Future Generation Computer Systems. Volume 105, 2020, Pages 331-345. https://doi.org/10.1016/j.future.2019.12.013

IoT type-of-traffic forecasting method based on gradient boosting neural networks

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

Abstract— Network traffic classification is an important task for any current data network. There any many possible classification targets for the traffic, but we have considered as especially important the activity state of a connection and the identification of elephant flows (few connections carrying most of the traffic). With these detection targets, this work presents a modification of the gaNet architecture for classification. gaNet is an additive network model formed by 'learning blocks' that are stacked iteratively following the principles of boosting models. The original gaNet model is intended for regression, being the purpose of this work to show that it can be extended to classification under several adaptations. The resulting architecture is a generic additive network applicable to any supervised classification problem (gaNet-C).

To obtain experimental results, the model is applied to a type-of-traffic forecast problem using real IoT traffic from a mobile operator. The paper presents a comprehensive comparison of results between the proposed new model and many alternative algorithms in terms of classification and performance metrics. The proposed classifier can perform a k-step ahead detection forecast based exclusively on a limited time-series of previous values for each network connection. The results include two very different challenges: detection forecast of active connections and elephant flows; showing that, in both cases, the proposed algorithm presents state of the art results.

Index Terms— neural network; gradient boosting; traffic classification; detection forecast;

* 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

In this work we provide an extension of the gaNet architecture [1] to perform classification on supervised categorical outputs. We call this extension as gaNet-C. gaNet was originally intended to perform prediction on continuous output data (regression) based on a sequence of additive blocks (learning blocks), with all blocks sharing the same input features and trained end-to-end with stochastic gradient descent (SGD). gaNet provides excellent regression performance and can be applied to hard prediction problems, such as k-steps ahead forecasting of network-traffic volumes based on a sequence of previous ones [1].

This work shows that the new extension of gaNet for classification (gaNet-C) can be applied in similar terms to accomplish k-steps ahead forecast of two important states of a data network connection: the state of activity vs non-activity of the network connection, and the state of a network connection being an elephant flow. In both cases, we start from traffic volumes that have been aggregated into sequential and discrete time periods. The aggregation interval has to be large enough to avoid unnecessary noisy fluctuations and small enough to provide a practical granularity for prediction. Similar to [1] and for the same reasons, in this work we have chosen an aggregation time period of 1-hour. Once the aggregation period is established, the active state for a particular connection in a certain period consist in determining if there has been any traffic on it, otherwise we assume the state as non-active. Likewise, the elephant state for a particular connection in a certain period dictates if the aggregated traffic in that period belongs to the 10% quantile of connections with the highest traffic, which are known as elephant flows.

Predicting whether a network flow, in a certain future period, will be active and/or associated with an elephant flow is of great practical importance for the operations and management of current networks, for example, in admission control and resource management, which focus on the best allocation of shared and limited resources, it is extremely important to predict which connections will be active to optimize the allocation of scarce resources [2,3]. Likewise, for the routing and classification of traffic,

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).

based on the best allocation of routes to network connections, it is equally important to predict whether a future flow will be an outlier in terms of an extremely high volume of traffic i.e. elephant flow [2,4,5,6,7]. It is interesting the importance of detecting elephant flows, for which there is a growing need due to the current nature of traffic flows [2]. It is also worth noting that this work, which provides a model suitable for k-steps ahead forecasting and traffic classification, is unique, as far as we know, in the literature for this field [8,2].

The original gaNet architecture [1] is formed by the aggregation (addition) of the outputs of a series of 'building blocks', each of them integrated by several neural network (NN) layers. This architecture tries to re-implement original ideas from gradient boosting models, such as XGBoost [9] or LightGBM [10], but using these 'building blocks' instead of 'decision stumps' (trees with a single split). An end-to-end training of the complete model (based on SGD) is proposed for gaNet, rather than a separate optimization each time a new block is added to the model. The original work [1] presents several variants of the algorithm, each of them following an evolutionary process from an architecture closer to gradient boosting ideas to others more connected to residual networks [11] and stacked models [12]. It can be said that these latter variants are similar to residual networks where all the short-cuts (layer jumps) share the same input (the initial features).

In this work we have considered all the gaNet-C variants and chose only two of them, which have proven to be the fastest and with the best performance. The two variants chosen are the closest to a residual network.

To accommodate the gaNet architecture, which was initially intended for regression, to perform classification, we propose several adaptations of the original model, such as: using tanh (hyperbolic tangent) layers as the final layer of each 'building block', adding a sigmoid fully connected layer (FC) prior to any network output and applying a log-loss instead of a quadratic-loss as the cost function. All these extensions and modifications are presented in detail in section 3.2.

To score the performance of gaNet-C against other ML models, we have evaluated the most appropriate alternative models for classification and forecasting, such as: classic models as logistic regression and random forest, FC networks with different numbers of layers, recurrent and convolutional neural networks, and combinations of them [13,14], and, finally, seq2seq models [15] and seq2seq with attention models [16,17]. To select alternative models to compare results, we have placed special emphasis on choosing other current deep learning models that are suitable for sequence prediction, such as recurrent and seq2seq models. For all models, we have based the comparison between them on the usual metrics for classification: accuracy, F1-score, precision and score. Another point of interest of the study has been the necessary computational resources in terms of the training and prediction times required by the different models. Finally, considering the type-of-traffic forecasting for k-steps ahead that is applied to all models, we present how the classification performance is affected by the proximity in prediction time, showing that the evolution is not generally a linear decrease in performance with the temporal prediction distance, but a more complex (and unexpected) evolution curve.

The contributions of this work are the following: (1) Expand the gaNet architecture to classification, analyzing the possible extension alternatives. (2) Apply the new classifier (gaNet-C) to two difficult problems of type-of-traffic forecasting: active and elephant connections, each of which presents different challenges. (4) Offer a model that can be deployed in current high-performance platforms (e.g. Tensorflow). (5) Present a comprehensive comparison of gaNet-C with alternative ML models. (6) Introduce a model that offers excellent prediction results with prediction times almost similar to much simpler classic ML models.

The organization of the paper is: Section 2. examines related works. Section 3 presents the work in detail. Section 4 analyzes the results and, Section 5 provides discussion and conclusions.

2. RELATED WORKS

Network traffic classification when applied to the forecast of discrete-valued time-series presents more difficulties than its continuous counterparts that forecasts real-valued time-series, for which there is a well-established set of algorithms and tools based on: (1) statistical analysis methods, such as Auto Regressive Moving Average (ARMA), Auto Regressive Integrated Moving Average (ARIMA)...[18] or, (2) Machine Learning (ML) methods, such as Random Forest, Neural Networks, ... [18,19].

The statistical techniques applied to categorical time-series are based on assuming an intermediate latent-space of real-valued random variables which are connected by a link function (e.g. sigmoid...) to the expected probability of each of the categories that can take the forecasted value. The latent space can be modeled with: a Markov model, a discrete ARMA (DARMA, NDARMA) model [20] or a generalized linear model [21]. The resulting models are theoretically sound, but they require a fair amount of observed data, they are based on many assumptions on the probability distributions of data and their implementation is

computationally demanding and complex. An excellent and updated review of categorical time-series models is presented in [21]. For the specific case of binary-valued time-series, [22] provides a solution to daily wet/dry state sequence forecasting using a Discrete ARMA (DARMA) model; it needs to ensure the stationarity of the process which requires separating the days into seasons assuming stationarity within the seasons. In [23] it is assumed that the binary data generation process is formed by an underlying random continuous-valued state-space which generates the discrete process by a response function (e.g. logistic, probit). Likewise, [24] presents a similar model where the binary times-series can exploit time-dependent external covariates in a generalized linear model framework. A solution based on an underlying Markov model is presented in [25] for an economic/financial problem. For the more general case of discrete/categorical time-series, [21] offers a review of the different techniques available, with the work in [26] providing a detailed analysis of different alternatives for the application of generalized linear models for modeling and forecasting of categorical time-series.

Network traffic classification, whether within a time-series forecasting problem or within a time-independent prediction problem, can be considered as part of the more general problem of network traffic analysis and prediction (NTAP). In the former case i.e. time-series forecasting framework, the prediction is based on past categorical values (categorical time-series) and in the latter case i.e. time-independent prediction framework, the prediction relies on covariates which are assumed to have a correlation or dependence relationship with the predicted values e.g. information contained in the flow of packet headers [27]. There are general surveys of machine learning applied to NTAP, which encompasses not only traffic classification and prediction, but also traffic routing, congestion control, resource management, fault management, network security and quality of service and experience (QoS, QoE) management [28,2,29]. There are also reviews presenting different approaches for type-of-traffic classification applied to time-independent prediction problems [30,31], presenting a taxonomy of the different approaches that can be taken to obtain the information to assign a type of traffic at the packet, flow or connection level. Similarly, [21] is a recent and unique review work of network traffic classification methods within a time-series forecasting framework i.e. classifying future traffic using time-series of past type-of-traffic values (categorical); where all techniques are statistics based. The area of network traffic classification using ML methods within a time-series forecasting framework, in which the present work is located, is an area that, as far as we know, does not have identified works.

There are many works related with type-of-traffic classification applied to the identification of elephant states of a network flow, but extremely few related to the identification of active connections. In this area, the work in [32] presents a review of time-series classification models applied to the identification of the active state of network flows, the work shows a comparison of several ML models adapted to work with time-series data e.g. Logistic Regression, Random Forest... against alternative classic time-series models e.g. Hidden Markov Model (HMM), ARIMA, ARIMAX,... The best results are obtained with the ARIMAX model, but it is shown to be extremely demanding in terms of training and prediction times, indicating that alternative ML models are more effective from an operational point of view. The area of elephant flows detection is an active research field due to its importance for network management, traffic routing, congestion control and intrusion detection [2]. As a non-exhaustive summary of the approaches for elephant flow detection, we can classify the methods in ML based, statistics based and ad-hoc algorithm (mainly implemented in hardware at the network device level) based. Considering ML based models, in [7] are used decision trees (C4.5) for elephant flows detection in a software defined networking (SDN). Neural networks combined with the wavelet transform are applied in [4] for the detection of elephant flows in inter-data-center traffic. In [5] is presented a comparison between three ML techniques: neural networks, gaussian process regression and gaussian mixture models, for elephant flow prediction using a threshold to transform traffic volumes into a binary detection value. Making use of statistics based methods, [33] applies a sampling strategy based on statistical assumptions to process a subset of the total traffic while maintaining a specified probability of detecting elephant flows. Models based on ad-hoc algorithms can be implemented in software, but they are generally implemented in hardware on network devices to increase their efficiency, allowing real-time processing. In this line, [34] employs an iterative algorithm based on hash tables for elephant flow detection, which can be implemented in both hardware and software. The work in [6] provides an interesting review of different alternatives for the deployment of ad-hoc algorithms on different network elements with their impact on accuracy, resources and cost and network solutions available in each case. A combined hardware/software ad-hoc solution is presented in [35] for real-time elephant flows detection in SDN networks.

It is also interesting to mention the work done in models based on the theory of fuzzy neural networks, such as Takagi–Sugeno (T-S) fuzzy delayed neural networks [36,37], adaptive neuro fuzzy inference system (ANFIS) [38,39,40,41], models based on random projection with non-linear transforms such as Extreme Learning machines (ELM) [50] and advanced statistical methods of time-series analysis [42,43]. All the work mentioned, in these latter areas of research, focuses on regression problems, but they can be considered promising areas to investigate their applicability to classification problems. As mentioned earlier, the use of statistical analysis methods is well established for time-series forecasting, but their use in the case of categorical outputs is less known and could be the matter for additional studies.

3. WORK DESCRIPTION

This section introduces (1) the dataset used for all experiments carried out with all the models for this work, and (2) the gaNet-C model in detail, showing the similarities and differences with the original gaNet model.

3.1 Dataset

We will follow the dataset presented in [1] corresponding to real IoT traffic from a mobile operator. The dataset contains information on transmitted and received volumes of traffic of 6214 mobile devices for consecutive 632 time periods of 1-hour. Details on the dataset structure, data preparation and distribution of network traffic volumes is provided in [1].

