A Type III model is a single model created by the composition (additive composition) of identical structures (learning blocks). It is an end-to-end trained model. All the learning blocks of the model are trained together end-to-end. This model has two inputs $[F_0, X]$ and one output F_m. The variable X is a single variable although it is internally reused several times. This is very different to Type I and II models that are formed as composition of several trained-as-you-progress learning blocks that have one single input X and output h_m. In Type I and II models, each learning block is trained separately and keeps its weights frozen once they are trained.

Type III models deliver several advantages over Type I and II: a) They provide faster training and prediction times. b) They show excellent convergence behavior during training which allows for deeper models. c) They have a regularization effect similar to stacked models. And, d) they present excellent prediction results, not only compared to Type I and II models but in comparison with other state-of-the-art time-series forecasting models (e.g. recurrent neural networks, seq2seq, attention...) (Section 4).



$$Loss \ = \sum_{i=0}^{N}(Y_i \ - F_{m,i})^2 = (Y - F_m)^T(Y - F_m) = L_2(Y, F_m)$$

(1) - *Train all the blocks together with an end-to-end training. The network is built once and not as an additive model by the aggregation of pre-trained blocks.*
- *The learning blocks have the same architecture but with different parameters (weights). They are different instances of the same model.*

Fig 8. gaNet Type III architecture: Training all the blocks together with an end-to-end training.

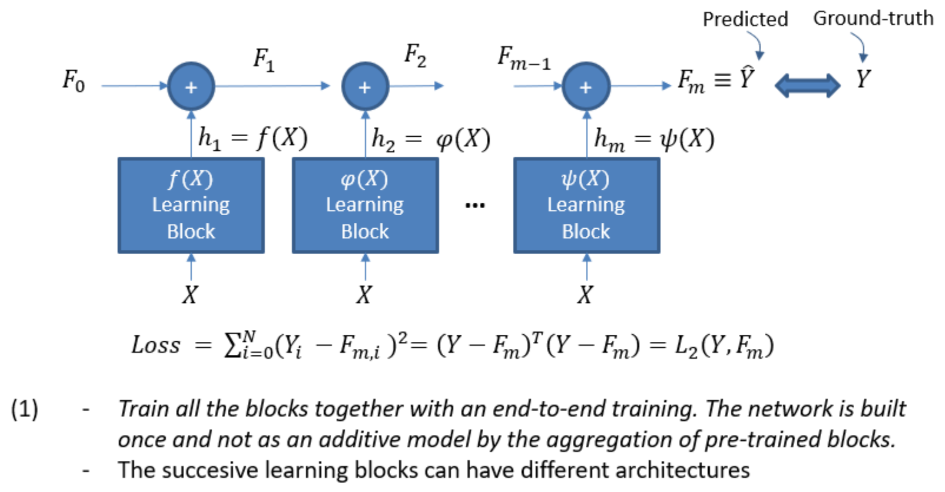
We can define some variants to Type III model. The Type III-A model is a variant of Type III with a similar structure to a Type III model but with a different loss function. The loss of a Type III model is $L_2(Y, F_m)$ while the loss of a Type III-A is $\sum_{j=1}^{m} L_2(Y, F_j)$, that is, the sum of quadratic losses between the ground-truth outcome values and each of the intermediate outputs of the aggregated model: F_j, where m is the total number of learning blocks. The intention of this variant is to try to reproduce the logic of Type I and II models that optimize independently the intermediate outputs F_j of the model. This variant has provided quite good prediction results (Section 4).

Another variant of the Type III model consists in having all the learning blocks sharing their weights. That implies that all learning blocks are identical: identical structure and weights. This variant is called Type III-B. The results obtained with this variant are worse in terms of prediction performance than with the original Type III with different weights per learning block (Section 4).

Finally, Fig. 9 presents the Type IV architecture which is an extension of Type III giving freedom to the building blocks to choose different structures. This architecture has a nature similar to that of a stacked model, but with the peculiarity that the input of each constituent model is not the output of its previous model, but the addition of this output with all the previous aggregate outputs. There are endless possibilities to choose the different order and architecture of the learning blocks in Type IV models.

The last variant of gaNet studied is a variant of the Type IV model with a loss function similar to the loss function provided to the Type III-A. We call this model Type IV-A.

The results obtained for all the above mentioned gaNet types, with different selections of learning blocks architectures, are provided in Section 4.



$$Loss \ = \sum_{i=0}^{N}(Y_i \ - F_{m,i})^2 = (Y - F_m)^T(Y - F_m) = L_2(Y, F_m)$$

(1) - *Train all the blocks together with an end-to-end training. The network is built once and not as an additive model by the aggregation of pre-trained blocks.*
- The succesive learning blocks can have different architectures

Fig 9. gaNet Type IV architecture: Training all the blocks together with an end-to-end training and with learning blocks of different architectures.

Fig. 10 shows a sample of the various architectures used to create the learning blocks. We can see that there is no restriction to the chosen architectures as far as they comply with the types and dimensions of the input (X) and output (h_m) vectors.

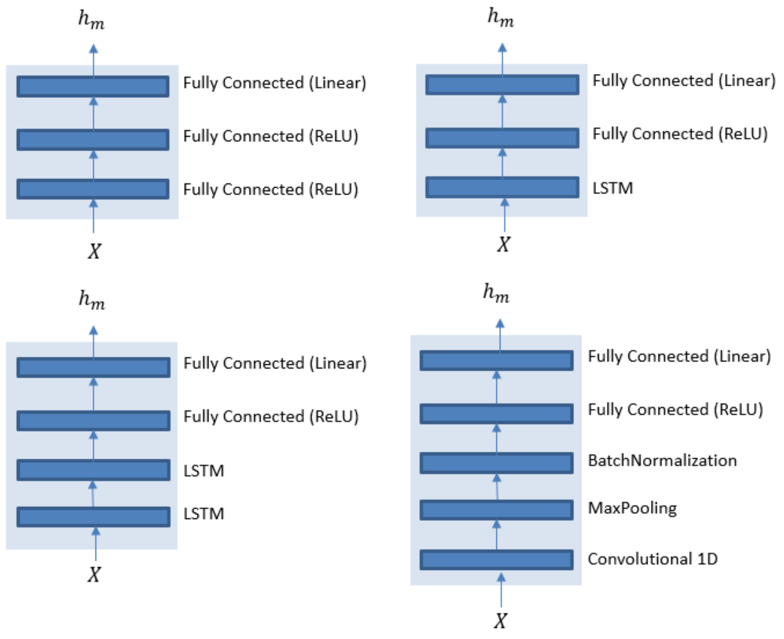


Fig 10. Examples of learning blocks used by gaNet models

Considering the results on the different prediction metrics provided in Section 4, in Fig. 11 is presented the best architecture for the mean squared error and R^2 metrics. It is interesting that, even when the model is quite big, we have not experienced convergence problems when training. The loss function used is a simple quadratic error between the model output and the ground-truth values (Fig. 11).

