Universidad de Valladolid

PROGRAMA DE DOCTORADO EN

Tecnologías de la Información y las Telecomunicaciones

Escuela Técnica Superior de Ingenieros de Telecomunicación

Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TESIS DOCTORAL

Novel applications of Machine Learning to Network Traffic Analysis and Prediction

Presentada por Manuel López Martín para optar al grado de

Doctor por la Universidad de Valladolid

Dirigida por:

Dra. Belén Carro

Dr. Antonio Javier Sánchez Esguevillas

Gracias a Estela, mi mujer, y mis hijos Arturo y Aurora por sobrellevarme en los momentos en que he estado ausente en cuerpo y…. mente. Sin su ayuda y comprensión este trabajo no lo podría haber llevado a cabo.

Gracias a mis directores de tesis Belén Carro y Antonio Javier Sanchez Esguevillas por ayudarme y darme soporte, por su motivación y preocupación en la realización de esta tesis. Les quiero agradecer también el haberme ofrecido la oportunidad de colaborar como investigador en la Universidad de Valladolid, lo que ha sido una gran experiencia tanto a nivel profesional como personal.

Quiero dar las gracias a Jaime Lloret por su participación y consejos en varios de los artículos aquí presentados. También quiero dar las gracias al equipo de Jaime Lloret en la Universitat Politècnica de València y a Santiago Egea de la Universidad de Valladolid por el trabajo conjunto realizado en el Proyecto Nacional de Investigación (Ministerio de Economía, Dirección General de Investigación Científica y Técnica): "Distribución inteligente de servicios multimedia utilizando redes cognitivas adaptativas definidas por software", que ha servido como apoyo económico de esta tesis.

Gracias a Telefónica por darme la oportunidad de trabajar como científico de datos en tantos proyectos interesantes, y en particular, gracias a Javier Martínez Elicegui por su dinamización del grupo de ciencia de datos en Telefónica, por su apoyo y su ayuda en el desarrollo del primer artículo de la tesis.

INDICE

I. RESUMEN **4**
II. THESIS **6**
1. RESEARCH OBJECTIVES **6**
2. THESIS FRAMEWORK **10**
3. RESEARCH CONTEXT AND RELATED WORKS REVIEW **12**
3.1 MACHINE LEARNING IN DATA NETWORKS – OVERVIEW 12
3.1.1 Machine learning in data networks *13*
3.1.2 Deep learning in data networks *16*
3.1.3 Generative models in data networks *18*
3.1.4 Machine learning in IoT networks *19*
3.2 SPECIFIC APPLICATION AREAS 21
3.2.1 Intrusion detection *23*
3.2.2 Traffic prediction *29*
3.2.3 Type of traffic prediction (traffic classification) *32*
3.2.4 QoE estimation *35*
3.2.5 Synthetic data generation *37*
4. RESEARCH SCOPE **40**
4.1 COMBINATION OF CONVOLUTIONAL AND RECURRENT NEURAL NETWORKS 43
4.2 GAUSSIAN PROCESSES 47
4.3 CONDITIONAL VARIATIONAL AUTOENCODERS FOR CLASSIFICATION 49
4.4 CONDITIONAL VARIATIONAL AUTOENCODERS FOR PRACTICAL DATA SYNTHESIS 52
4.5 MACHINE LEARNING FOR TIME-SERIES PREDICTION 54
5. CONTRIBUTIONS AND LESSONS LEARNED **56**
5.1 CONTRIBUTIONS 56
5.2 LESSONS LEARNED 59
6. METHODOLOGY **61**
7. PAPERS SUMMARY **67**
7.1 PAPER 1: REVIEW OF METHODS TO PREDICT CONNECTIVITY OF IOT WIRELESS DEVICES 67
7.1.1 Objectives *67*
7.1.2 Datasets *67*
7.1.3 Models *67*
7.1.4 Results/Conclusions *68*
7.2 PAPER 2: NETWORK TRAFFIC CLASSIFIER WITH CONVOLUTIONAL AND RECURRENT NEURAL NETWORKS FOR INTERNET OF THINGS 69
7.2.1 Objectives *69*
7.2.2 Datasets *69*
7.2.3 Models *69*