Fig 1 offers a view of the arrangement made for the training dataset used in this work. Unlike [1], that proposes an algorithm to predict continuous values, in this work we want to perform a classification of the state of network flows, using categorical values instead of continuous ones. More specifically, we establish a binary classification for the state in each time period (e.g. active vs non-active state or elephant vs normal state).

Stating from the dataset presented in [1], which contains a time-series per device with volume of traffic aggregated in 1-hour slots, we transform it to create a dataset consisting of two time-series (with binary values) per device. One of these time-series is associated with the active connection status of each device (with a value of 1 for an active connection in each interval of 1-hour, and 0 otherwise), and the other time-series is associated with the elephant traffic status of each device (with a value of 1 for an elephant connection in each interval of 1-hour, and 0 otherwise). Since the objective of the experiments is to predict the state of the next k time periods based on the state of the n previous periods, we form the training dataset with a sequence of N vectors of previous activity (vectors with length n) and their associated vectors of future activity (vectors with length k). We have opted for 24 and 6 for the values of n and k respectively, which are similar values to those used in [1] and chosen for the same reasons. The number of samples for this dataset (value of N) is 130494, which are additionally grouped into a training and test sets with 105638 and 24856 samples, respectively. It is important to mention that the distribution of traffic volumes for this dataset is quite unbalanced [1], which is an additional challenge for any classifier that works with it.

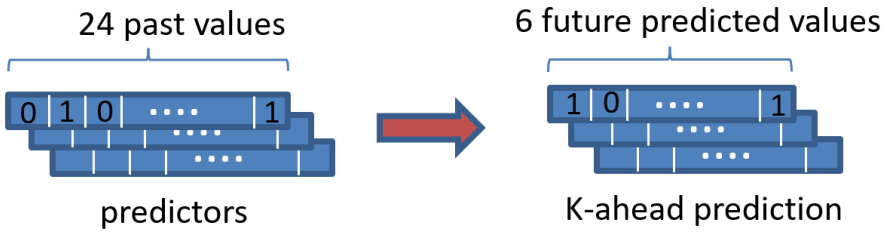

Fig 1. Structure of the training dataset: where n past categorical values are used to predict k future categorical values.

Once the traffic volumes for each flow have been aggregated and their corresponding states (activity or elephant) have been determined, it is possible to extract statistics on their frequency distributions. Fig 2 provides two charts with the frequency distribution for connections with active and elephant network traffic, where we can observe how the frequency distribution for the state of activity is quite balanced, while the frequencies for elephant traffic are very unbalanced, both corresponding to binary values. The opportunity to have these two scenarios is important, as it tests the proposed models under two very different conditions (balanced vs unbalanced dataset).

It is interesting to mention that for the elephant flows, the traffic volume in the 10% quantile of the connections that have the highest traffic corresponds to 98.67% of the total traffic volume. Therefore, for this particular dataset, the 10% quantile condition to define an elephant flow really imposes an even harder requirement, by creating a highly unbalanced dataset with connections classified into two traffic categories associated with 98.67% vs 1.33% of the total traffic volume.

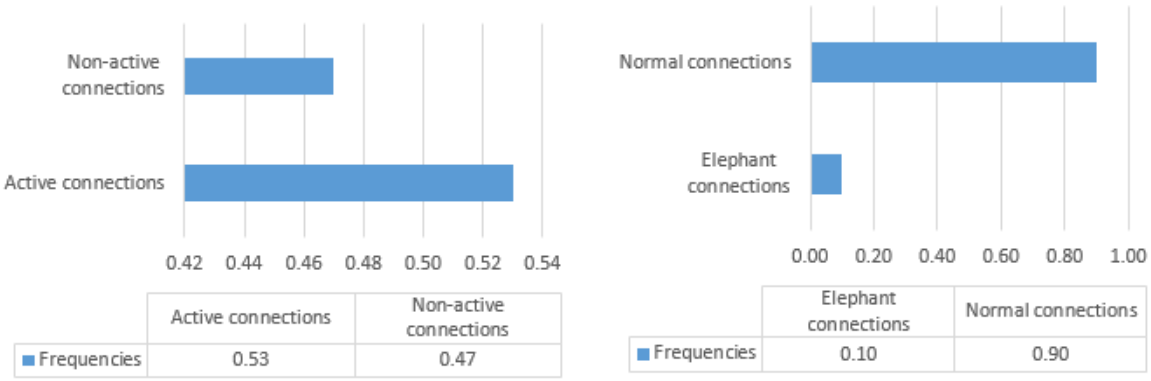

Fig 2. Distribution of active network traffic connections (left) and elephant connections (right)

3.2 gaNet-C model description

The main purpose of this work is to expand and adapt the gaNet architecture to be applicable to classification problems. We will not extend here to present in detail the original gaNet model which is presented in [1], but we will present in detail the modifications to gaNet necessary to transform it into a classifier (gaNet-C).

gaNet-C is an additive model formed by the sequential addition of 'building blocks'. The blocks are formed by independent NN architectures that can have a different number of layers and configurations. gaNet-C can have many variants, with all 'building blocks' sharing similar architectures or with completely different architectures. Common elements among all variants are: (1) All 'building blocks' have the same input. (2) The output of each block is added to the outputs of the other blocks. (3) The activation function for all layers is ReLU, with the exception of the final layer of each block, which has an activation of type: 'hyperbolic tangent'. (4) All blocks are trained together in an end-to-end training using SGD. (5) The joint output of the blocks is formed by the sum of the outputs of all the blocks, and serves itself as the input to a final fully connected network (FCN) with a single layer and a sigmoid activation. The output from this final FCN is the output of the model (Fig 3)

The gaNet-C variants considered for this work are only two: gaNet-C Type I and II. The original gaNet model presented four variants, but in reality, the important ones were only two, which are the two considered here.

gaNet-C Type I architecture assumes that all blocks have the same architecture (Fig 3). This variant is equivalent to the Type III variant of the original gaNet. In Fig 3 it is shown how the outputs from the blocks (h_k) are added to the sum of outputs of the previous blocks (F_k) and the final output (F_m) is the entry to the one-sigmoid-layer (a single layer with sigmoid output) FCN producing the final output (F_m^*). This final output is used in a log-loss cost function to implement an end-to-end training with all the blocks. The optimization method to find the lowest value of the cost function is SGD.

The log-loss cost function for gaNet-C Type I is:

$$Loss = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=1}^{C}\left[Y_{j,i}\log\left(F^*_{(j,m),i}\right) + \left(1 - Y_{j,i}\right)\log\left(1 - F^*_{(j,m),i}\right)\right] = LogLoss(Y, F_m^*) \quad (1)$$

where N is the total number of samples, and C is the number of label outputs (multi-label classification); in our case this number is 6, since we forecast the next 6 time-periods. The meaning of the different parts of expression (1) are: $Y_{j,i}$ is the value of the index position j of the true label (the j step ahead true value) for the sample number i. And, $F^*_{(j,m),i}$ is the value of the index position j of the final predicted label (the j step ahead predicted value) for the sample number i. The predicted value F_m^* corresponds with the final output value of the model after the one-sigmoid-layer FCN (Fig 3).

For the initial input to the model (F_0), we use the average value of the ground-truth labels; that is: $F_0 = Mean(Y)$. This initial configuration applies to all gaNet-C variants.

gaNet-C Type I admits two additional variants: gaNet-C Type I-A (Fig 4), is similar to gaNet-C Type I but with a modified cost function that consists in adding a cost function similar to expression (1) for each of the intermediate outputs of the model (F_k). To apply the log-loss cost function to the intermediate output values (F_k) it is necessary to add a one-sigmoid-layer FCN to these outputs to obtain the F_k^* values (Fig 4), which are scaled in the range [0-1] and are suitable for applying the log-loss function. The log-loss cost function for gaNet-C Type I-A is:

$$Loss = -\frac{1}{N}\sum_{k=1}^{m}\sum_{i=0}^{N}\sum_{j=1}^{C}\left[Y_{j,i} log\left(F^*_{(j,k),i}\right) + \left(1 - Y_{j,i}\right) log\left(1 - F^*_{(j,k),i}\right)\right] = \sum_{k=1}^{m} LogLoss(Y, F_k^*) \quad (2)$$

Where N and C have been previously defined, and m is the total number of blocks. The meaning of the different parts of

expression (2) is similar to expression (1) with the difference that now $F^*_{(j,m),i}$ becomes $F^*_{(j,k),i}$ where, instead of having a single value for the last output (m), we add all the intermediate outputs (k) of the model.

gaNet-C Type I-B is totally similar to gaNet-C Type I with the only difference that all blocks share their weights, that is, all blocks are not only similar but identical.

The gaNet-C Type II variant assumes that the blocks can have different architectures (Fig 5). This variant is equivalent to the Type IV variant of the original gaNet. In Fig 5, the difference in the architectures of the different blocks is represented using different symbols to identify them ($f(X)$, $\varphi(X)$, …. $\psi(X)$).

gaNet-C Type II admits a single additional variant: gaNet-C Type II-A, which is identical to gaNet-C Type I-A but formed by blocks with different architectures. Type II-B (with all blocks sharing the same weights) is not a feasible variant because the blocks do not have the same architecture, so they cannot share weights.

Some examples of possible architectures for the blocks are presented in Fig. 6. We can observe that there is freedom to choose the number of layers and their composition for the blocks, with the only restriction of having a final layer with a tanh (hyperbolic tangent) activation and a ReLU activation for all other layers. It has been proven that the scaling imposed by the final tanh layer is beneficial for the prediction performance of the classifier as shown in section 4, where an improvement is seen when using the tanh activation instead of a linear activation.

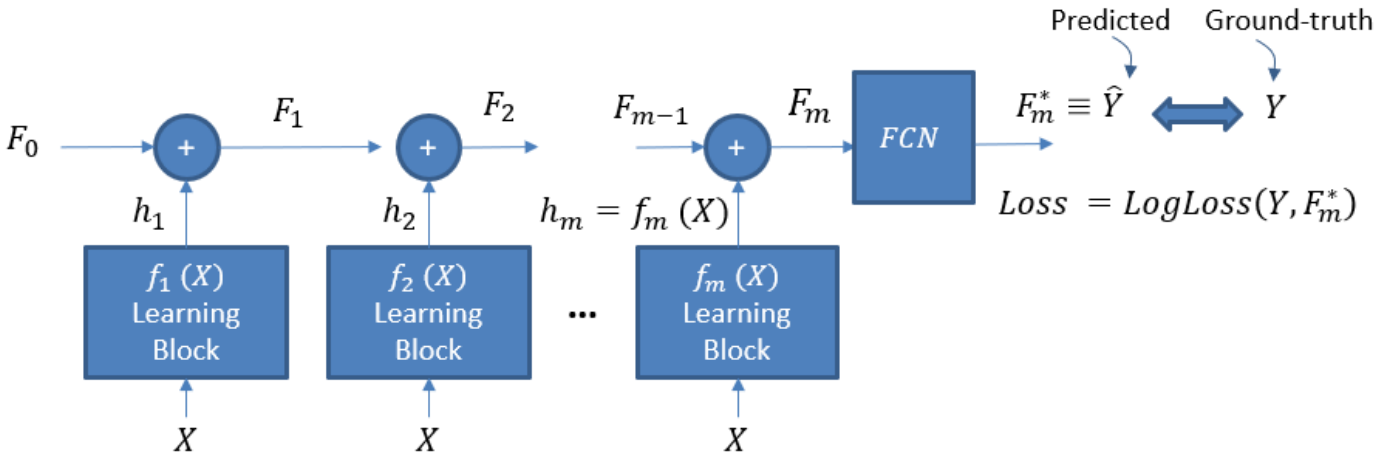

Fig 3. gaNet-C Type I architecture: Each learning block has the same architecture.