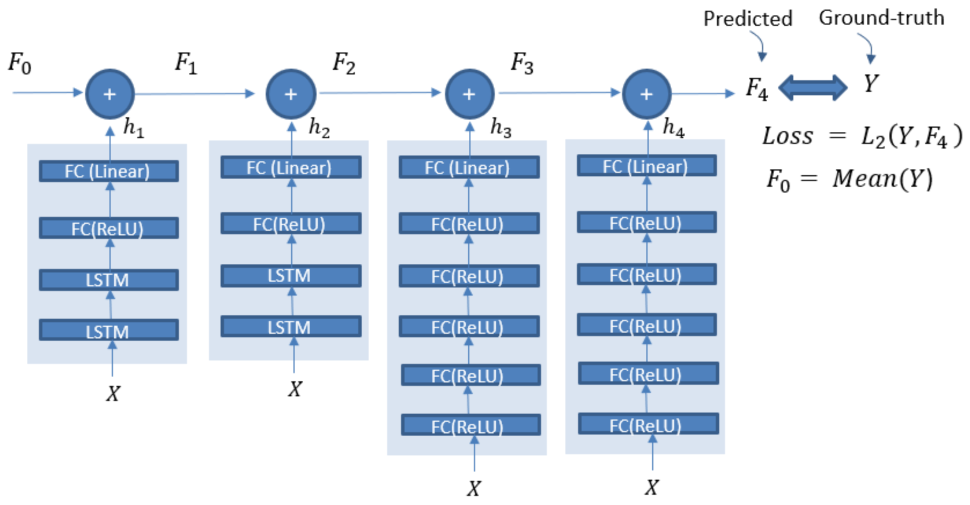
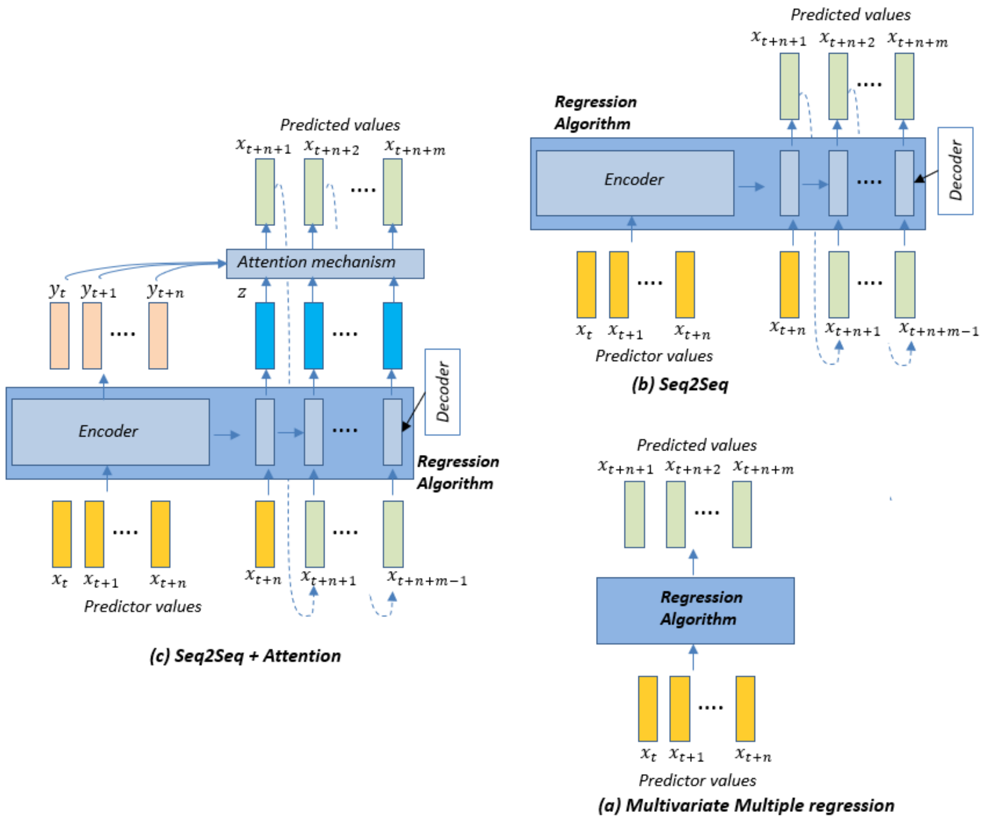


Fig 11. gaNet Type IV architecture with best results on mean squared error and R^2 metrics.

In order to position gaNet in comparison with other alternative regression models that can be used for time-series forecasting, we provide a classification of the different models in three groups (Fig. 12). The models in group (a) use a regression algorithm to forecast the m future time-slot values using as predictors the past n values. Group (b) models are based in a sequence to sequence (se2seq) architecture [19] that is based in encoding all the information provided by the past n values (predictors) in a single state vector which is iteratively used by a decoding block. The decoding block produces the m future values one by one using as input the previous forecasted value. This group of models can be enriched if we add the information generated by past values, which can be considered as a type of memory. This is the strategy followed by group (c) models. Group (c) models introduce an attention mechanism to the seq2seq models. The attention mechanism consists of an additional layer that performs a similarity comparison between the initial output and the information related to the past history (n previous values used as predictors). The similarity operation can be smooth and differentiable (soft attention) or based on a choice and not differentiable (hard attention). In this work we have used soft attention based on a softmax applied to the dot product of the initial output with all the intermediate results produced by the n values used as predictors [20].

As described in the Introduction, we have not considered the use of classic time-series models (e.g. ARIMA…) because they are based on the forecasting of a specific time series and not on a single model that can be used for the forecasting of any time series of a dataset of time-series used for training. For the dataset used for this work, that would have implied to have finally 105638 ARIMA models.

Group (b) and (c) models are multi-output in nature, that is, they are intended to produce an output with multiple values associated to a single input. Group (a) models are based on different regression algorithms and do not need to support multi-output. In this case, for group (a) models that are not multi-output we need to create a different model for each output value. gaNet belongs to the models in group (a) with multi-output capabilities. Other models in group (a), which are also studied in this work (Section 4), belong to the class of models that need to create as many models as the number of output values (e.g. Random Forest, Linear Regression…). In these cases, there is an associated toll in terms of the number of models to train and maintain.



Predicted values
x_{t+n+1} x_{t+n+2} x_{t+n+m}
Attention mechanism
y_t y_{t+1} y_{t+n}
z
Encoder
Decoder
Regression Algorithm
x_t x_{t+1} x_{t+n}
x_{t+n} x_{t+n+1} $x_{t+n+m-1}$
Predictor values
(c) Seq2Seq + Attention
Predicted values
x_{t+n+1} x_{t+n+2} x_{t+n+m}
Regression Algorithm
Encoder
Decoder
x_t x_{t+1} x_{t+n}
x_{t+n} x_{t+n+1} $x_{t+n+m-1}$
Predictor values
(b) Seq2Seq
Predicted values
x_{t+n+1} x_{t+n+2} x_{t+n+m}
Regression Algorithm
x_t x_{t+1} x_{t+n}
Predictor values
(a) Multivariate Multiple regression

Fig 12. Different typology of time-series forecasting models studied in this work.

3.3 Performance for time-ahead prediction

As expected, all the models experience a degradation of performance depending on the time-ahead distance to the specific time-slot value forecasted. For predictions more distant in time, the degradation of prediction performance is greater. It is interesting that this degradation is not linear but follows a chair-shape curve with a strong degradation at the beginning and end of the time-ahead prediction window with an almost plateau evolution in the middle of this window. It is also interesting that this evolution is followed by all algorithms considered in this work, with subtle but important differences in the slopes of the degradation curves and length of the middle plateau.

Fig. 13 present the time-ahead prediction curves for different algorithms studied (Section 4). We can see as the gaNet models are able to maintain good predictions (R^2) for a longer time, which justifies their better performance.

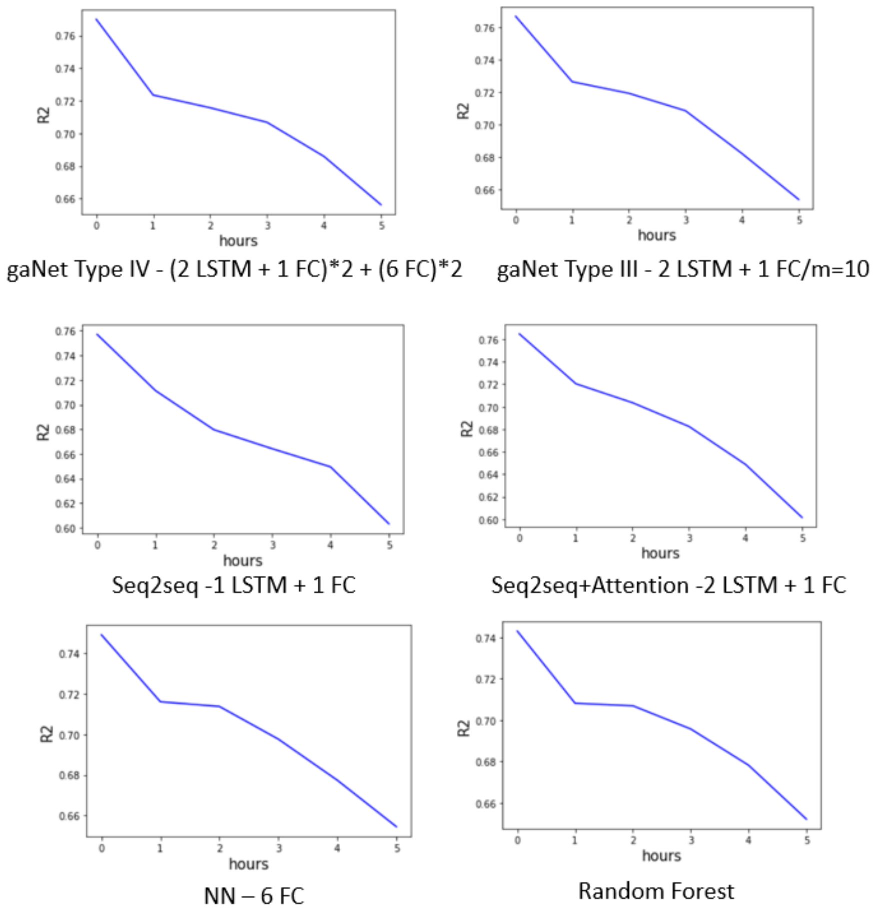


Fig 13. Evolution of R^2 metric vs. time-ahead prediction hours.

4. Results

This section presents the results obtained by applying different algorithms to the dataset described in section 3.1 with the aim to compare their forecasting capabilities using different metrics. We analyze the results from a time-series multi-step ahead forecasting problem treated as a multivariate-multiple prediction problem.