7.2.4 Results/Conclusions *70*
7.3 PAPER 3: CONDITIONAL VARIATIONAL AUTOENCODER FOR PREDICTION AND FEATURE RECOVERY APPLIED TO INTRUSION DETECTION IN IOT 71
7.3.1 Objectives *71*
7.3.2 Datasets *71*
7.3.3 Models *72*
7.3.4 Results/Conclusions *72*
7.4 PAPER 4: DEEP LEARNING MODEL FOR MULTIMEDIA QUALITY OF EXPERIENCE PREDICTION BASED ON NETWORK FLOW PACKETS 73
7.4.1 Objectives *73*
7.4.2 Datasets *73*
7.4.3 Models *74*
7.4.4 Results/Conclusions *74*
7.5 PAPER 5: VARIATIONAL DATA GENERATIVE MODEL FOR INTRUSION DETECTION 76
7.5.1 Objectives *76*
7.5.2 Datasets *77*
7.5.3 Models *77*
7.5.4 Results/Conclusions *78*
8. TOOLS 79
9. GENERAL CONCLUSIONS AND SUMMARY OF CONTRIBUTIONS 80
10. FUTURE LINES OF RESEARCH 83
11. RESEARCH DISSEMINATION PLAN 85
12. LIST OF REFERENCES 86
III. PAPERS 95
PAPER 1 95
PAPER 2 111
PAPER 3 128
PAPER 4 147
PAPER 5 161
POSTER MLSS-2018 182

I. RESUMEN

El objetivo de esta tesis es el presentar la aplicación de técnicas novedosas de aprendizaje automático (ML-Machine Learning) en el campo de la Telecomunicaciones, y en particular a problemáticas relacionadas con el análisis y la predicción de tráfico en redes de datos (NTAP – Network Traffic Analysis and Prediction).

Las aplicaciones de NTAP son muy amplias, por lo que esta Tesis se focaliza en las siguientes cinco áreas específicas:

- Predicción de la conectividad de dispositivos wireless.
- Clasificación de tráfico de red, utilizando las cabeceras de los paquetes transmitidos
- Detección de intrusiones de seguridad, utilizando información de tráfico de red
- Generación de tráfico sintético asociado a ataques de seguridad y utilización de dicho tráfico sintético para mejorar los algoritmos de detección de intrusiones de seguridad.
- Estimación de la calidad de la experiencia percibida por el usuario (QoE) al visualizar secuencias de video, utilizando información agregada de los paquetes transmitidos

La intención última es crear modelos de predicción y análisis que supongan mejoras en las áreas de NTAP arriba mencionadas. Para ello, en esta Tesis se plantean avances en la aplicación de técnicas de aprendizaje automático al área de NTAP. Estos avances consisten en:

- Desarrollo de nuevos modelos de aprendizaje automático específicos para NTAP
- Especificar nuevas formas de estructurar y transformar los datos de entrenamiento para que los modelos de aprendizaje automático existentes se puedan aplicar a problemas específicos de NTAP.
- Definir algoritmos para la creación de tráfico de red sintético que corresponda con eventos específicos en la operativa de la red (p. ej. tipos específicos de intrusiones), asegurando que los nuevos datos sintéticos puedan ser usados como nuevos datos de entrenamiento.
- Extensión y aplicación de modelos clásicos de aprendizaje automático al área de NTAP, obteniendo mejoras en las métricas de clasificación o regresión, y/o mejoras en las medidas de rendimiento de los algoritmos (p. ej. tiempo de entrenamiento, tiempo de predicción, necesidades de memoria, …)

En esta Tesis se han aplicado tanto las técnicas más conocidas de aprendizaje automático (p.ej. regresión logística, arboles aleatorios, máquinas de vector soporte...) como las nuevas técnicas de aprendizaje profundo (DL – Deep Learning). En los últimos años, han aumentado la variedad y éxito en la aplicación de las técnicas de aprendizaje profundo. Las técnicas de aprendizaje profundo son un área específica de las técnicas de aprendizaje automático, caracterizándose por incluir redes neuronales con varias capas y arquitecturas y conectividad diversa. Los algoritmos relacionados con el aprendizaje profundo se han aplicado ampliamente en las áreas de: procesamiento de imágenes y video, audio, tratamiento de textos, comprensión y traducción del lenguaje natural, finanzas, medicina, ventas. En esta Tesis veremos cómo su aplicación se puede extender a las problemáticas asociadas con NTAP.