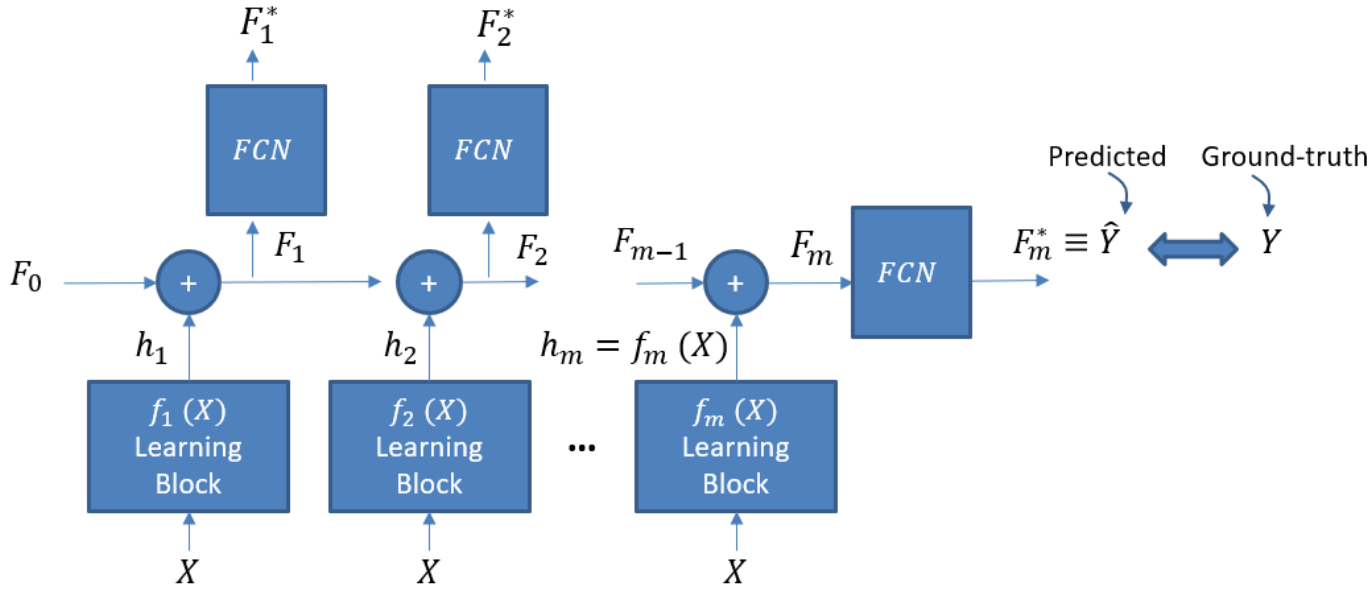

Fig 4. gaNet-C Type I-A architecture: Each learning block has the same architecture, with a loss formed by adding up all the losses of all intermediate outputs.

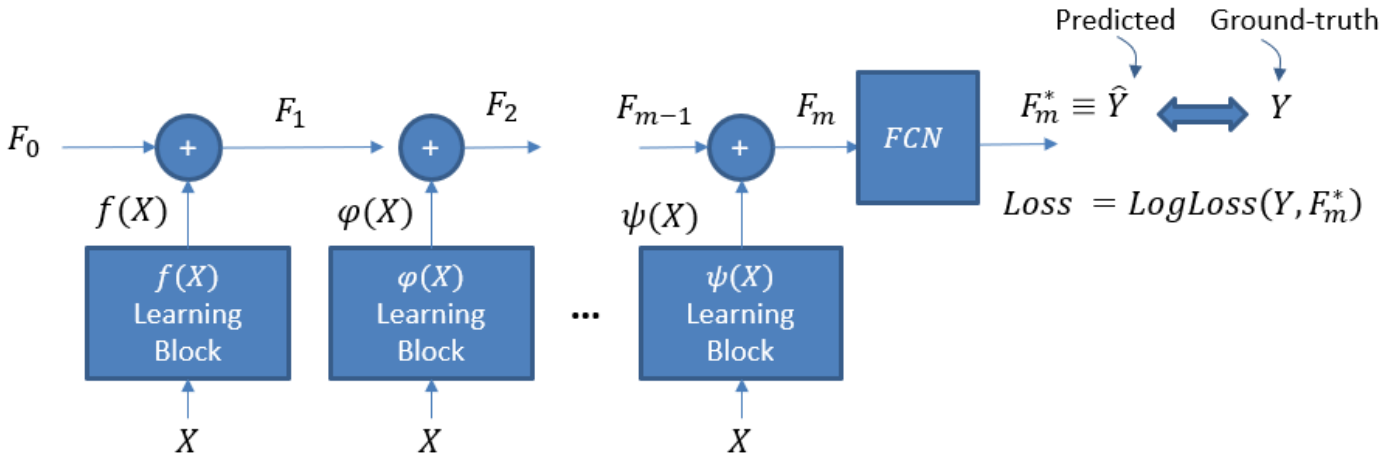

Fig 5. gaNet-C Type II architecture: Each learning block can have a different architecture

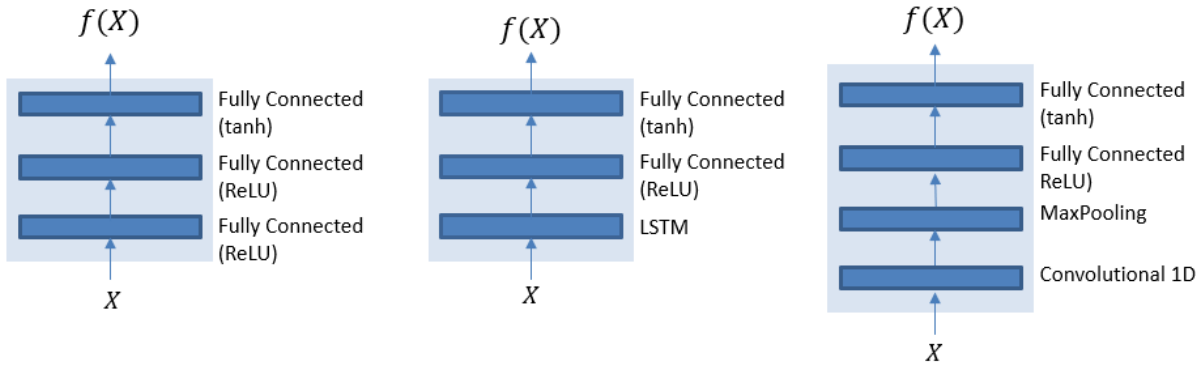

Fig 6. Learning block examples

As seen in the previous description of gaNet-C models, these can have many blocks and each block can have many layers. For such an architecture, with many layers, it is important to address the difficulties of convergence. gaNet-C handles this situation through several techniques: (1) the use of ReLU layers that have proven to be especially useful to prevent the possible vanishing or explosion of gradients in a deep network, and (2) using batch normalization, mainly in the architectures that use CNN layers.

gaNet-C architectures can be seen as connected with residual networks and stacked/ensemble models. All these models have regularization capabilities built into them (by construction). The highly appreciated regularization effect noted in stacked models (a type of ensemble models) is also observed in gaNet-C models [1] . The intrinsic regularization properties of gaNet-C can be observed in the results presented in section 4, where we can see that we do not need to apply regularization techniques (e.g. drop-out or L1/L2 regularization) to the models.

4. RESULTS

This section presents the results when the dataset presented in section 3.1 is used with different classifiers with the intention of rating the performance of gaNet-C against classic and state-of-the-art algorithms. The problem posed by the dataset is to perform a multi-label classification where the value of index k of the predicted label corresponds to a binary predicted value (binary classification) for the k-step ahead forecast. The binary output for each component of the label (6 components, Fig 1) corresponds to a type-of-traffic classification for two different scenarios: (1) detection of activity for a connection in a certain time period or (2) detection of an extremely large volume of traffic for that connection (elephant flow) in a certain time period. For each of these scenarios, we present separate results (section 4.1 and 4.2).

The different algorithms have been categorized into groups (Fig. 7 and 12): (a) Sequence to sequence. (b) Sequence to sequence plus attention. With different encoder and decoder architectures for both groups. (c) NN architectures based on recurrent (LSTM), convolutional (CNN), fully connected (FCN) layers, and combinations of them (d) Classic ML models: random forest and logistic regression. And, (e) the different gaNet-C models based on their different variants.

The metrics used for classification forecast have been accuracy, F1-score, precision and recall. To define these metrics, we have considered that the presence of an active connection or an elephant connection is a 'positive' result and the opposite state a 'negative' result; with these definitions, a true positive (TP) is a positive prediction corresponding to a real (ground-truth) positive state, a true negative (TN) is a negative prediction corresponding to a real negative state, a false positive (FP) is a positive prediction corresponding to a real negative state, and a false negative (FN) is a negative prediction corresponding to a real positive state. Then, the metrics can be defined as:

$$Accuracy = \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN} \quad (3)$$

$$Precision = \frac{\#TP}{\#TP + \#FP} \quad (4)$$

$$Recall = \frac{\#TP}{\#TP + \#FN} \quad (5)$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (6)$$

Where a '#' sign before a symbol indicates the total number of that symbol e.g. $\#TP$ is the total number of true positives in our prediction results.

Fig. 7 and 12 present three groups of columns, with the first group corresponding to the average results for all k-ahead steps forecasts, the second group (T0) for the prediction results of the time-period closest to the known past values, and the third group (T5) that gives the results for the last time-period, considering that we are providing a forecast of 6 time-periods steps ahead.

The F1-score is more suitable for an unbalanced dataset, since accuracy in that case is a misleading metric. Precision and recall are also important depending on the application and the relative importance of false positives and false negatives. If false negatives are important and we need to reduce them (e.g. tumor detector) then we need a high recall, while if false positives are important and we need to reduce them (e.g. movie recommender in commercial web) then we need a high precision. The F1-score is the geometric mean of both precision and recall and attempts to provide a uniquely significant metric of detection performance. The four metrics (accuracy, F1, precision and recall) have the same range of values between 0 and 1, with a value of 0 being the worst result and a value of 1 for the best.

The 'Model' column in Fig 7 and 12 provides additional information on the corresponding model. For the NN models, describes the number of CNN, LSTM and/or FC layers. The additive blocks of the gaNet-C models are described by the list of layers of each block included in parenthesis, with an asterisk to the right of the parenthesis and a number that indicates how many times (m) that block is repeated. The numbers given within the parenthesis and before a layer type indicate the consecutive number of layers of that type. For example, a gaNet-C Type I identified as (2 LSTM + 1 FC)*3 corresponds to a model with three additive blocks, all three with the same architecture: two LSTM layers followed by one FC layer. For the gaNet-C Type II variant, which can have blocks with different architectures, each block is represented in the manner mentioned previously with a '+' sign to join the different architectures.

The implementation of the ML models (random forest and logistic regression) was carried out in python with the scikit-learn package [44], using Tensorflow / Keras [45] for the rest of the models.

We have not used any regularization techniques (e.g. L1 or L2) with the exception of drop-out that is clearly indicated in the models in which it has been applied. To train the models, we used gradient descent with an Adam optimizer with a learning rate of 0.001 and 0.9 and 0.99 for the β_1 and β_2 parameters, which are the default parameters in [46].

All the results presented in this section have been obtained with the test set described in section 3.1. This test set consists of a random sampling of 20% of the full dataset. In addition, in the resulting training set, consisting of the remaining 80%, we have detached another 20% as a validation set. This validation set is especially important for training SGD-based models, since the validation set is used to know when to stop training. An early-stopping criterion is used to stop training when a chosen performance metric applied to the validation set does not improve in a certain number of training cycles (epochs). In our case, we have used 10 epochs for the early-stopping criterion with an overall result of between 10 to 100 epochs required for training, depending on the model and its difficulty to converge.

The resulting data scheme used for training comprises three sets: training, validation and test with 64%, 16% and 20% of the complete dataset, respectively. The test set remains fixed for all models, while the validation and training sets contain a random distribution of samples for each training epoch. All results presented in this section are obtained using only the test set.

As a summary, the process flow (tasks sequence) followed to obtain the results presented in this section, has been: (1) Data acquisition. (2) Data preparation. (3) Validation strategy and dataset partitioning (training, validation, test). (4) Selection of performance metrics. (5) Selection of models to perform the comparison of results. (6) Comparison of classification results and running times for the different algorithms. (7) Conclusions.