We have grouped the different algorithms into different classes (Fig. 14): (a) Sequence to sequence. (b) Sequence to sequence plus attention. For both (a) and (b) we have studied various architectures for the encoder/decoder. (c) Neural network architectures based on different configurations of recurrent (LSTM), convolutional (CNN) and fully connected (FC) layers. (d) Classic machine learning (ML) models such as random forest or linear regression. And, (e) gaNet models of all types described in section 3.2, considering different architectures for the learning blocks.

To measure the forecasting results we have applied regression metrics to assess the prediction capacities of the different algorithms to forecast the 6 future values of a time-series considering its past 24 values. The time-series have been arranged in a dataset of 24 predictors and 6 outcomes. Each time series correspond to the sequential aggregation of traffic (in time-slots of 1-hour) for each of the 6214 mobile devices.

The forecasting metrics used have been mean square error (MSE), mean absolute error (MAE), median absolute error (MAD), coefficient of determination (R^2) and normalized root mean square deviation (NRMSD). The definition of these metrics is the following, considering Y as the ground-truth values, $\hat{Y}$ the predicted values and $\bar{Y}$ the mean value of Y [44]:

$$MSE = Mean\left(\left(Y - \hat{Y}\right)^2\right); \quad MAE = Mean\left(\left|Y - \hat{Y}\right|\right); \quad MAD = Median\left(\left|Y - \hat{Y}\right|\right)$$

$$NRMSD = \frac{\sqrt{Mean\left(\left(Y - \hat{Y}\right)^2\right)}}{range(Y)}$$

$$R^2 = 1 - \frac{\sum_{i=0}^{N}(Y_i - \hat{Y}_i)^2}{\sum_{i=0}^{N}(Y_i - \bar{Y})^2}$$

In Fig. 14 are presented all the performance metrics in three groups of columns. The first (average) gives the average value for all metrics. This average is performed along the 6 forecasted values. The second group (T0) provides the metrics for the nearest predicted value, and the last group (T5) provides the metrics for the last predicted value.

Taking all metrics into account, R^2 and MAD can be considered particularly important since they provide two different views of the results. R^2 is more sensitive to extreme prediction errors (due to the quadratic function) and MAD is more sensitive to good predictions around central values (mean/median). All metrics have values greater than zero with unbounded upper values, with the exception of R^2 that has an upper limit of 1 with unbounded lower limit. Fig. 15 and 16 provide a more detailed view of the results for the R^2 and MAD metrics.

R^2 provides an indication of the variance explained by the model. A value of 1 is its best result corresponding to a perfect fit to the data, a value of zero indicates that the prediction of the model is as good as that given when choosing the average value as the predicted value and a negative value (the smaller the worse) implies that our model is worse than choosing the mean as the predicted value. The other metrics (MSE, MAE, MAD and NRMSD) are error metrics. They are always positive by construction, with a value of zero corresponding to the best result.

The models in Fig. 14 are grouped by class (Section 3.2) with additional details in the column: 'Model'. The information provided for each model is the number and type of layers used: CNN, LSTM and Fully Connected (FC) layers. For gaNet models that are formed by repeating blocks, the configuration of the repeating block is included in parenthesis with and asterisk and a number to the right of the parenthesis that indicates the number of repetitions (m) of the block. Inside the parenthesis there is a list of layer names preceded with a number that corresponds to the number of repeating layers of that type. For example, a gaNet Type III- (4FC)*15 corresponds to a Type III gaNet model with 15 repeated blocks, each of them formed by 4 FC layers; a gaNet Type III- (2 LSTM + 4 FC)*5 corresponds to a Type III gaNet model with 5 repeated blocks, each consisting of 2 LSTM and 4 FC layers; and, a gaNet Type IV- (2 LSTM + 1 FC)*2 + (6 FC)*2 corresponds to a Type IV gaNet model with 2 repeated blocks, each of them formed by 2 LSTM and one FC layers, followed by another sequence of 2 repeated blocks, each consisting of 6 FC layers.

It is interesting that combining CNN and RNN (LSTM) networks does not improve the prediction results, contrary to the behavior observed in other forecasting problems related to classification [17,18], where adding a CNN as the first layer significatively improves the results. Moreover, for the Seq2Seq and Seq2Seq+Attention models, the results are worsened by the addition of a CNN to an LSTM network. In all cases, the CNN network used have been a 1D CNN that performs the convolution with a one-dimensional kernel.

In order to appreciate the excellent convergence behavior during training of the gaNet models, we can compare the training results of a NN with 180 hidden layers (Fig. 14) and the training results of a gaNet Type III model with m=30 and a building block formed by 6 FCs that add a total of 180 layers. We can see that the NN with 180 layers provides extremely bad results since it does not converge properly, instead the gaNet model with an identical number of layers converges perfectly providing satisfactory results.

The values in Fig. 14 are color-coded, where the greenest is better and the redder is worse (comparison of values is applied column-wise). We can observe that the models in the classes: Seq2Seq and Seq2Seq+Attention (Section 3.2) have obtained the best results considering the MAE and MAD metrics, while the models of class: gaNet type IV obtained the best results for the R^2, MSE and NRMSD.

Observing Fig. 14 it is easy to appreciate that the best results are concentrated on the following models: (a) gaNet III and IV models, (b) Random Forest and (c) Seq2seq/Seq2Seq+Attention models. The Seq2Seq+Att models with 2 LSTM layers provide the best results for MAE and MAD metrics but they require and excessive time for training and prediction (Fig. 17 and 18). The Random Forest algorithm needs a separate model per forecasted value. It provides good results for MAE and MAD metrics but not really good results for the other metrics. The gaNet models provide best results for MSE, NRMSD and R^2 metrics and do not require an excessive time for training and prediction, being in a good balanced position between prediction performance and required resources. Some of the models in Fig. 14 have not been transferred to Fig. 15 and 16, since they correspond to models that provide extremely bad results with values of R^2 and MAD that are too high compared to the rest of the models. Otherwise they would have altered the graphics due to the required change of scale. These models are: Seq2Seq/(2CNN + 1 LSTM + 1FC), Seq2Seq+Attention/2 CNN + 1 LSTM + 1 FC and NN/180 FC.

In addition to using traffic volumes as predictors, we have tried to incorporate the time of day and day of the week (of the time-slots) as additional predictors (exogenous variables). Interestingly, this addition did not significantly improve the prediction results. In order to incorporate these exogenous variables, and, depending on the model, a flattening of the input predictors has been required.