Para la realización de la tesis se ha optado por el formato de compendio de artículos publicados en revistas indexadas JCR. Se han publicado cinco artículos. Esta tesis se centra solo en los

artículos publicados. Todos los artículos utilizan técnicas relacionadas (aprendizaje automático y aprendizaje profundo), unas problemáticas conectadas (NTAP), con objetivos comunes (detección y predicción) y que giran alrededor de un campo de actuación común (redes de datos). Los artículos en conjunto forman una línea de trabajo coherente, dirigida a la aplicación de técnicas avanzadas de inteligencia artificial a la resolución de problemas complejos de análisis y predicción planteados en nuevas arquitecturas de redes de datos.

II. THESIS

1. RESEARCH OBJECTIVES

It is now clear that machine learning will be widely used in future telecommunication networks as it is increasingly used in today's networks. However, despite its increasing application and its enormous potential, there are still many areas in which the new techniques developed in the area of machine learning are not yet fully utilized.

The aim of this thesis is to present the application of innovative techniques of machine learning (ML-Machine Learning) in the field of Telecommunications, and specifically to problems related to the analysis and prediction of traffic in data networks (NTAP - Network Traffic Analysis and Prediction).

The applications of NTAP are very broad, so this thesis focuses on the following five specific areas:

- Prediction of connectivity of wireless devices.
- Security intrusion detection, using network traffic information
- Classification of network traffic, using the headers of the transmitted network packets
- Estimation of the quality of the experience perceived by the user (QoE) when viewing multimedia streaming, using aggregate information of the network packets
- Generation of synthetic traffic associated with security attacks and use of that synthetic traffic to improve security intrusion detection algorithms.

The final intention is to create prediction and analysis models that produce improvements in the NTAP areas mentioned above. With this objective, this thesis provides advances in the application of machine learning techniques to the area of NTAP. These advances consist of:

- Development of new machine learning models and architectures for NTAP
- Define new ways to structure and transform training data so that existing machine learning models can be applied to specific NTAP problems.
- Define algorithms for the creation of synthetic network traffic associated with specific events in the operation of the network (for example, specific types of intrusions), ensuring that the new synthetic data can be used as new training data.
- Extension and application of classic models of machine learning to the area of NTAP, obtaining improvements in the classification or regression metrics and/or improvements in the performance measures of the algorithms (e.g. training time, prediction time, memory needs, ...)

We have applied more classical machine learning techniques (e.g. logistic regression, random forest, support vector machines ...) as well as new deep learning techniques (DL - Deep Learning). In recent years, the variety and success in the application of deep learning techniques has increased. The deep learning techniques are a specific area of machine learning,

which is characterized by including neural networks with multiple layers and a variety of architectures and connectivity between layers. The algorithms related to deep learning have been widely applied in the areas of: image and video processing, audio, word processing, comprehension and translation of natural language, finance, medicine, sales, etc... One of the main objectives of this thesis is to show how its application can be extended to the problems associated with NTAP.

Considering the five specific areas of NTAP that are the subject of this thesis, four correspond to classification problems and one with the problem of generating synthetic data that can be used to improve a classification problem.