4.1 Classification of active connections

Fig 7 provides the metrics for the 6-steps ahead forecasting of the 'activity state' of each connection in periods of 1-hour, following the dataset described in section 3.1 (Fig 1). To perform the training and prediction, we use the previous 24 activity values (binary values) for each connection. As mentioned earlier, the Table in Fig 1 provides four performance metrics: accuracy, F1-score, prediction and recall, for 3 groups of predictions: average (for all time steps), T0 (for the first time-step predicted) and T5 (for the sixth time-step predicted). Fig 8 and 9 give the same information contained in Fig 7 but in a chart format and focusing only on the average F1 (Fig 8) and average accuracy (Fig 9) for the most significant results, to avoid cluttering the diagrams.

The table in Fig 7 is color-coded, with a green color indicating a good result and a red color a bad result; the color palette between green and red is used to indicate intermediate results. In summary, the results are coded with the greenest for the best result and the reddest for the worst result.

Taking into account the results in Fig 7, we can see that Seq2Seq and Seq2Seq+Attention provide quite bad results, which is paradoxical considering that these models are specifically intended for time-series forecasting. Random forest also gives poor results, but logistic regression, despite its simplicity, provides quite satisfactory results. For each of these two methods, we need to train a different model for each of the 6 output values, since they are not multi-output methods. It is also interesting that for gaNet-C the best results are for the Type I variant, with the other variants obtaining quite unsatisfactory scores.

For best results, we can see that these are found with the following models: (1) NN models: mainly with architectures based on several CNN layers or based on several (6 or more) FC layers. The addition of LSTM layers does not provide any advantage for these configurations. (2) gaNet-C Type I models: with blocks formed by simple FC layers: (3 FC)*2 and (6 FC)*5, and combinations of LSTM and FC layers: (1 LSTM + 1 FC)*2. The configurations with blocks that use CNN layers provide poor results.

It is interesting to note the concentration of good results for gaNet-C Type I models, and that in order to obtain the best results it is not necessary to add many blocks: in general, a number of blocks between 2 and 6 provides the best results. We can also observe how the prediction results for T0 and T5 are also better, in general, for gaNet-C Type I models. This ability of gaNet-C to continue offering good results for distant time-ahead predictions can be considered as an explanation for the good average results obtained, as explained in more detail in section 4.3.

		Average				T0				T5			
Class	Model	Accuracy	F1	Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1	Precision	Recall
Seq2Seq	1 LSTM + 1 FC	0.6068	0.4692	0.9421	0.3124	0.9445	0.9497	0.9596	0.9401	0.4493	0.0498	0.6673	0.0259
	2 LSTM + 1 FC	0.6103	0.4804	0.9306	0.3237	0.9478	0.9529	0.9596	0.9462	0.4528	0.0659	0.6916	0.0346
	2 CNN+ 1 LSTM + 1 FC	0.7723	0.8192	0.7336	0.9275	0.8980	0.9045	0.9456	0.8668	0.5579	0.7162	0.5579	1.0000
Seq2Seq + Attention	1 LSTM + 1 FC	0.6117	0.4813	0.9371	0.3238	0.9465	0.9516	0.9599	0.9435	0.4587	0.0836	0.7534	0.0443
	2 LSTM + 1 FC	0.6090	0.4732	0.9452	0.3156	0.9476	0.9525	0.9625	0.9428	0.4503	0.0461	0.7237	0.0238
	2 CNN+ 1 LSTM + 1 FC	0.7902	0.8347	0.7430	0.9523	0.7948	0.8260	0.7832	0.8737	0.6061	0.7366	0.5874	0.9875
NN Achitectures	1 LSTM + 1 FC	0.9240	0.9306	0.9456	0.9161	0.9371	0.9431	0.9514	0.9349	0.9175	0.9239	0.9520	0.8975
	2 LSTM + 1 FC	0.9266	0.9331	0.9470	0.9196	0.9440	0.9492	0.9605	0.9381	0.9179	0.9241	0.9542	0.8959
	2 CNN + 1 FC	0.9298	0.9363	0.9460	0.9267	0.9452	0.9503	0.9597	0.9412	0.9216	0.9281	0.9508	0.9064
	2 CNN (MaxPooling) + 1 FC	0.9316	0.9377	0.9500	0.9258	0.9468	0.9518	0.9612	0.9426	0.9245	0.9307	0.9536	0.9089
	2 CNN + 1 LSTM + 1 FC	0.9301	0.9361	0.9526	0.9201	0.9473	0.9522	0.9627	0.9420	0.9200	0.9261	0.9549	0.8990
	2 CNN + 2 LSTM + 1 FC	0.9260	0.9324	0.9487	0.9166	0.9448	0.9500	0.9607	0.9395	0.9147	0.9213	0.9499	0.8943
	3 FC	0.9014	0.9157	0.8734	0.9624	0.9166	0.9284	0.8904	0.9698	0.8895	0.9058	0.8640	0.9518
	6 FC	0.9304	0.9367	0.9482	0.9255	0.9445	0.9499	0.9569	0.9430	0.9224	0.9279	0.9628	0.8955
Alternative ML	Random Forest	0.9257	0.9258	0.9262	0.9257	0.9417	0.9473	0.9548	0.9399	0.9144	0.9216	0.9418	0.9023
	Logistic Regression	0.9290	0.9291	0.9297	0.9290	0.9433	0.9486	0.9580	0.9394	0.9212	0.9277	0.9503	0.9061
gaNet-C Type I	(3 FC)*2	0.9318	0.9380	0.9492	0.9272	0.9465	0.9515	0.9616	0.9415	0.9216	0.9280	0.9519	0.9053
	(3 FC)*2 (linear activation)	0.9318	0.9379	0.9497	0.9265	0.9468	0.9519	0.9597	0.9443	0.9237	0.9301	0.9521	0.9090
	(3 FC)*3	0.9316	0.9377	0.9514	0.9243	0.9464	0.9516	0.9587	0.9446	0.9233	0.9295	0.9537	0.9066
	(3 FC)*5	0.9317	0.9379	0.9501	0.9260	0.9462	0.9514	0.9582	0.9447	0.9228	0.9287	0.9581	0.9011
	(3 FC)*10	0.9295	0.9359	0.9472	0.9249	0.9464	0.9516	0.9579	0.9454	0.9201	0.9269	0.9474	0.9072
	(3 FC)*10 + dropout(0.5)	0.9212	0.9285	0.9375	0.9197	0.9348	0.9409	0.9501	0.9319	0.9057	0.9140	0.9301	0.8985
	(6FC)*2	0.9290	0.9349	0.9549	0.9156	0.9440	0.9493	0.9586	0.9401	0.9213	0.9271	0.9594	0.8969
	(6 FC)*5	0.9320	0.9379	0.9535	0.9229	0.9463	0.9517	0.9557	0.9477	0.9218	0.9278	0.9569	0.9003
	(6 FC)*5 (linear activation)	0.9307	0.9367	0.9531	0.9209	0.9454	0.9507	0.9567	0.9448	0.9207	0.9265	0.9265	0.8965
	(6 FC)*5 + dropout(0.3)	0.9257	0.9320	0.9486	0.9160	0.9447	0.9498	0.9613	0.9386	0.9132	0.9201	0.9451	0.8965
	(6 FC + dropout(0.3))*5	0.9289	0.9355	0.9453	0.9258	0.9455	0.9508	0.9564	0.9453	0.9217	0.9283	0.9494	0.9081
	(1 LSTM + 1 FC)*2	0.9316	0.9377	0.9501	0.9257	0.9484	0.9534	0.9609	0.9459	0.9206	0.9274	0.9465	0.9091
	(1 LSTM + 1 FC)*3	0.9286	0.9350	0.9467	0.9236	0.9460	0.9513	0.9568	0.9459	0.9180	0.9247	0.9487	0.9019
	(2 LSTM + 1 FC)*2	0.9302	0.9367	0.9460	0.9275	0.9466	0.9517	0.9591	0.9445	0.9182	0.9253	0.9430	0.9082
	(2 LSTM + 1 FC)*3	0.9312	0.9372	0.9508	0.9241	0.9469	0.9520	0.9593	0.9449	0.9209	0.9272	0.9534	0.9023
	(2 CNN (MaxPooling) + 1 FC)*2	0.9263	0.9333	0.9406	0.9261	0.9435	0.9490	0.9556	0.9424	0.9146	0.9224	0.9358	0.9094
gaNet-C Type I-A	(3 FC)*2	0.9291	0.9356	0.9454	0.9260	0.9450	0.9505	0.9537	0.9472	0.9189	0.9260	0.9426	0.9100
	(4 FC)*3	0.9286	0.9349	0.9489	0.9213	0.9457	0.9509	0.9584	0.9436	0.9209	0.9274	0.9504	0.9055
	(2 LSTM + 1 FC)*5	0.9176	0.9247	0.9404	0.9095	0.9297	0.9355	0.9574	0.9146	0.9050	0.9134	0.9298	0.8975
gaNet-C Type I-B	(3 FC)*3	0.9214	0.9276	0.9514	0.9051	0.9308	0.9364	0.9590	0.9149	0.9156	0.9221	0.9510	0.8949
	(4 FC)*10	0.9072	0.9117	0.9695	0.8603	0.9257	0.9308	0.9686	0.8958	0.8611	0.8588	0.9920	0.7572
	(2 LSTM + 1 FC)*5	0.9169	0.9239	0.9420	0.9065	0.9305	0.9369	0.9480	0.9262	0.9077	0.9155	0.9354	0.8965
gaNet-C Type II	(2 LSTM + 1 FC)*2 + (3 FC)*2	0.9237	0.9304	0.9452	0.9161	0.9337	0.9398	0.9515	0.9285	0.9139	0.9212	0.9417	0.9016
	(2 LSTM + 1 FC)*2 + (6 FC)*2	0.9237	0.9299	0.9505	0.9103	0.9378	0.9429	0.9644	0.9224	0.9124	0.9192	0.9476	0.8923
	(2 LSTM + 1 FC)*3 + (6 FC)*3	0.9242	0.9306	0.9480	0.9139	0.9366	0.9421	0.9592	0.9256	0.9190	0.9249	0.9580	0.8940
	(1 LSTM + 1 FC)*2 + (2 LSTM + 1 FC)*2 + (6 FC)*2	0.9204	0.9267	0.9509	0.9037	0.9344	0.9404	0.9530	0.9280	0.9103	0.9168	0.9497	0.8861
gaNet-C Type II-A	(2 LSTM + 1 FC)*2 + (6 FC)*2	0.9212	0.9276	0.9486	0.9075	0.9318	0.9376	0.9564	0.9195	0.9107	0.9171	0.9520	0.8846

Fig 7. Performance metrics for classification of active connections for all algorithms.

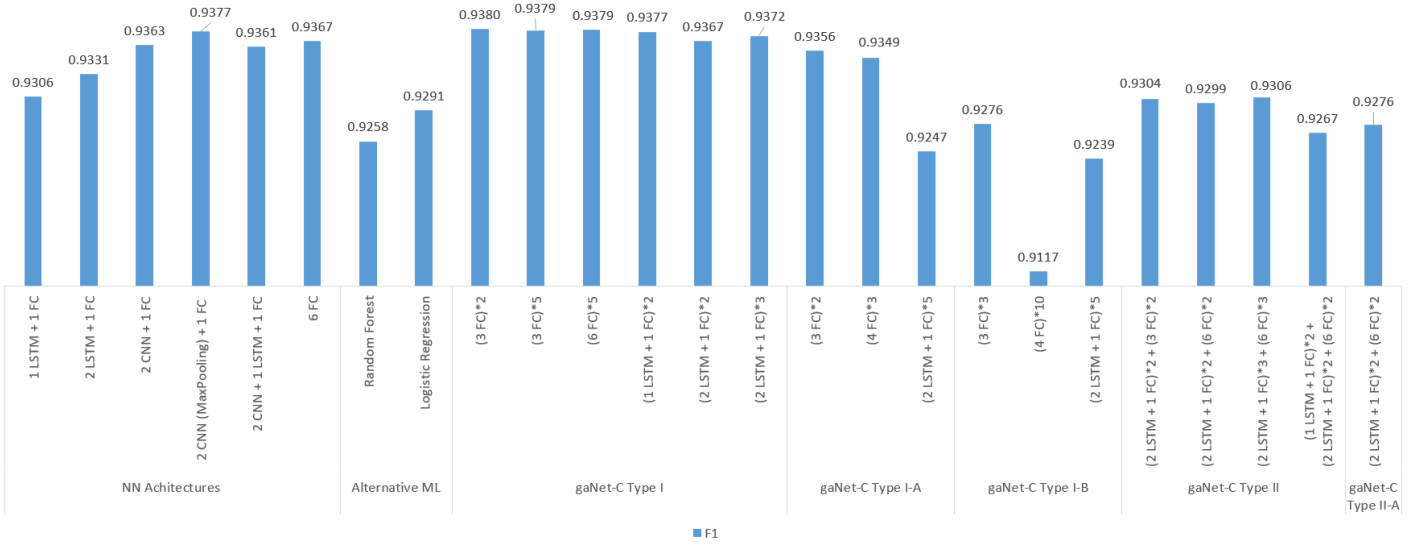

Fig 8. Comparison of average $F1$results for classification of active connections