		Average					T0					T5				
Class	Model	MSE	MAE	MAD	R2	NRMSD (%)	MSE	MAE	MAD	R2	NRMSD (%)	MSE	MAE	MAD	R2	NRMSD (%)
Seq2Seq	1 LSTM + 1 FC	3.3123	0.9803	0.4142	0.6778	10.4112	2.5253	0.7043	0.0949	0.7571	9.0906	4.0646	1.1589	0.8074	0.6032	11.5331
	2 LSTM + 1 FC	3.1489	0.8242	0.1675	0.6937	10.1512	2.4283	0.6725	0.1009	0.7664	8.9143	3.9053	0.9493	0.1777	0.6188	11.3049
	2 CNN + 1 LSTM + 1 FC	9.5032	2.5103	2.2607	0.0756	17.6349	5.3560	1.7122	1.7061	0.4848	13.2391	13.4781	3.0542	2.6774	-0.3158	21.0016
Seq2Seq + Attention	1 LSTM + 1 FC	3.1978	0.8347	0.1437	0.6889	10.2298	2.4956	0.7077	0.1187	0.7599	9.0371	3.9019	0.9556	0.1538	0.6191	11.2999
	2 LSTM + 1 FC	3.2158	0.7906	0.1100	0.6872	10.2584	2.4451	0.6732	0.1074	0.7648	8.9451	4.0822	0.8836	0.1016	0.6015	11.5581
	2 CNN+ 1 LSTM + 1 FC	6.7897	1.8170	1.1733	0.3395	14.9061	4.9206	1.5427	1.2854	0.5266	12.6896	8.3136	1.9623	0.7630	0.1884	16.4942
NN Achitectures	1 LSTM + 1 FC	3.1328	0.8670	0.1876	0.6952	10.1253	2.6751	0.7431	0.1024	0.7427	9.3563	3.6661	0.9628	0.1876	0.6421	10.9532
	2 LSTM + 1 FC	3.0650	0.8471	0.2005	0.7018	10.0151	2.4869	0.6858	0.0831	0.7608	9.0213	3.6207	0.9560	0.2657	0.6465	10.8851
	2 CNN + 1 FC	3.0801	0.8559	0.2182	0.7004	10.0397	2.6056	0.7285	0.1249	0.7493	9.2340	3.5567	0.9356	0.2605	0.6528	10.7885
	2 CNN (MaxPooling) + 1 FC	3.0577	0.8765	0.2044	0.7026	10.0031	2.6040	0.7964	0.2068	0.7495	9.2311	3.5224	0.9339	0.2375	0.6561	10.7364
	2 CNN + 1 LSTM + 1 FC	3.0842	0.8487	0.2247	0.7000	10.0464	2.5907	0.7565	0.1470	0.7508	9.2077	3.6131	0.9351	0.2330	0.6473	10.8737
	2 CNN + 2 LSTM + 1 FC	3.0950	0.8647	0.2335	0.6989	10.0639	2.5871	0.7281	0.1370	0.7511	9.2012	3.6217	0.9763	0.2746	0.6464	10.8866
	1FC (linear regression)	3.1360	0.8833	0.2098	0.6949	10.1304	2.7268	0.7629	0.1356	0.7377	9.4464	3.5928	0.9826	0.2019	0.6493	10.8431
	6 FC	3.0686	0.8817	0.2243	0.7015	10.0209	2.6105	0.7926	0.2335	0.7489	9.2428	3.5386	0.9945	0.2578	0.6545	10.7610
	180 FC	10.3533	2.8675	2.9141	-0.0071	18.4068	10.4896	2.8870	2.9141	-0.0091	18.5275	10.3269	2.8645	2.9131	-0.0081	18.3833
Alternative ML	Random Forest	3.2577	0.7921	0.1327	0.6831	10.3251	2.5505	0.6349	0.0794	0.7546	9.1359	3.8106	0.8864	0.1934	0.6280	11.1669
	Linear Regression	3.1090	0.8684	0.1953	0.6976	10.0867	2.6712	0.7530	0.1070	0.7430	9.3495	3.5623	0.9359	0.1775	0.6522	10.7969
gaNet Type I	(1 FC)*10	3.1439	0.8831	0.2090	0.6942	10.1431	2.7135	0.7372	0.0914	0.7390	9.4233	3.5706	0.9601	0.2093	0.6514	10.8095
	(2 FC)*20	3.0771	0.8975	0.2890	0.7007	10.0349	2.5720	0.7749	0.2142	0.7526	9.1742	3.5938	0.9732	0.2266	0.6492	10.8447
gaNet Type II	(2 FC)*2	3.1009	0.8601	0.2140	0.6984	10.0735	2.6529	0.7391	0.1296	0.7448	9.3175	3.5637	0.9466	0.2140	0.6521	10.7991
	(3 FC)*2	3.0576	0.8422	0.1652	0.7026	10.0029	2.5561	0.7300	0.1453	0.7541	9.1458	3.5719	0.9143	0.1480	0.6513	10.8115
gaNet Type III	(3 FC)*3	3.0413	0.8543	0.2084	0.7042	9.9762	2.5302	0.6972	0.0959	0.7566	9.0994	3.5300	0.9214	0.1563	0.6554	10.7480
	(4 FC)*15	3.0401	0.8317	0.1553	0.7043	9.9743	2.5518	0.7094	0.1135	0.7545	9.1382	3.5552	0.9192	0.1905	0.6529	10.7863
	(6 FC)*15	3.0405	0.8462	0.1928	0.7042	9.9749	2.4636	0.6997	0.1136	0.7613	8.9789	3.5798	0.9455	0.1990	0.6562	10.8234
	(6 FC)*30	3.0915	0.8464	0.1750	0.6993	10.0582	2.5028	0.7041	0.1192	0.7592	9.0500	3.5873	0.9707	0.2094	0.6498	10.8349
	(1 LSTM + 1 FC)*10	3.0345	0.8104	0.1773	0.7048	9.9651	2.4809	0.6672	0.0878	0.7613	9.0103	3.6038	0.9053	0.2086	0.6482	10.8596
	(2 LSTM + 1 FC)*5	3.0058	0.8472	0.2283	0.7076	9.9179	2.4746	0.7224	0.1797	0.7620	8.9988	3.5672	0.9246	0.2192	0.6518	10.8045
	(2 LSTM + 1 FC)*10	2.9848	0.8192	0.2224	0.7096	9.8832	2.4241	0.6806	0.1200	0.7668	8.9066	3.5473	0.9370	0.2797	0.6537	10.7742
	(2 CNN (MaxPooling) + 1 FC)*3	3.0303	0.8554	0.2535	0.7052	9.9582	2.5253	0.7662	0.2480	0.7571	9.0906	3.5180	0.9413	0.3355	0.6566	10.7296
gaNet Type III-A	(3 FC)*2	3.0384	0.8303	0.2021	0.7044	9.9715	2.4988	0.7131	0.1334	0.7596	9.0427	3.5209	0.9076	0.2588	0.6563	10.7340
	(4 FC)*3	3.0428	0.8314	0.1754	0.7040	9.9787	2.5226	0.7063	0.1295	0.7573	9.0857	3.5314	0.8953	0.1708	0.6553	10.7500
	(2 LSTM + 1 FC)*5	3.0217	0.8002	0.1478	0.7061	9.9440	2.4754	0.6717	0.0895	0.7619	9.0004	3.6584	0.8957	0.2349	0.6429	10.9416
gaNet Type III-B	(3 FC)*3	3.0446	0.8274	0.1545	0.7038	9.9817	2.5379	0.7091	0.0886	0.7559	9.1133	3.5375	0.9370	0.1496	0.6547	10.7594
	(4 FC)*10	3.1062	0.8601	0.1736	0.6978	10.0820	2.5728	0.7060	0.0936	0.7525	9.1757	3.6019	0.9247	0.1687	0.6484	10.8569
	(2 LSTM + 1 FC)*10	3.0287	0.8556	0.2160	0.7054	9.9556	2.4546	0.7326	0.1470	0.7639	8.9625	3.6178	0.9338	0.2354	0.6468	10.8807
gaNet Type IV	(2 LSTM + 1 FC)*2 + (6 FC)*2	2.9828	0.8428	0.2474	0.7098	9.8799	2.3909	0.7208	0.1822	0.7700	8.8454	3.5221	0.9372	0.2508	0.6562	10.7359
	(2 LSTM + 1 FC)*3 + (6 FC)*3	3.0363	0.8276	0.2012	0.7046	9.9680	2.4747	0.6960	0.1031	0.7619	8.9991	3.5402	0.9322	0.1844	0.6544	10.7634
	(1 LSTM + 1 FC)*2 + (2 LSTM + 1 FC)*2 + (6 FC)*2	3.0048	0.8307	0.1818	0.7077	9.9161	2.4789	0.7026	0.1169	0.7615	9.0067	3.5351	0.9094	0.1341	0.6549	10.7557
gaNet Type IV-A	(2 LSTM + 1 FC)*2 + (6 FC)*2	3.0316	0.8213	0.2117	0.7051	9.9603	2.4474	0.6557	0.0906	0.7646	8.9494	3.5305	0.9286	0.2117	0.6553	10.7486

Fig 14. Table of metrics for performance metrics for all algorithms studied in this work.

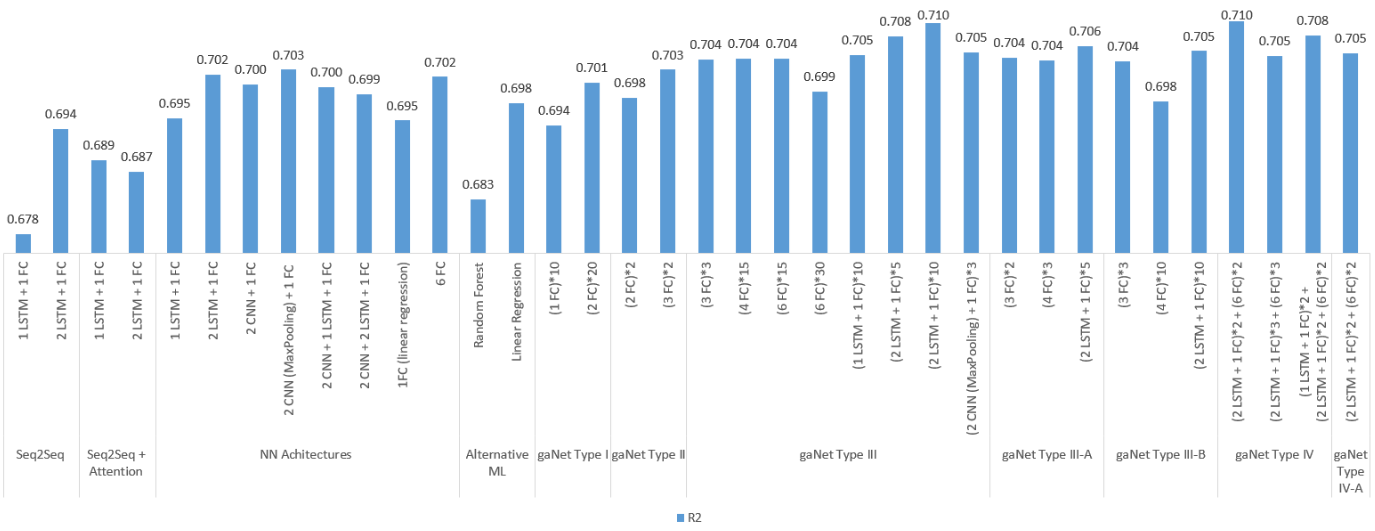


Fig 15. Comparison of R^2results for all algorithms.