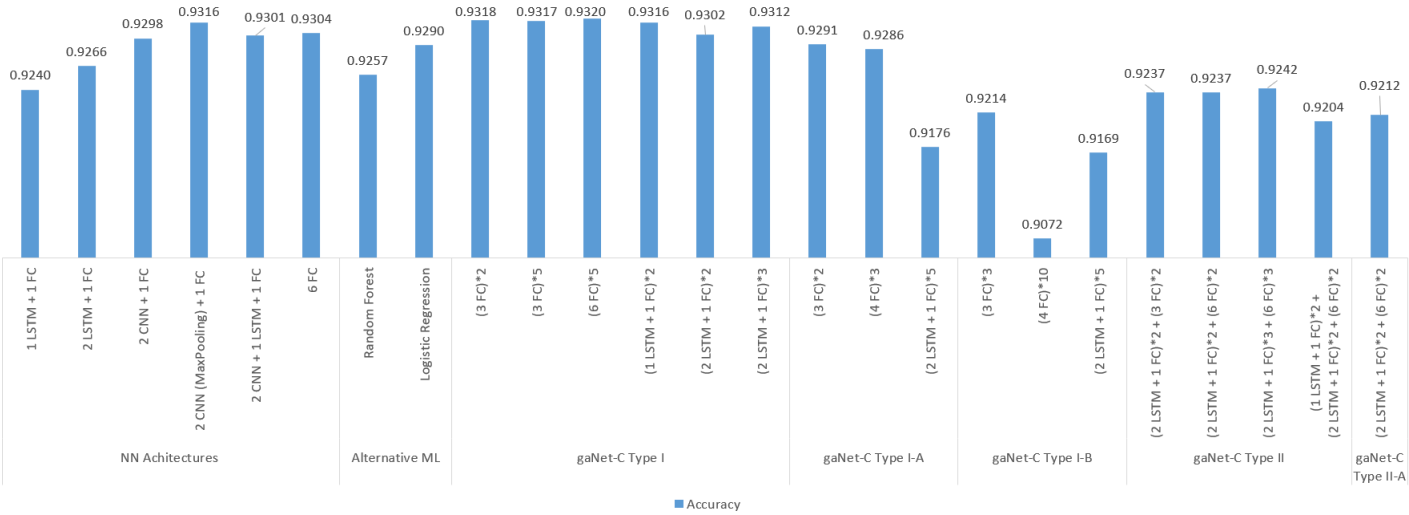

Fig 9. Comparison of average Accuracy for classification of active connections

Forecasting the type-of-traffic of network connections requires not only good prediction performance, but also limited training and prediction times, due to the real-time nature of the applications (e.g. admission control, routing, resource management...) where the classifier will be deployed.

Fig 10 and 11 provide the training and prediction times necessary to perform training and prediction with the dataset described in section 3.1 for forecasting active connections. As expected, logistic regression requires the smallest training and prediction times, but it is necessary to train a model for each output value because we are performing multi-label classification (i.e. more than one positive and concurrent output value) for which logistic regression is not a suitable model. The same problem is found with random forest, for which training and prediction times are also small. As expected, Seq2Seq and Seq2Seq+Attention models require longer training and prediction times. NN models need different computational times depending on whether they include CNN, LSTM and combinations of them.

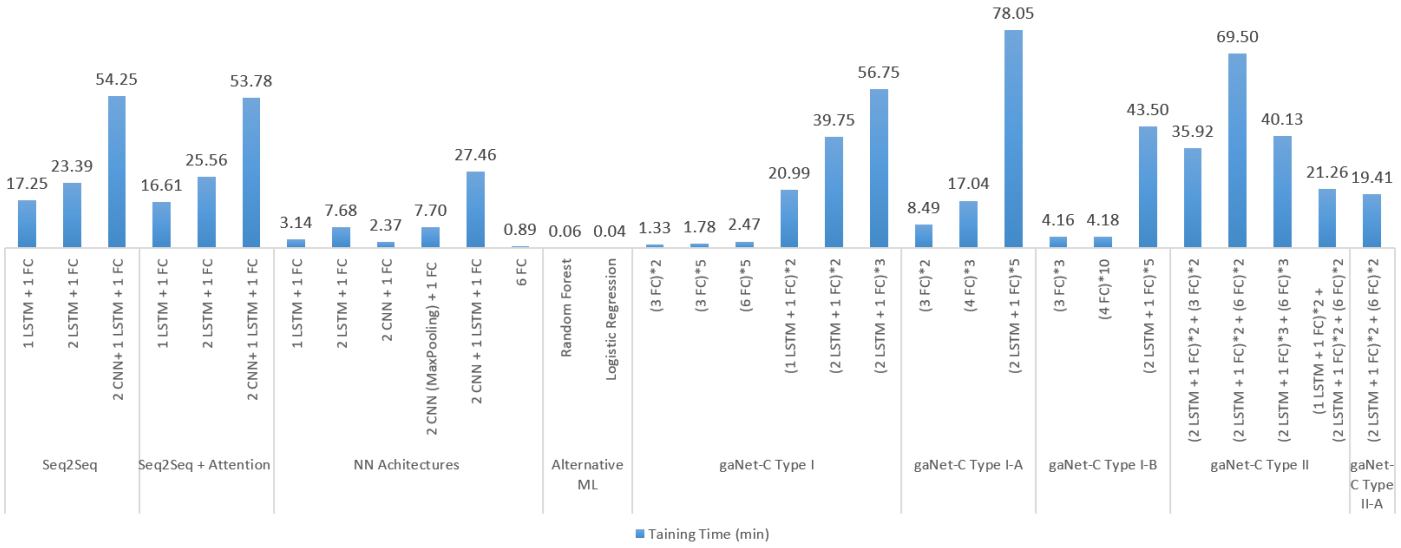

Fig 10. Training times (minutes) for all algorithms for classification of active connections

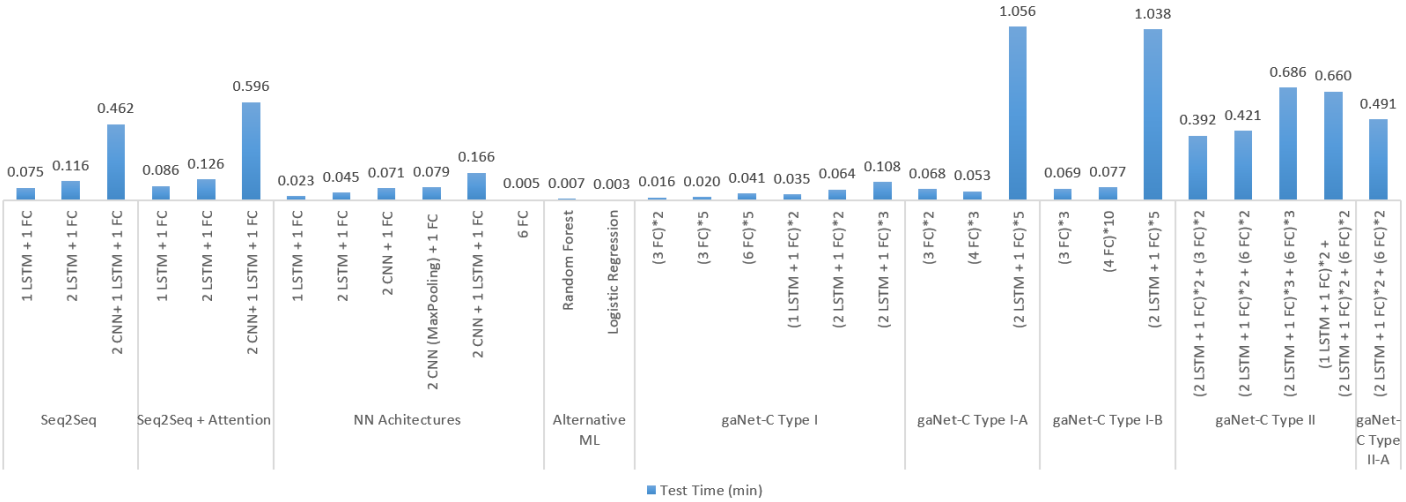

Fig 11. Prediction times (minutes) for all algorithms for classification of active connections

4.2 Classification of elephant connections

The structure of this section is similar to section 4.1, but in this case considering the results for the 6-steps ahead forecasting of the 'elephant state' of each connection in periods of 1-hour (Fig 12). The color-coding, structure and information contained in Fig 12, 13 and 14 are also similar to those provided in the previous section. In particular, Fig 13 and 14 provide a visual summary of the average F1 (Fig 13) and average accuracy (Fig 14) for the most significant results, to avoid cluttering the diagrams.

Analyzing the results in Fig 12 we can observe how Seq2Seq and Seq2Seq + Attention offer again the worst results followed by logistic regression. It is interesting how logistic regression obtained some of the best results for the problem of detecting the activity of connections (section 4.1) and one of the worst to classify connections as elephant or not. Similar to section 4.1 for the detection of active connections, the best results for the detection of elephant flows are found with the following models: (1) NN models: with architectures based on combinations of CNN and LSTM layers: 2 CNN + 2 LSTM + 1 FCN. And, (2) gaNet-C Type I models: with blocks based only on FC layers: (3 FC)*10, and combinations of LSTM and FC layers: (1 LSTM + 1 FC)*3. Unlike section 4.1, the other variants of gaNet-C, in addition to Type I, provide good results; some of them comparable with the best results of gaNet-C Type I, for example: Type I-A (3 FC)*2, Type II (2 LSTM + 1 FC)*2 + (3 FC)*2, as well as Type II-A (2 LSTM + 1 FC)*2 + (6 FC)*2.

It is also worth noting that the best gaNet-C Type I models offer the best average results, as well as, in general, the best results for the first and last prediction periods (shown in columns T0 and T5 in Fig 12).

		Average				T0				T5			
Class	Model	Accuracy	F1	Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1	Precision	Recall
Seq2Seq	1 LSTM + 1 FC	0.9060	0.3268	0.8260	0.2037	0.9435	0.7309	0.8288	0.6536	0.8898	0.0051	1.0000	0.0026
	2 LSTM + 1 FC	0.9062	0.3260	0.8357	0.2025	0.9428	0.7290	0.8214	0.6554	0.8896	0.0015	0.5000	0.0007
	2 CNN+ 1 LSTM + 1 FC	0.9302	0.5892	0.8656	0.4466	0.9276	0.5964	0.8631	0.4556	0.9294	0.5647	0.8836	0.4149
Seq2Seq + Attention	1 LSTM + 1 FC	0.9062	0.3296	0.8255	0.2059	0.9429	0.7258	0.8324	0.6434	0.8904	0.0194	0.9331	0.0098
	2 LSTM + 1 FC	0.9080	0.3449	0.8513	0.2162	0.9424	0.7229	0.8313	0.6396	0.8900	0.0116	0.7619	0.0058
	2 CNN+ 1 LSTM + 1 FC	0.9274	0.6282	0.7369	0.5474	0.9293	0.6121	0.8599	0.4752	0.9199	0.6077	0.6614	0.5621
NN Achitectures	1 LSTM + 1 FC	0.9408	0.6826	0.8547	0.5681	0.9417	0.7108	0.8506	0.6105	0.9389	0.6586	0.8607	0.5333
	2 LSTM + 1 FC	0.9404	0.6790	0.8562	0.5626	0.9421	0.7119	0.8565	0.6091	0.9373	0.6481	0.8526	0.5228
	2 CNN + 1 FC	0.9394	0.6723	0.8521	0.5552	0.9421	0.7196	0.8336	0.6331	0.9361	0.6249	0.8885	0.4820
	2 CNN (MaxPooling) + 1 FC	0.9397	0.6749	0.8531	0.5583	0.9405	0.7033	0.8478	0.6009	0.9380	0.6506	0.8613	0.5228
	2 CNN + 1 LSTM + 1 FC	0.9395	0.6720	0.8570	0.5526	0.9426	0.7262	0.8261	0.6478	0.9366	0.6433	0.8503	0.5173
	2 CNN + 2 LSTM + 1 FC	0.9409	0.6884	0.8401	0.5831	0.9438	0.7324	0.8322	0.6540	0.9375	0.6506	0.8506	0.5268
	3 FC	0.9404	0.6860	0.8372	0.5810	0.9422	0.7197	0.8356	0.6321	0.9389	0.6711	0.8271	0.5647
	6 FC	0.9390	0.6643	0.8670	0.5384	0.9413	0.7111	0.8434	0.6146	0.9363	0.6328	0.8700	0.4973
Alternative ML	Random Forest	0.9367	0.6765	0.7906	0.5912	0.9391	0.7075	0.8113	0.6273	0.9342	0.6506	0.7870	0.5545
	Logistic Regression	0.9135	0.6813	0.5803	0.8247	0.9218	0.7132	0.6263	0.8280	0.9055	0.6425	0.5518	0.7690
gaNet-C Type I	(3 FC)*2	0.9405	0.6816	0.8512	0.5684	0.9409	0.7115	0.8338	0.6204	0.9390	0.6573	0.8660	0.5297
	(3 FC)*3	0.9411	0.6854	0.8525	0.5731	0.9426	0.7196	0.8438	0.6273	0.9397	0.6748	0.8342	0.5665
	(3 FC)*5	0.9408	0.6921	0.8291	0.5939	0.9432	0.7281	0.8312	0.6478	0.9377	0.6592	0.8331	0.5454
	(3 FC)*10	0.9411	0.6918	0.8357	0.5902	0.9421	0.7156	0.8458	0.6201	0.9385	0.6725	0.8160	0.5719
	(3 FC)*10 + dropout(0.5)	0.9381	0.6615	0.8529	0.5403	0.9394	0.6849	0.8796	0.5608	0.9325	0.6180	0.8230	0.4947
	(6 FC)*5	0.9405	0.6847	0.8426	0.5767	0.9427	0.7249	0.8305	0.6430	0.9383	0.6486	0.8745	0.5155
	(6 FC + dropout(0.3))*5	0.9406	0.6899	0.8316	0.5894	0.9422	0.7162	0.8456	0.6211	0.9392	0.6696	0.8375	0.5577
	(1 LSTM + 1 FC)*2	0.9410	0.6866	0.8480	0.5768	0.9439	0.7256	0.8528	0.6314	0.9380	0.6563	0.8473	0.5355
	(1 LSTM + 1 FC)*3	0.9415	0.6974	0.8285	0.6021	0.9431	0.7317	0.8195	0.6608	0.9389	0.6687	0.8340	0.5581
	(2 LSTM + 1 FC)*2	0.9409	0.6871	0.8452	0.5789	0.9433	0.7247	0.8425	0.6358	0.9391	0.6675	0.8411	0.5534
	(2 LSTM + 1 FC)*3	0.9406	0.6857	0.8421	0.5783	0.9416	0.7152	0.8366	0.6245	0.9393	0.6763	0.8225	0.5741
	(2 CNN (MaxPooling) + 1 FC)*2	0.9404	0.6874	0.8325	0.5854	0.9408	0.7134	0.8263	0.6276	0.9384	0.6605	0.8432	0.5428
gaNet-C Type I-A	(3 FC)*2	0.9406	0.6904	0.8293	0.5913	0.9416	0.7183	0.8282	0.6341	0.9385	0.6606	0.8455	0.5421
	(4 FC)*3	0.9403	0.6856	0.8360	0.5811	0.9417	0.7179	0.8307	0.6321	0.9384	0.6535	0.8636	0.5257
	(2 LSTM + 1 FC)*5	0.9377	0.6759	0.8104	0.5797	0.9359	0.6860	0.8078	0.5961	0.9349	0.6444	0.8114	0.5344
gaNet-C Type I-B	(3 FC)*3	0.9317	0.6155	0.8340	0.4878	0.9405	0.7051	0.8442	0.6053	0.9392	0.6631	0.8546	0.5417
	(4 FC)*10	0.9353	0.6300	0.8765	0.4918	0.9359	0.6722	0.8418	0.5594	0.9310	0.5634	0.9342	0.4033
	(2 LSTM + 1 FC)*5	0.9376	0.6611	0.8450	0.5429	0.9352	0.6596	0.8605	0.5348	0.9400	0.6677	0.8599	0.5457
gaNet-C Type II	(2 LSTM + 1 FC)*2 + (3 FC)*2	0.9403	0.6932	0.8172	0.6019	0.9415	0.7252	0.8081	0.6578	0.9380	0.6714	0.8097	0.5734
	(2 LSTM + 1 FC)*2 + (6 FC)*2	0.9391	0.6693	0.8550	0.5498	0.9380	0.6771	0.8708	0.5540	0.9367	0.6455	0.8463	0.5217
	(2 LSTM + 1 FC)*3 + (6 FC)*3	0.9394	0.6788	0.8362	0.5713	0.9415	0.7110	0.8459	0.6132	0.9370	0.6476	0.8479	0.5239
	(1 LSTM + 1 FC)*2 + (2 LSTM + 1 FC)*2 + (6 FC)*2	0.9399	0.6825	0.8361	0.5765	0.9395	0.7204	0.7875	0.6639	0.9393	0.6622	0.8598	0.5384
gaNet-C Type II-A	(2 LSTM + 1 FC)*2 + (6 FC)*2	0.9395	0.6940	0.8007	0.6124	0.9380	0.7060	0.7973	0.6334	0.9376	0.6713	0.8017	0.5774

Fig 12. Performance metrics for classification of elephant connections for all algorithms.

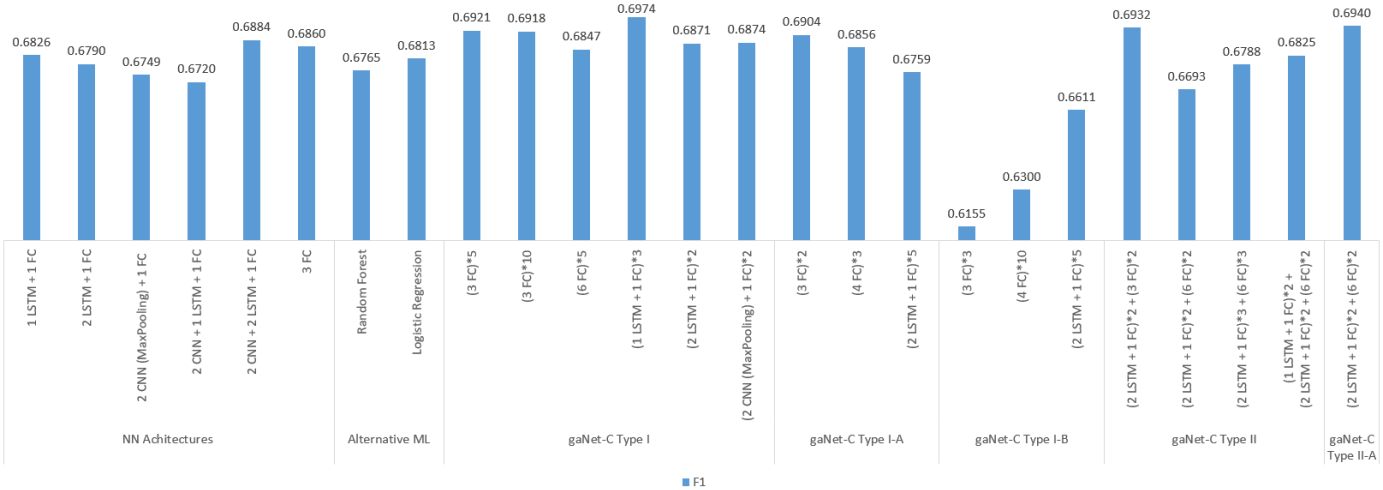

Fig 13. Comparison of average *F*1results for classification of elephant connections

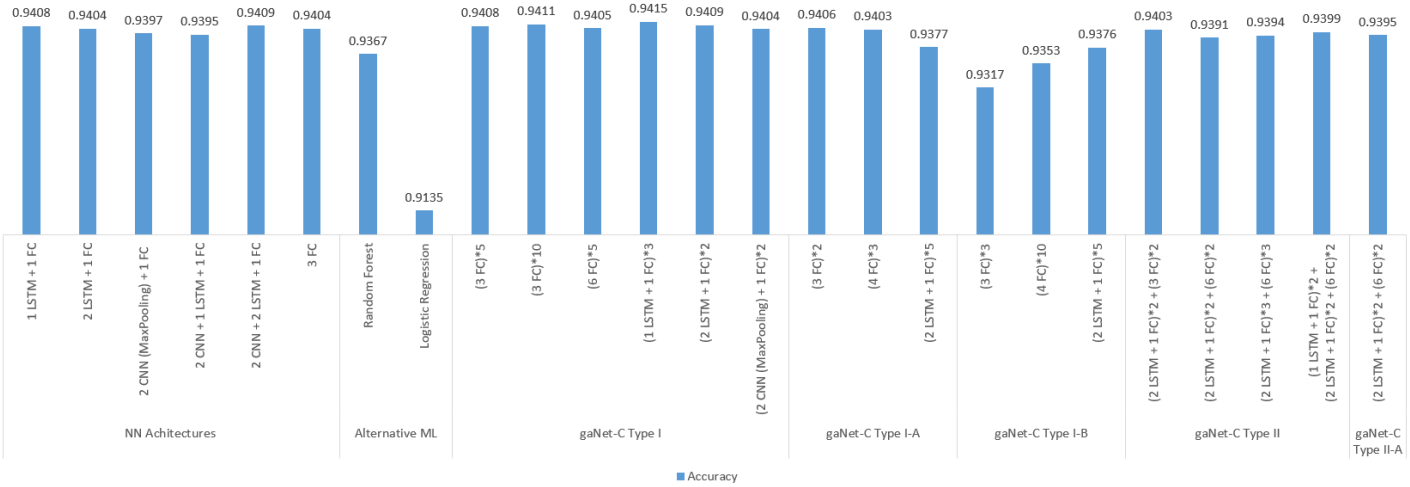

Fig 14. Comparison of average Accuracy results for classification of elephant connections

Fig 15 and 16 provide the training and prediction times necessary to perform training and prediction with the dataset described in section 3.1 to forecast elephant connections. The computational requirements for this scenario are quite similar to those obtained in the scenario for forecasting active connections (Fig 10 and 11). Training times (Fig 15) are similar in both scenarios for the classic ML algorithms, but with significant fluctuations for the models implemented with gradient descent due to the aforementioned early-stopping criterion, which halts training when there is no increase in prediction accuracy after a predefined number of training epochs. This is a stochastic criterion that can introduce differences in training times.

Prediction times (Fig 16) are also similar in both scenarios, being almost identical for all models, with the exception of gaNet-C Type I models with LSTM layers, where for the scenario of forecasting elephant connections the times are noticeably higher.