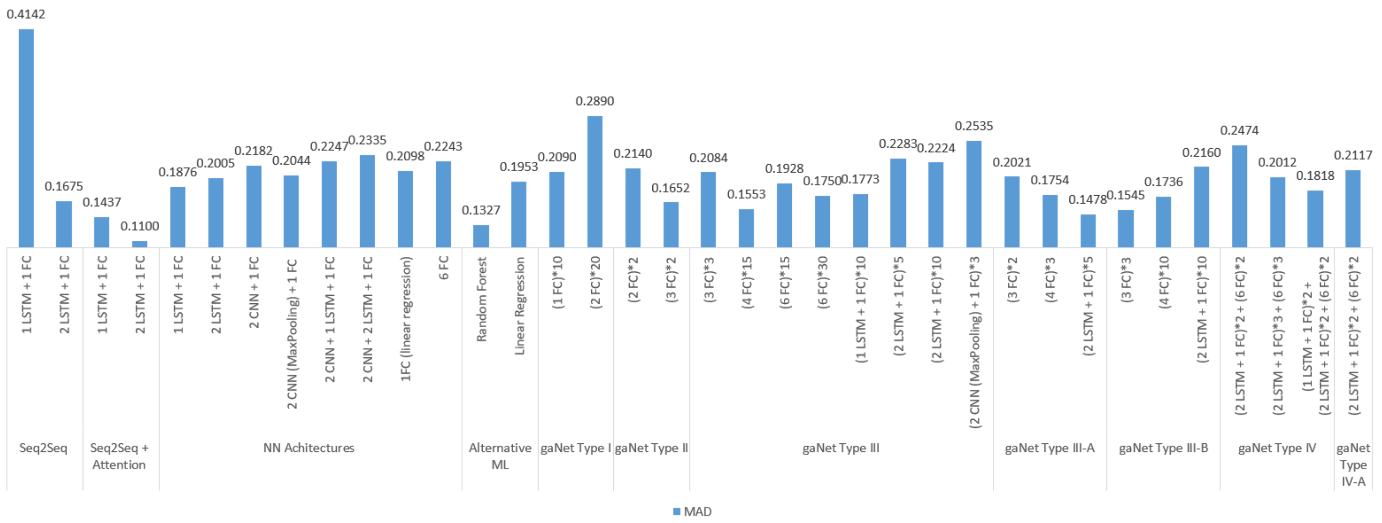


Fig 16. Comparison of MAD results for all algorithms

To check the significance of the results obtained, Table I presents the application of the Wilcoxon signed-rank test for the comparison of the prediction results, for each metric, between the best model for each of the performance metrics and the rest of the models. The alternative hypothesis used has been that the values for the error metrics produced by the best models are smaller than the rest, and, that they are greater than the rest for the R^2 metric. Considering that the p-values is much smaller than the significance threshold (0.01 at 1% significance level) we can conclude that the best values obtained are significantly better than the rest.

Observing Table I we conclude than the gaNet model: Type IV/(2 LSTM + 1 FC)*2 + (6 FC)*2 is the best model for the MSE, R^2 and NRMSD metrics and that the model Seq2Seq+Attention with 2 LSTM layers is the best model for MAE and MAD. That is, the gaNet model is the best model to reduce extreme values of errors (quadratic errors) and Seq2Seq is the best model to reduce more centered errors (absolute value errors). Focusing on the MAE and MAD errors, after Seq2Seq, the best model is Random Forest closely followed by an gaNet model: gaNet Type III-A/(2 LSTM + 1 FC)*5, which seems to indicate that the learning block: 2 LSTM + 1 FC is a good choice in almost any occasion.

We can conclude that gaNet provides better performance if we focus on data with a strong presence of outliers, in line with the properties of GB models.

Metric	*p-value*	*Significance Level (1%)*	*Best model*	*Mean value - rest of models*	*Best model class/name*	
MSE	6.675E-15	Yes	2.9828	3.5280	gaNet Type IV	(2 LSTM + 1 FC)*2 + (6 FC)*2
MAE	8.900E-16	Yes	0.7906	0.9675	Seq2Seq + Attention	2 LSTM + 1 FC
MAD	8.904E-16	Yes	0.1100	0.3491	Seq2Seq + Attention	2 LSTM + 1 FC
R2	8.896E-16	Yes	0.7098	0.6568	gaNet Type IV	(2 LSTM + 1 FC)*2 + (6 FC)*2
NRMSD	8.904E-16	Yes	9.8799	10.5758	gaNet Type IV	(2 LSTM + 1 FC)*2 + (6 FC)*2

Table I. Wilcoxon signed-rank test: significance of results for best model vs. performance metrics

For the type of problem for which we have applied gaNet (forecast of continuous traffic volumes per device for data network applications), it is not only important to achieve good prediction results, but also to require small training and prediction times. Linear regression (using either NN or other ML methods) requires the smallest times but does not achieve good prediction results. Random forest also requires very small times for training and prediction, but it is necessary to train a model for each output value. Type III and IV gaNet models require small training times and even smaller prediction times when compared to other models of similar prediction performance, mainly when compared to Seq2Seq and Seq2Seq+Attention. Fig. 17 and 18 provide details on the training and prediction (test) times required for all algorithms considering the training and the test data sets described in Section 3.1.

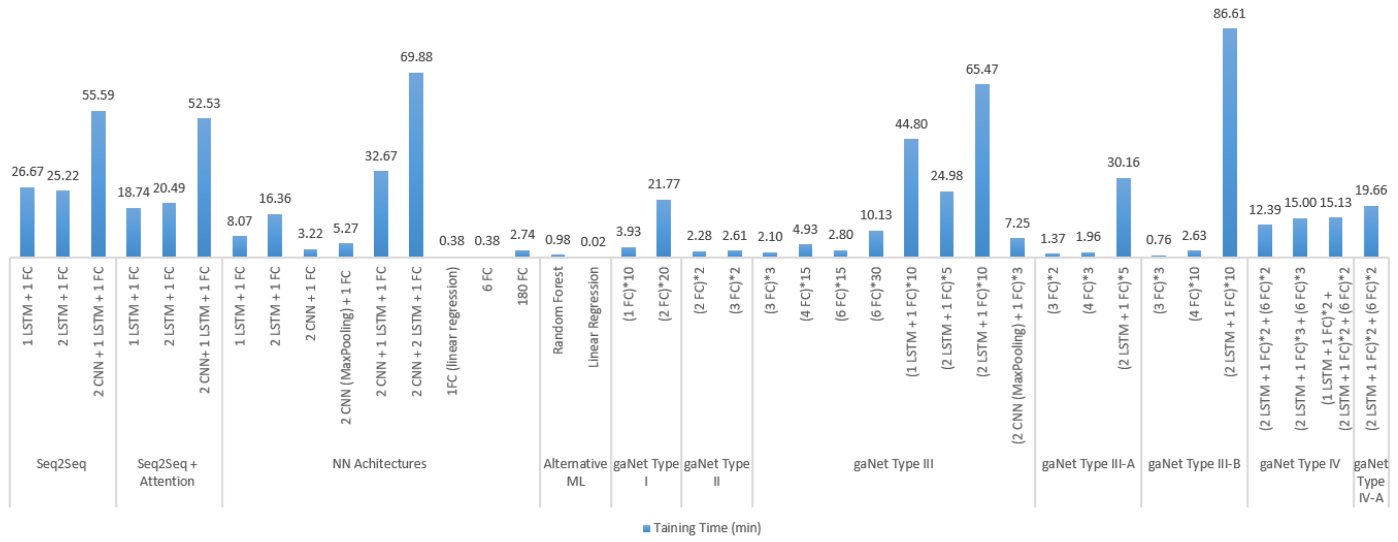


Fig 17. Training times for all algorithms

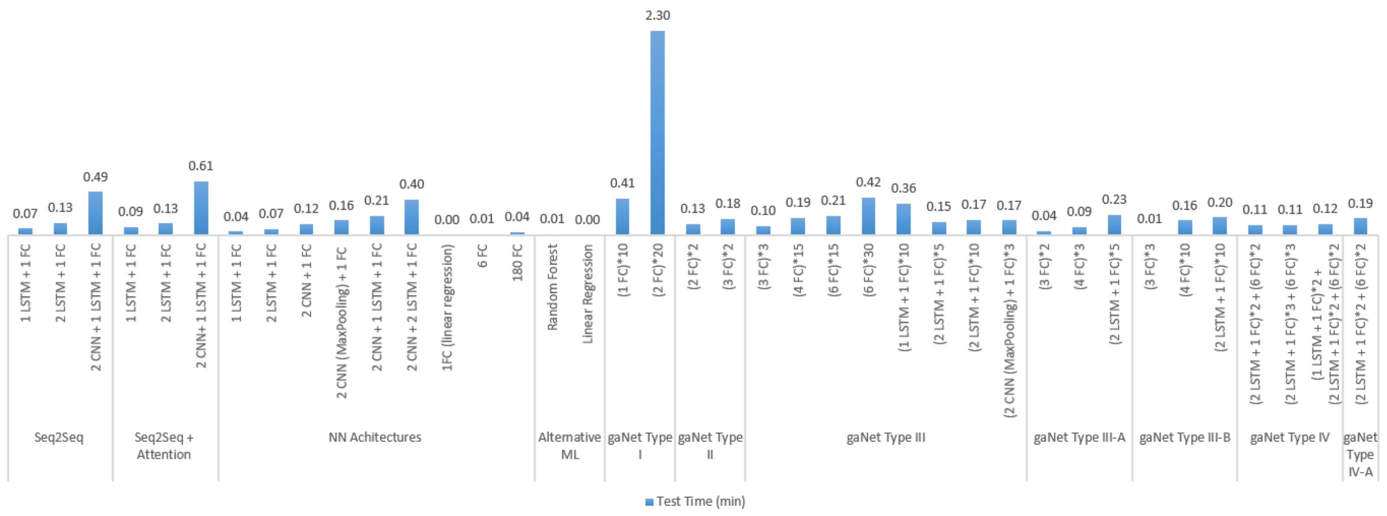


Fig 18. Prediction times for all algorithms

Considering the deep learning nature of the models proposed for this work, it is important to analyze the results of these models from the point of view of computational complexity. One way to measure complexity is through the analysis of training and prediction times (Fig 18 and 19); however, this does not provide a complete picture of computational complexity. Therefore, to better understand this important aspect, we have also considered the number of parameters (trained parameters) of each model and its relation to the training and prediction times. The number of parameters per model is provided in Fig. 19 and the relationship between number of parameters and training and prediction times is given in Fig. 20. This integrated view on computational complexity is necessary, since deep learning models impose challenges [45] to measure their complexity, since it is based not only on the number of parameters of the models but also on the choice of the processor (e.g. CPU, GPU, ..), parallelism and memory issues, and in the number of training epochs required to reach a certain state of convergence. Taking into account these difficulties, Fig 19 and 20 try to offer a practical view of the complexity of the different models.

Each dot in Fig 20 corresponds to a model, but only the best models (Table I) and the outliers (extreme values) are identified. Regarding the outliers, it is clear that extreme cases with 180 layers (NN/180 FC and gaNet Type III/(6 FC)*30) will appear as outliers. It is interesting that the second best gaNet model (gaNet Type III/(2 LSTM + 1 FC)*10) presents a high training time, but its prediction time is very limited. The reason for the extreme training time for gaNet Type III-B/(2 LSTM + 1 FC)*10 is due to require more epochs to converge than other models, and for NN/(2 CNN + 2 LSTM + 1 FC) the reason is different, and is due to the very long time required for each backpropagation cycle (even with a GPU). Regarding the best models: Seq2Seq+Att/(2 LSTM + 1 FC) and gaNet Type IV/(2 LSTM + 1 FC)*2 + (6 FC)*2, both have comparatively small training and prediction times; smaller for the gaNet model, which is interesting considering the much larger number of parameters needed for the gaNet model.

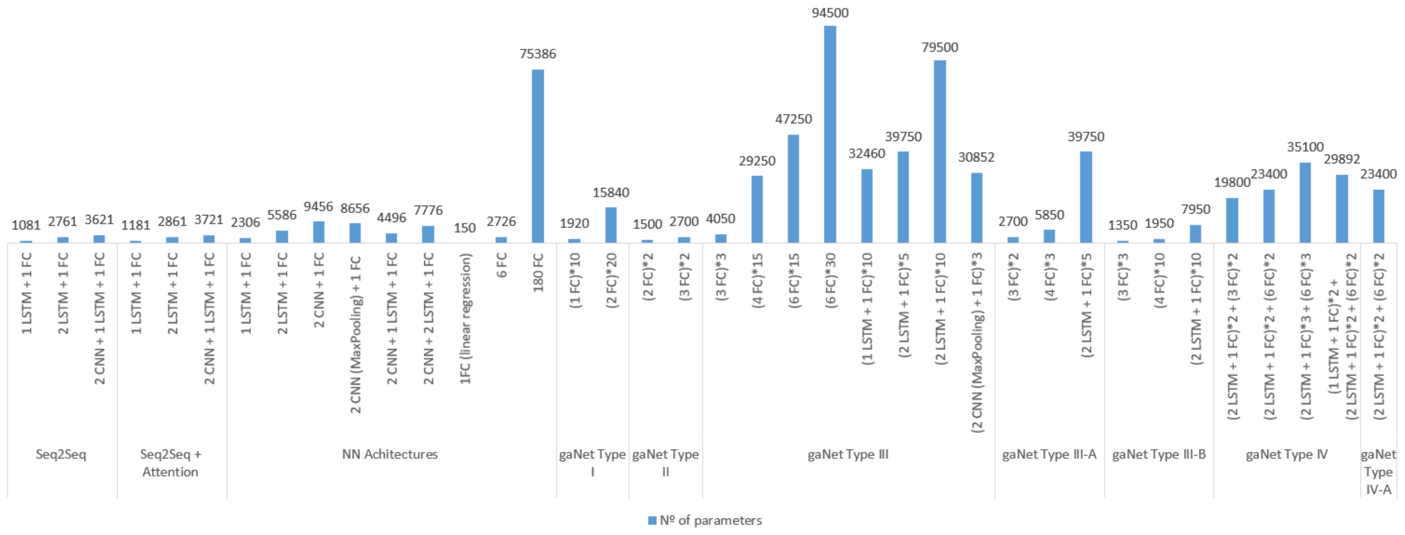
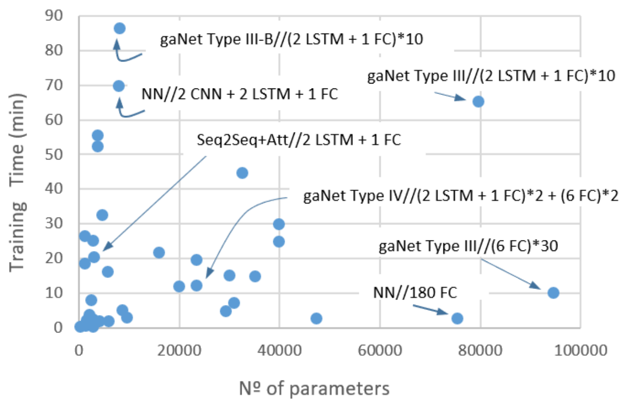


Fig 19. Number of parameters used by the different models



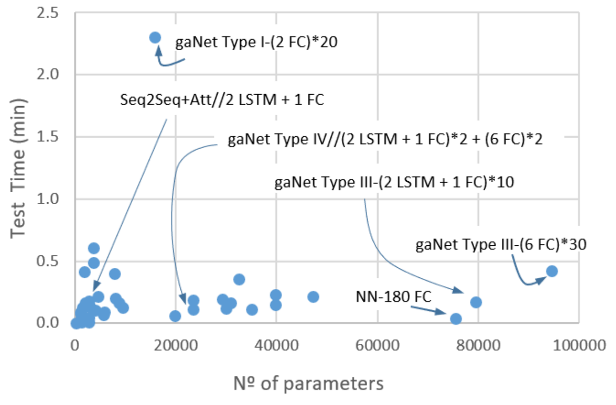


Fig 20. Scatterplot for training times (left) and test (right) vs. number of parameters of each model.

The main difficulty found with the gaNet models have been the large number of layer combinations available when choosing a particular architecture, implying a large hyper-parameter search space. This is particularly clear with the Type IV variant of the gaNet model. Nevertheless, the good results of gaNet Type IV indicate that diversity in the composition of the blocks is the best option for the additive structure of gaNet. An explanation to this interesting result can be articulated considering that each block, having a different structure, is capable of capturing different patterns of the data, in a similar way to other ensemble models.

gaNet is a deep learning architecture with a possible large number of hidden layers. Some of the models whose results are shown in Fig. 14 have from several tens of layers, such as: gaNet Type III-B – (2 LSTM + 1 FC)*10 with 30 layers, up to an extreme case of 180 layers: gaNet Type III – (6 FC)*30. gaNet handles the difficulties incurred by deep learning architectures by using various techniques to facilitate the convergence of the algorithm during training: (a) ReLU layers to address the vanishing gradient problem, (b) batch normalization layers, used in conjunction with CNN layers, to achieve faster convergence, and (c) the residual network properties obtained by a stagewise additive model.