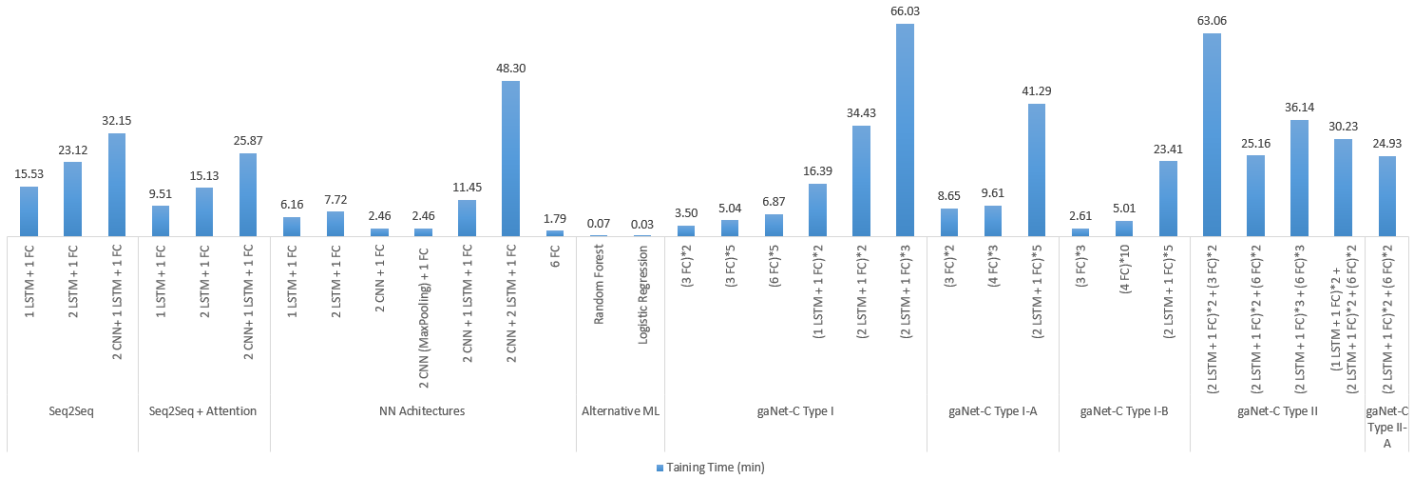

Fig 15. Training times (minutes) for all algorithms for classification of elephant connections

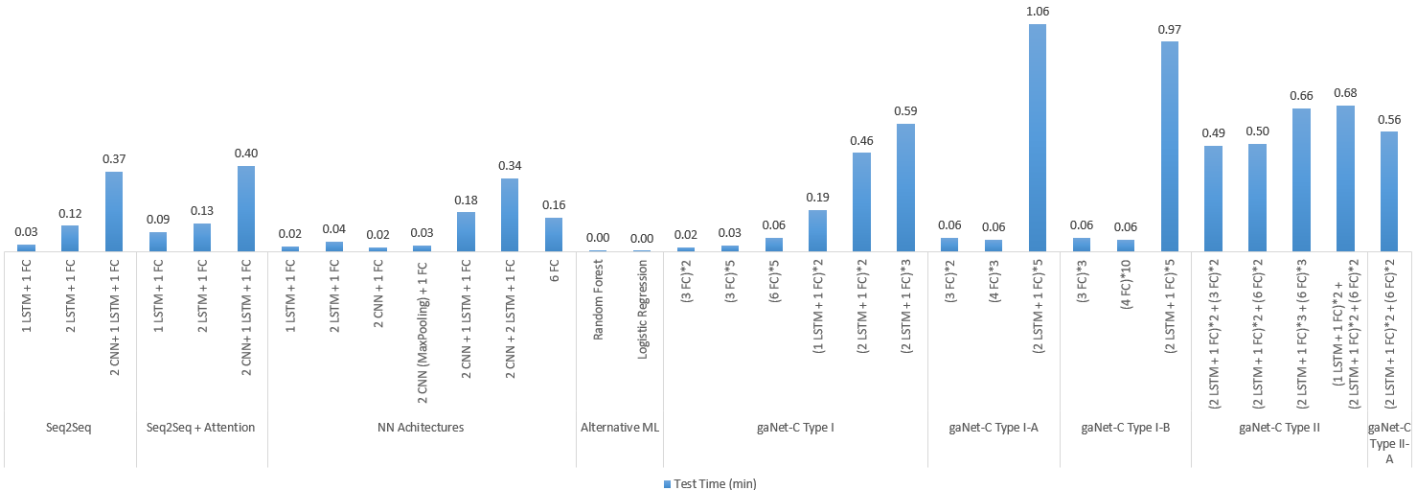

Fig 16. Prediction times (minutes) for all algorithms for classification of elephant connections

4.3 Time-ahead predictions details

The evolution of prediction performance vs prediction time distance (number of time-ahead prediction periods) is usually not a monotonic decreasing linear function, but a more complex curve as shown in detail in Fig 17 and 18, which provide this evolution for different models and for the two problems posed in this work: the prediction of active connections (Fig 17) and elephant connections (Fig 18).

In both cases, the performance metric used has been the F1-score, which is more appropriate for the case of elephant detection due to the unbalanced nature of the labels to be predicted, but is used in both cases to facilitate comparison between them. The chair-shaped curve followed by prediction performance vs prediction time, which was also observed in gaNet [1], is observed in both Fig 17 and 18, and more clearly in the Seq2Seq models. The prediction metric for the detection of active flows follows quite well, in general, the chair-shaped curve, with the gaNet-C models approaching a linear decrease. The prediction metric for the detection of elephant flows follows a more complex evolution with an abrupt decrease in performance followed by a rebound and a subsequent decrease with a smaller slope. Looking at Fig 17 and 18, we can see that gaNet-C models generally exhibit more robust and better behavior against the inevitable degradation in prediction performance vs. the prediction time distance.

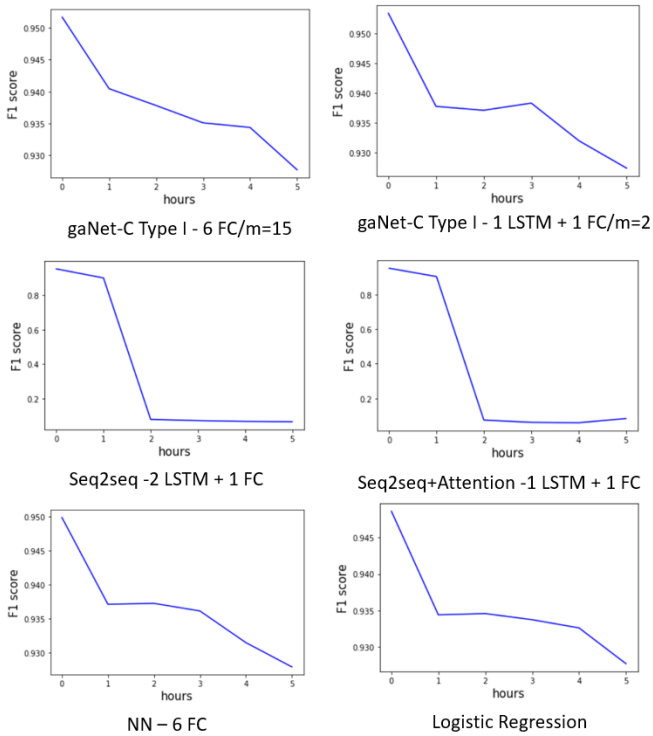

Fig 17. Evolution of $F1-score$ metric vs. time-ahead prediction hours (active vs non-active connections)

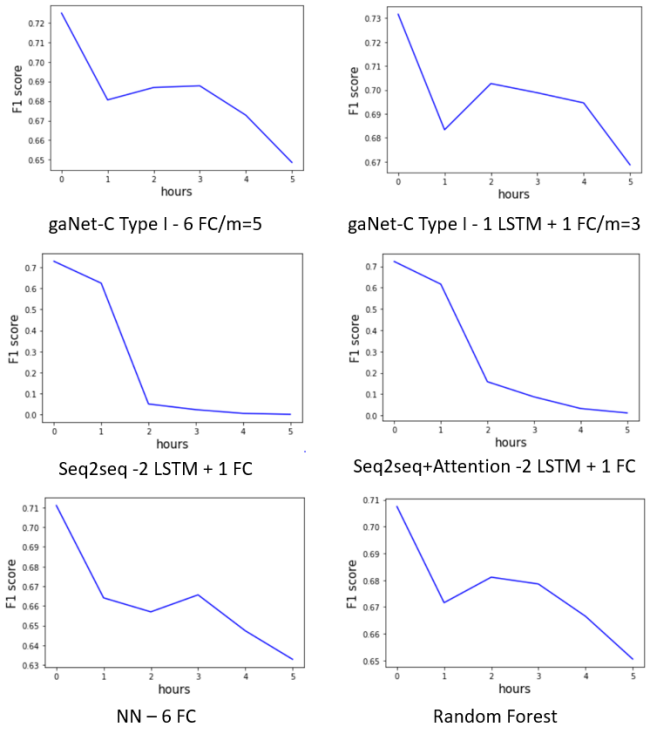

Fig 18. Evolution of $F1-score$ metric vs. time-ahead prediction hours (elephant connections)

4.4 Generic results

To see in one place the forecasting capabilities of the proposed models, in this section all classification performance metrics (accuracy, F1, precision and recall) are presented in detail for the forecast of all k-ahead periods (6 hours in total), including an additional metric: Area Under the Curve (AUC) [47] (Fig 19 and 20). We also present the Receiver Operating Characteristic (ROC) [47] curves for the first and last hour of the forecast, to appreciate the evolution of the metrics depending on the time distance of the forecast (Fig 19 and 20).

Both the AUC metric and the ROC curve offer important alternative metrics for binary classification. ROC curve offers a way to visually assess the trade-off between precision and recall. The more the ROC curve is adjusted to a rectangular shape in the range of values [0,1] for false and true positive rates, the better the classification performance of the classifier. The ROC curve is intended for binary classification and explores how good the classifier is over all possible thresholds used to discriminate between positive and negative outputs. The AUC, which is the area under the ROC curve, has an important interpretation as the probability that the classifier classifies a positive instance chosen at random over a negative instance also chosen at random [47]. Similar to the F1-score, the AUC provides a single metric to evaluate the necessary balance between precision and recall.

Due to the large number of metrics and forecast results, we have to select a few models to show the complete metrics information in a clear manner. That is the reason for choosing only two of the best models for forecasting active and elephant connections. Fig 19 presents the results for the best model (gaNet-C Type I: (6 FC)*5) for the forecast scenario for active connections, while Fig 20 presents similar information for the best model (gaNet-C Type I: (1 LSTM + 1 FC)*3) for the forecast of elephant connections.

We can observe (Fig 19 and 20) that the scenario of forecasting active connections offers better results in all per-hour metrics compared to the scenario of forecasting elephant connections. We also see, that the deterioration in forecast quality, with the time distance to the prediction, is much slower in the former scenario, as shown by the ROC curves and the evolution of the AUC values in Fig 19 and 20. This behavior is due to the unbalanced dataset associated with the elephant connections (section 3.1) which adds significant difficulties to the classification task. Nevertheless, we can see that even with a highly unbalanced dataset the proposed architecture can provide good performance in a k-ahead forecast.

	T0	T1	T2	T3	T4	T5
Accuracy	0.9463	0.9339	0.9317	0.93	0.9286	0.9218
F1	0.9517	0.9405	0.9378	0.9351	0.9344	0.9278
Precision	0.9557	0.9506	0.9489	0.9565	0.9524	0.9569
Recall	0.9477	0.9305	0.927	0.9146	0.9171	0.9003
AUC	0.9868	0.9815	0.9774	0.9781	0.9748	0.969

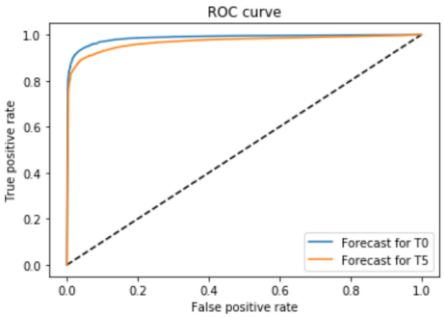

Fig 19. Detailed results for the best model in the forecast scenario of active connections: (Left) Table with all classification performance metrics for each forecast period. (Right) ROC curve for the first (T0) and last (T5) hour forecast.

	T0	T1	T2	T3	T4	T5
Accuracy	0.9431	0.9379	0.9431	0.9432	0.9425	0.9389
F1	0.7317	0.6834	0.7027	0.6988	0.6946	0.6687
Precision	0.8195	0.823	0.8327	0.8305	0.8338	0.834
Recall	0.6608	0.5842	0.6078	0.6032	0.5952	0.5581
AUC	0.9561	0.9496	0.9482	0.9442	0.9315	0.9164

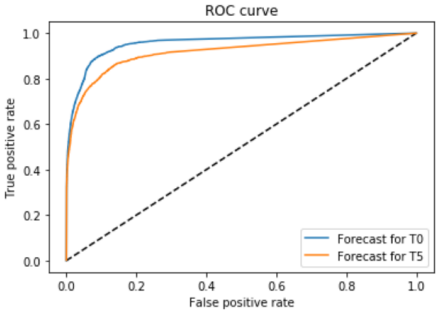

Fig 20. Detailed results for the best model in the forecast scenario of elephant connections: (Left) Table with all classification performance metrics for each forecast period. (Right) ROC curve for the first (T0) and last (T5) hour forecast.