Considering the gaNet models, it is interesting that to avoid overfitting we did not need to apply specific regularization techniques, such as: dropout or L2/L1 regularization, which were used for some of the Seq2Seq and NN models. This fact indicates that gaNet has an intrinsic regularization property

The architecture of gaNet, which is based on a lego-like assembly of similar structures, is appropriate for the distribution and parallelism provided by modern decentralized platforms designed to handle the processing of deep learning networks (e.g. Tensorflow) [41,42,43]. This aspect positions gaNet as an option in networking solutions with demanding processing requirements (e.g. large data volumes or real-time response).

In order to guarantee the generalization of results, we have applied all the models of this work to the test set described in section 3.1. This test set corresponds to a randomly selected 20% of the total number of the available time-series. In addition, considering the large number of training cycles required (backpropagation cycles), we have applied a further validation approach by randomly separating another 20% of the remaining training data to form a validation set. This validation set is used to obtain validation metrics for each backpropagation cycle. The composition of the validation set is changed in each epoch. An epoch is defined as a complete training cycle on all the training data. An epoch may correspond to a large number of backpropagation cycles, depending on the size of the batches used for each cycle. To carry out the experiments of this work, we have used between 20 to 100 epochs with a batch size of 100 samples (individual time-series). The number of training epochs is not a fixed number but based on an early-stopping criterion if the validation error does not improve in 10 cycles.

The use of a validation set is necessary to avoid overfitting, since during the training of the models we store the weights associated with the best performance metrics obtained with the validation set. If we were using the training set to obtain the best weights, we would be fitting the model to the training set, avoiding the generalization of results.

In summary, the final distribution for the test, validation and training data corresponds to 20%, 16% and 64% of the total data, respectively. The test set remains fixed for all models, while the distribution of samples for the validation and training sets changes randomly for each training epoch.

Fig. 21 shows the evolution of the prediction error: MSE, for the training and validation sets during the training of several gaNet architectures. We can observe how the validation error reaches its best value quite rapidly: after 5 to 10 epochs, and that even after the first epoch the model starts to convergence.

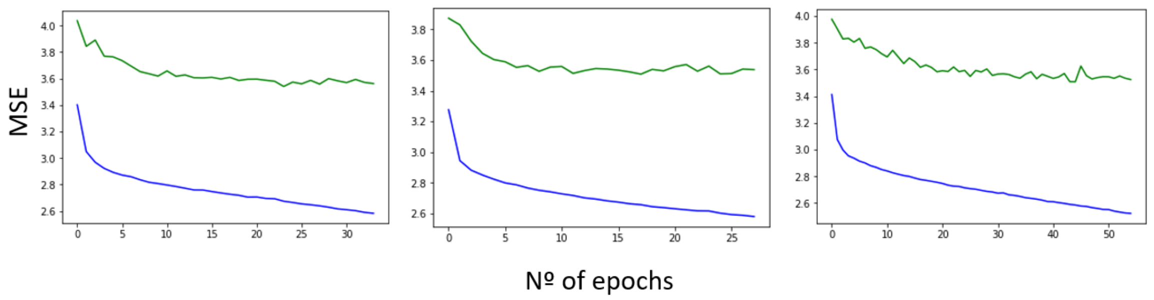


Fig 21. Evolution of MSE for the training (blue) and validation (green) sets during training for various gaNet architectures

It is interesting to analyze the different results of the algorithms. In particular, the good results for the MAE and MAD metrics for the Random Forest and Seq2Seq+Attention models, and the good results for MSE and R^2 for gaNet. Random Forest is an ensemble model based on bagging that justifies its best performance with MAD, which is more sensitive to good predictions around central values. Meanwhile, R^2 is more sensitive to extreme prediction errors, and it is in this case that gaNet outperforms the rest; the reason for that can be understood by applying gradient boosting. Gradient boosting increases the size of the model by adding new learning blocks whose objective is to reduce the remaining errors, focusing on the larger ones and, consequently, improving the metrics related to the quadratic errors (e.g. MSE). With respect to the Seq2Seq+Att models, it is more challenging to explain their good behavior for metrics related with errors around central values (median/mean). In this case, the attention mechanism seems to provide a similar averaging effect in its results to the ensemble models.

Focusing on the different results of gaNet architectures, we can conclude that Type I and Type II do not provide good results in terms of training/prediction times or prediction performance, this is due to having several optimization cycles per iteration without end-to-end training. Types III and IV provide good results, due to the incorporation of end-to-end training and the variety of block architectures, in the case of Type IV models. Type III models generally have better training times with comparatively similar prediction times. As expected, the weight sharing of Type III-B models gives them the best training and prediction times. It is also interesting to mention that Type III-A (with a sum of loss functions over all intermediate errors) provides the best MAD results for all gaNet models but does not provide the best results in R^2. In this case, the sum of loss functions for Type III-A seems to provide an averaging effect similar to that presented above for the cases of Random Forest and Seq2Seq+Attention.

We have implemented all the ML models in python using the scikit-learn package [46] and for all other models we have used Tensorflow/Keras [43].

5. CONCLUSION

This work presents a novel NN architecture (called gaNet) for multi-output regression, based on an additive model of independent building blocks made of NN layers in a lego-like configuration. The complete architecture is trained end-to-end and can be deployed to high-performance deep learning platforms.

gaNet is based on ideas from gradient boosting models and has connections with residual networks and stacked models, but it is different from any of them and still shares many interesting properties with these models. The architecture resembles a residual network with different network topology and connections, and shares with residual networks its robust convergence behavior during training even with an extreme number of layers added to the network. It has also a similar configuration to a stacked model. The regularization effect observed in stacked models is also observed in gaNet models.

The architecture has been applied to the real and difficult problem of k-ahead forecasting of traffic volumes using a real dataset from an IoT mobile network. A detailed comparison of results between gaNet and an extensive selection of alternative modern ML algorithms demonstrate that gaNet is an excellent alternative to state-of-the-art solutions, such as sequence to sequence models and attention models. gaNet requires less resources than these alternative solutions and provides best results for all metrics based on quadratic errors. This implies a better performance for data highly unnormalized and with a strong presence of outliers, in line with the properties of GB models.

As future lines of research it would be interesting to apply gaNet to classification problems and to study the necessary modifications and results. Considering the use of CNN models in the architectures presented in this work, it is appropriate to mention the current research works in transfer learning applied to time-series classification and forecasting [47, 48], which, in line with the good results in the image classification field [49, 50] could be a promising area to investigate in conjunction with the models presented here. An additional research area would be to analyze the impact of using alternative activation functions in the network. Finally, since gaNet has a generic nature, it might be useful to study its applicability to other areas, such as language and image processing, and to experiment with additional network configurations such as those based on DenseNet [50, 51].

References

[1] Wang M. et al., "Machine Learning for Networking: Workflow, Advances and Opportunities," in IEEE Network, vol. 32, no. 2, pp. 92-99, March-April 2018.

[2] Boutaba R. et al. “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities”, Journal of Internet Services and Applications (2018) 9:16, https://doi.org/10.1186/s13174-018-0087-2

[3] Mahdavinejad M.S. et al., “Machine learning for Internet of Things data analysis: A survey”, Digital Communications and Networks, vol. 4 (3), pp. 161-175. 2018. https://doi.org/10.1016/j.dcan.2017.10.002

[4] Shu Y. et al., "Wireless traffic modeling and prediction using seasonal ARIMA models," IEEE International Conference on Communications, 2003. pp. 1675-1679, vol.3. doi: 10.1109/ICC.2003.1203886

[5] Zhani M. F. and Elbiaze H. “Analysis and Prediction of Real Network Traffic”. Journal of Networks, vol. 4, no. 9, 2009

[6] Feng H. and Shu Y, "Study on network traffic prediction techniques," Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005., pp. 1041-1044. doi: 10.1109/WCNM.2005.1544219

[7] Huang C., Chiang C. and Li Q., "A study of deep learning networks on mobile traffic forecasting," 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), pp. 1-6. doi: 10.1109/PIMRC.2017.8292737

[8] Xie M. et al, “A seasonal ARIMA model with exogenous variables for Elspot electricity prices in Sweden”. 2013 10th International Conference on the European Energy Market (EEM), Stockholm (May 2013), Pages 1–4

[9] Zhou Z-H. (2012), “Ensemble Methods: Foundations and Algorithms”, Chapman & Hall/CRC. pp. 83-86.