It is important to appreciate the regularization capabilities of the proposed models. This can be seen in the results of Fig 7 and 12, where it can be observed that the inclusion of additional regularization (e.g. drop-out) is not necessary to achieve the best prediction performance, and is even counterproductive. It is interesting to note the large size of some of the architectures presented in Fig 7 and 12. For example, the architecture identified as (6FC)*5 corresponds to 5 additive blocks of 6 layers each, with a total of 30 layers, and, the architecture (4FC)*10 corresponds to a total of 40 layers. The fact that such large architectures can provide good prediction results with an independent test set can be considered as an indication of the capabilities of the model.

It is interesting how gaNet-C Type I provides a good balance between the computational resources needed for training and prediction (Fig 10, 11, 15, 16) and normally requiring no more time than NN models with the same block configuration. The other variants of gaNet-C offer, in general, worse results in terms of training and prediction times. The conclusion from Fig 10, 11, 15 and 16 is that the gaNet-C Type I models based on fully connected layers provide excellent training and prediction times, giving a perfect balance between prediction performance and computational needs.

The architecture of gaNet-C based on an additive integration of similar structures is a perfect candidate for the distribution and parallelism attained by modern decentralized platforms optimized for the deployment of deep learning networks (e.g. Tensorflow) [45]. This characteristic makes gaNet-C a candidate algorithm for network traffic analysis and prediction problems with hard processing requirements (e.g. real-time, large data volumes...). The generic nature of these models makes them applicable in diverse areas that require an accurate forecast of categorical variables, with an additional requirement of small prediction times. Some feasible areas would be: cybersecurity with an emphasis on intrusion detection, manufacturing fault detection, demand change forecasting, traffic change forecasting,…

5. CONCLUSION

The gaNet model [1] is a regression model intended to predict continuous values, in this work we provide an adaptation of this original model to be suitable for classification problems. It is shown that the resulting adaptation (gaNet-C) exhibits excellent

classification capabilities for two difficult and important problems associated with the detection of the state of a network connection. The two classification problems consist of detecting the active state of a connection (connection with some transmitted data) and the state associated with an "elephant connection" (connection that carries most of the traffic). The detection is carried out considering the traffic volume of the connection for consecutive and fixed time-periods (1-hour). In addition, the initial problem of type-of-traffic prediction has been extended to a forecast problem when applied, not to the detection of the current state of the connection, but to detect the k future states associated with the immediate k future time-periods. In this way, we are finally faced with a problem of multi-label classification [48], where the k predicted labels contain the connection states for each of the k future time-periods. The traffic used for training and prediction is real traffic from a Mobile Operator.

We provide a thorough comparison of the results of gaNet-C against many alternative state-of-the-art ML models, for the two classification problems mentioned above performing a k-steps ahead forecasting which is handled as a multi-label classification. The two scenarios (detection of active and elephant connections) present different requirements (balanced vs unbalanced labels distribution) that impose additional difficulties for the models.

The model incorporates many good properties of other models to which it is connected, such as: residual networks, gradient boosting and ensemble models. In particular, the model offers excellent convergence capabilities and built-in regularization properties.

This work presents several variants of the gaNet-C architecture in line with the variants in [1]. All adaptations made for gaNet-C are presented in detail. All the results obtained, with the different gaNet-C variants and the alternative ML models, are presented under different points of view: (1) several performance metrics, (2) training and prediction computational times, and (3) evolution of prediction performance vs the number of time-ahead prediction steps. Considering the results, we show that gaNet-C provides excellent classification capabilities, with better or similar results to the best alternative models. The good results of gaNet-C are maintained for the two classification scenarios considered, while the other models stand out in only one of them. The computational times required for training and prediction are also very limited for gaNet-C, which makes the model suitable to be deployed in classification problems for highly demanding data networks (e.g. IoT networks)

As future lines of research, it would be interesting to apply gaNet-C to other fields, being cybersecurity and intrusion detection of special interest [2]. It would also be interesting to perform additional comparisons of results and explore the application of other techniques to the classification problem presented in this work, such as the techniques already mentioned (section 2): T-S fuzzy delayed neural networks, ANFIS and ELM.

REFERENCES

[1] Lopez-Martin M., Carro B., and Sanchez-Esguevillas A., “Neural network architecture based on gradient boosting for IoT traffic prediction”, Future Generation Computer Systems, vol 100, 2019, pp. 656-673. https://doi.org/10.1016/j.future.2019.05.060

[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] Mignanti S., Giorgio A.D. and Suraci V. “A Model Based RL Admission Control Algorithm for Next Generation Networks”. 2009 Eighth International Conference on Networks, Gosier, 2009, pp. 191-196. doi: 10.1109/ICN.2009.39

[4] Li Y. et al., "Predicting Inter-Data-Center Network Traffic Using Elephant Flow and Sublink Information," in IEEE Transactions on Network and Service Management, vol. 13, no. 4, pp. 782-792, 2016. doi: 10.1109/TNSM.2016.2588500

[5] Poupart P. et al., "Online flow size prediction for improved network routing," 2016 IEEE 24th International Conference on Network Protocols (ICNP), Singapore, 2016, pp. 1-6. doi: 10.1109/ICNP.2016.7785324

[6] Wang B. and Su J., "A survey of elephant flow detection in SDN," 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, 2018, pp. 1-6. doi: 10.1109/ISDFS.2018.8355352

[7] Xiao P. et al., "An efficient elephant flow detection with cost-sensitive in SDN," 2015 1st International Conference on Industrial Networks and Intelligent Systems (INISCom), Tokyo, 2015, pp. 24-28. doi: 10.4108/icst.iniscom.2015.258274

[8] Zhang C. et al., "Deep Learning in Mobile and Wireless Networking: A Survey," in IEEE Communications Surveys & Tutorials. 2019. doi: 10.1109/COMST.2019.2904897

[9] Chen T. and Guestrin C. “XGBoost: A scalable tree boosting system”. arXiv:1603.02754 [cs.LG]

[10] Ke G. et al., “LightGBM: A highly efficient gradient boosting decision tree”. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), pp 3149–3157, 2017.

[11] 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.

[12] 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

[13] 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

[14] 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

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

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

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

[18] 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

[19] 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

[20] Jacobs, P.A. and Lewis, P.A.W. "Stationary discrete autoregressive-moving average time series generated by mixtures". Journal of Time Series Analysis, 4: pp. 19-36. 1983. doi:10.1111/j.1467-9892.1983.tb00354.x

[21] Weiß, C. "Models for Categorical Time Series". In book: An Introduction to Discrete-Valued Time Series, C. Weiß (Ed.). 2018. doi:10.1002/9781119097013.ch7

[22] Chang T.J et al., "Application of Discrete Autoregressive Moving Average models for estimation of daily runoff". Journal of Hydrology. Vol 91, Issues 1–2, 1987. pp. 119-135. https://doi.org/10.1016/0022-1694(87)90132-6

[23] Keenan, D.M. "A Time Series Analysis of Binary Data." Journal of the American Statistical Association 77, no. 380. 1982. pp.816-21. doi:10.2307/2287312.

[24] Kedem B., Fokianos K. "Regression Models for Binary Time Series". In: Modeling Uncertainty. International Series in Operations Research & Management Science, vol 46. Springer, Boston, MA. 2005. https://doi.org/10.1007/0-306-48102-2_9

[25] Harding D. and Pagan A. "An Econometric Analysis of Some Models for Constructed Binary Time Series". Journal of Business & Economic Statistics, 29:1, pp. 86-95. 2011. doi: 10.1198/jbes.2009.08005

[26] Fokianos K. and Kedem B. "Regression Theory for Categorical Time Series." Statistical Science 18, no. 3. 2003. pp. 357-76. http://www.jstor.org/stable/3182755.

[27] Myung-Sup Kim M-S et al., "Characteristic analysis of internet traffic from the perspective of flows". Computer Communications. 29, 10 (June 2006), pp. 1639-1652. DOI=http://dx.doi.org/10.1016/j.comcom.2005.07.015

[28] 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.

[29] 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

[30] Callado A. et al., "A Survey on Internet Traffic Identification and classification" in IEEE Communications Surveys & Tutorials, vol. 11, no. 3, pp. 37-52, 2009. doi: 10.1109/SURV.2009.090304

[31] Nguyen T.T.T and Armitage G., "A survey of techniques for internet traffic classification using machine learning," in IEEE Communications Surveys & Tutorials, vol. 10, no. 4, pp. 56-76, Fourth Quarter 2008. doi: 10.1109/SURV.2008.080406

[32] 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.

[33] Ros-Giralt J. et al., "High Speed Elephant Flow Detection Under Partial Information", 2017. arXiv:1701.01683 [cs.NI]

[34] Basat R.B. et al., "Optimal Elephant Flow Detection", 2017. arXiv:1701.04021 [cs.DS]

[35] Madanapalli S.C. et al., "Real-time detection, isolation and monitoring of elephant flows using commodity SDN system," *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Taipei, 2018, pp. 1-5. doi: 10.1109/NOMS.2018.8406200

[36] Shi K.,, Wang J., Tang Y. and Zhong S., "Reliable asynchronous sampled-data filtering of T–S fuzzy uncertain delayed neural networks with stochastic switched topologies", Fuzzy Sets and Systems, 2018, https://doi.org/10.1016/j.fss.2018.11.017.

[37] Shi K. et al., "Non-fragile memory filtering of T-S fuzzy delayed neural networks based on switched fuzzy sampled-data control", Fuzzy Sets and Systems, 2019, https://doi.org/10.1016/j.fss.2019.09.001.

[38] Zeynoddin M. et al. "Novel hybrid linear stochastic with non-linear extreme learning machine methods for forecasting monthly rainfall a tropical climate". 2018. Journal of Environmental Management.; 222:190-206. doi: 10.1016/j.jenvman.2018.05.072

[39] Zeynoddin M. et al. "A reliable linear stochastic daily soil temperature forecast model", Soil and Tillage Research, 2019, Vol 189, Pages 73-87, https://doi.org/10.1016/j.still.2018.12.023.

[40] Bonakdari, H., Moeeni, H., Ebtehaj, I. et al. "New insights into soil temperature time series modeling: linear or nonlinear

Theoretical and Applied Climatology, 2019, Vol 135, Issue 3–4, pp 1157–1177. https://doi.org/10.1007/s00704-018-2436-2

[41] Yaseen Z.M., et al., "Novel approach for streamflow forecasting using a hybrid ANFIS-FFA model", Journal of Hydrology, 2017, Vol 554, pp 263-276, https://doi.org/10.1016/j.jhydrol.2017.09.007

[42] Ebtehaj I., Bonakdari H. and Gharabaghi B., "A reliable linear method for modeling lake level fluctuations", Journal of Hydrology, 2019, Vol 570, Pages 236-250, https://doi.org/10.1016/j.jhydrol.2019.01.010.

[43] Moeeni, H. and Bonakdari, H. "Impact of Normalization and Input on ARMAX-ANN Model Performance in Suspended Sediment Load Prediction", Water Resources Management, 2018, Vol 32, Issue 3, pp 845–863. https://doi.org/10.1007/s11269-017-1842-z

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

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

[46] Kingma D. and Ba J. "Adam: A Method for Stochastic Optimization", 2014. arXiv:1412.6980 [cs.LG]

[47] Fawcett T. "An introduction to ROC analysis", 2006, Pattern Recognition Letters, Volume 27, Issue 8, pp. 861-874. https://doi.org/10.1016/j.patrec.2005.10.010.

[48] Tsoumakas G. and Katakis I. "Multi-Label Classification: An Overview". International Journal of Data Warehousing and Mining (IJDWM), vol 3, pp. 1-13. 2007. https://doi.org/10.4018/jdwm.2007070101