[10] Friedman, J. H. (2001). “Greedy Function Approximation: A Gradient Boosting Machine”. The Annals of Statistics, 29(5), pp. 1189-1232. http://www.jstor.org/stable/2699986

[11] Hastie, T., Tibshirani, R. and Friedman, J. H. (2009). “The Elements of Statistical Learning (2nd ed.)”. New York: Springer. pp. 337–384. http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[12] Borchani H., Varando G., Bielza C. and Larrañaga P. 2015. “A survey on multi-output regression. Data Mining and Knowledge Discovery”. 5, 5 pp. 216-233. http://dx.doi.org/10.1002/widm.1157

[13] He K., Zhang X., Ren S. and Sun J., "Deep Residual Learning for Image Recognition". Computer Vision and Pattern Recognition, IEEE, pp. 770-778, 2016.

[14] David H. Wolpert D. H., “Stacked generalization”. Neural Networks. vol. 5 (2). 1992. pp. 241-259. https://doi.org/10.1016/S0893-6080(05)80023-1

[15] Lopez-Martin M., Carro B., and Sanchez-Esguevillas A., “Review of Methods to Predict Connectivity of IoT Wireless Devices”. Ad Hoc & Sensor Wireless Networks, vol. 38, no. 1-4, pp. 125-141. 2017.

[16] Wang A.X. et al. “Deep Transfer Learning for Crop Yield Prediction with Remote Sensing Data”. In Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS '18). ACM, New York, NY, USA, Article 50, 5 pages. 2018. https://doi.org/10.1145/3209811.3212707

[17] Lopez-Martin M et al., “Network traffic classifier with convolutional and recurrent neural networks for Internet of Things”, IEEE Access, vol. 5, pp. 18042-18050, 2017. doi: https://doi.org/10.1109/ACCESS.2017.2747560

[18] Lopez-Martin M., Carro B., Lloret J., Egea S. and Sanchez-Esguevillas A., "Deep Learning Model for Multimedia Quality of Experience Prediction Based on Network Flow Packets," in IEEE Communications Magazine, vol. 56, no. 9, pp. 110-117, Sept. 2018. https://doi.org/10.1109/MCOM.2018.1701156

[19] Sutskever I., Vinyals O and Le Q.V. “Sequence to Sequence Learning with Neural Networks”. 2014. arXiv:1409.3215 [cs.CL]

[20] Bahdanau D., Cho K. and Bengio Y. “Neural Machine Translation by Jointly Learning to Align and Translate”. 2014. arXiv:1409.0473 [cs.CL].

[21] Luong M-T, Pham H. and Manning C.D. “Effective Approaches to Attention-based Neural Machine Translation”. 2015. arXiv:1508.04025 [cs.CL].

[22] Fadlullah Z. M. et al., "State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow’s Intelligent Network Traffic Control Systems," in IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2432-2455. 2017. doi: 10.1109/COMST.2017.2707140

[23] Cenggoro T.W. and Siahaan I., "Dynamic bandwidth management based on traffic prediction using Deep Long Short Term Memory," 2016 2nd International Conference on Science in Information Technology (ICSITech), Balikpapan, 2016, pp. 318-323. doi: 10.1109/ICSITech.2016.7852655

[24] Lv J., Li X., Ran C. and He T., "Network traffic prediction and fault detection based on adaptive linear model," 2004 IEEE International Conference on Industrial Technology, 2004. IEEE ICIT '04., Tunisia, 2004, pp. 880-885 Vol. 2. doi: 10.1109/ICIT.2004.1490191

[25] Schwenk H. and Bengio Y., “Boosting Neural Networks”. Neural Computation. 12(8) (August 2000), pp.1869-1887. 2000. http://dx.doi.org/10.1162/089976600300015178.

[26] Moghimi M. et al., “Boosted Convolutional Neural Networks”. Proceedings of the British Machine Vision Conference (BMVC), pp. 24.1-24.13. BMVA Press. 2016.

[27] Ponomareva N. et al., “TF Boosted Trees: A scalable TensorFlow based framework for gradient boosting”. arXiv:1710.11555v1 [stat.ML]. 2017

[28] Dong M. et al., “GrCAN: Gradient Boost Convolutional Autoencoder with Neural Decision Forest”. arXiv:1806.08079v2 [cs.LG]. 2018

[29] Huang F. et al. “Learning Deep ResNet Blocks Sequentially using Boosting Theory”. arXiv:1706.04964 [cs.LG]. 2017

[30] Nitanda A. and Suzuki T., “Functional Gradient Boosting based on Residual Network Perception”. arXiv: 1802.09031v2 [stat.ML]. 2018

[31] Zhang C. et al., “Citywide Cellular Traffic Prediction Based on Densely Connected Convolutional Neural Networks”. IEEE Communications Letters, vol. 22, no. 8, pp. 1656-1659, 2018. https://doi.org/10.1109/LCOMM.2018.2841832

[32] Hua Y., “Traffic Prediction Based on Random Connectivity in Deep Learning with Long Short-Term Memory”. arXiv:1711.02833v2 [cs.NI]. 2018

[33] Fang W, Lu Z, Wu J and Cao Z. “RPPS: a novel resource prediction and provisioning scheme in cloud data Center” In: Services Computing (SCC), 2012 IEEE Ninth International Conference on. IEEE; 2012. p. 609-616.

[34] Zhou B, He D, Sun Z and Ng W.H. “Network traffic modeling and prediction with ARIMA/GARCH”. In: Proc. of HET-NETs Conference; 2005. p. 1-10.

[35] Sang A. and Li S., "A predictability analysis of network traffic," Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Tel Aviv, Israel, 2000, pp. 342-351 vol.1. doi: 10.1109/INFCOM.2000.832204

[36] Chen Z., Wen J. and Geng Y., "Predicting future traffic using Hidden Markov Models," 2016 IEEE 24th International Conference on Network Protocols (ICNP), Singapore, 2016, pp. 1-6. doi: 10.1109/ICNP.2016.7785328

[37] Mozo A, Ordozgoiti B, Gómez-Canaval S “Forecasting short-term data center network traffic load with convolutional neural networks”. PLOS ONE 13(2): e0191939. 2018. https://doi.org/10.1371/journal.pone.0191939

[38] Che Z., Purushotham S., Cho K., Sontag D. and Liu Y. “Recurrent neural networks for multivariate time series with missing values”. arXiv:1606.01865, 2016.

[39] Karim F., Majumdar S., Darabi H., and Chen S. “Lstm fully convolutional networks for time series classification”. arXiv:1709.05206, 2017.

[40] Qin Y., Song D., Cheng H., Cheng W., Jiang G., and Cottrell G. “A dual-stage attention-based recurrent neural network for time series prediction”. 2017. arXiv:1704.02971.

[41] Verma S. et al., "A Survey on Network Methodologies for Real-Time Analytics of Massive IoT Data and Open Research Issues," in IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1457-1477, thirdquarter 2017. doi: 10.1109/COMST.2017.2694469

[42] Izal M, Morató D, Magaña E, García-Jiménez S. “Computation of Traffic Time Series for Large Populations of IoT Devices”. Sensors (Basel). 2018;19(1):78. Published 2018 Dec 26. doi:10.3390/s19010078

[43] Abadi M. et al., “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems”. 2016. arXiv:1603.04467v2 [cs.DC]

[44] Hyndman R.J and Koehler A.B. “Another look at measures of forecast accuracy”. International Journal of Forecasting. Vol 22, Issue 4, pp. 679-688, 2006. https://doi.org/10.1016/j.ijforecast.2006.03.001

[45] Justus D. et al., “Predicting the Computational Cost of Deep Learning Models”. arXiv:1811.11880v1 [cs.LG], Nov 2018

[46] Pedregosa F. et al., “Scikit-learn: Machine Learning in Python”, Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[47] Ye R. and Dai Q., “A novel transfer learning framework for time series forecasting”. Knowledge-Based Systems. Vol. 156, pp. 74-99. 2018. https://doi.org/10.1016/j.knosys.2018.05.021.

[48] Fawaz H.I et al., "Transfer learning for time series classification," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 1367-1376. https://doi.org/10.1109/BigData.2018.8621990

[49] Wang S.H, Xie S, Chen X, Guttery D.S, Tang C, Sun J, Zhang Y.D. “Alcoholism Identification Based on an AlexNet Transfer Learning Model”. Frontiers in psychiatry vol. 10 205. 11 Apr. 2019, doi:10.3389/fpsyt.2019.00205

[50] Wang S.H, Tang C, Sun J, Zhang Y.D. “Cerebral micro-bleeding detection based on Densely connected neural network”. Frontiers in Neuroscience. 2019, doi: 10.3389/fnins.2019.00422

[51] Huang G. et al., “Densely Connected Convolutional Networks”. arXiv:1608.06993v5 [cs.CV] 28 Jan 2018