The four areas related with classification pose many challenges to a classification and detection algorithm: 1) highly unbalanced data with labels strongly biased to some of the classes, 2) noisy data and 3) high cardinality of the labels to classify. For these reasons, different types of deep learning algorithms have been explored: generative algorithms (variational autoencoders) and prediction algorithms based on convolutional and recurrent neural networks, considering that these algorithms have shown remarkable results in other business areas. In this thesis we show that these algorithms are applicable to NTAP and the work in this thesis contributes to provide novel architectures based on them. Specifically, it is important to mention the contributions made in this thesis to the areas of: 1) estimation (both detection and prediction) of the quality of a user's experience when viewing multimedia streaming and 2) network traffic classification based on new architectures formed by convolutional neural networks (CNN) and recurrent neural networks (RNN).

To generate synthetic data there are many over-sampling algorithms that generate the synthetic data corresponding to a specific class based on the (topological) proximity to existing data of that class. These algorithms need a predefined (often complex) distance definition. In this thesis, an alternative method for the creation of synthetic data is proposed, which is based on a latent probability distribution that is learned from the data and that does not need to assume a predefined distance function. The proposed method consists of a generative model based on a variational autoencoder, with an architecture adapted to the generation of synthetic data associated with specific events. The new method offers operational advantages (speed, simplicity) and quality in the synthesized data (similar probability distributions and improvements in their properties) compared to the usual algorithms.

In addition to the novel architectures based on deep learning algorithms, we propose also advances in the application of machine learning models to the problem of network traffic prediction. In this case, due to the time series nature of network traffic, the techniques usually applied have been methods related to the solution of time series prediction problems (e.g. ARIMA, ARIMAX ...). This thesis presents the suitability of alternative machine learning techniques for time series prediction applied to network traffic, as well as a detailed comparison of both options: a) the methods based on classical time series techniques and b) of those based on machine learning. In this case the critical point is the necessary transformation of the training dataset from a time series structure (longitudinal-like data) to a supervised learning structure (matrix-like data).

An important type of data network is related to IoT (Internet of Things) devices. This type of network imposes some new and difficult requirements due to the large number of associated devices of heterogeneous nature and with very different connectivity and service characteristics. Therefore, IoT networks are a good place to test new machine learning models, to assess whether they can provide an improvement in their performance, manageability and/or security. This is the reason why, even when the results obtained in this thesis are applicable to any type of data networks, the IoT networks have been the focus of many of the experiments carried out in the thesis.

The modality of this thesis is the compendium of publications in JCR-indexed journals in the telecommunications and data networking field, with a total of five papers published. This thesis focuses only on the published papers. All the papers use related techniques (machine learning and deep learning), some related problems (NTAP), with common objectives (detection and prediction) and that revolve around a common field of action (data networks). The papers together form a coherent line of work, aimed at the application of advanced techniques of machine learning to the resolution of complex problems of analysis and prediction raised in new data network architectures.

The first paper [1] of this compendium: "Review of methods to predict connectivity of IoT wireless devices", focuses on the prediction of activity of wireless devices using different machine learning techniques and classical techniques for time-series prediction, unifying both types of techniques in a common framework and providing a comparative analysis between them and the behaviour of the connected devices.

The second paper [2]: "Network traffic classifier with convolutional and recurrent networks for Internet of Things", studies the application of deep learning models based on convolutional and recurrent neural networks to the prediction of the type of service of a network flow, using exclusively information of the headers of the network flow packets.

The third paper [3]: "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT", investigates the use of generative models based on variants of variational autoencoders to the detection of intrusions in data networks, as well as for the synthesis of features associated with different types of intrusions.

The fourth paper [4]: “Deep learning model for multimedia Quality of Experience prediction based on network flow packets”, proposes a new classifier to perform QoE estimation of multimedia content transmitted by a data network. It is based on a deep learning model (convolutional and recurrent networks) plus a Gaussian process as the final layer. The resulting classifier can perform QoE detection (current time) and prediction (short-term forecast) using exclusively aggregated information extracted from the network packets.

The fifth paper [5]: "Variational data generative model for intrusion detection ", presents the possibility of generating synthetic traffic data (with both discrete and continuous features), where the synthetic data can be conditioned to different types of intrusions (security attacks). In this way, the probability distribution of the features for the synthetic traffic follows the distribution of the real features for each type of intrusion. Moreover, we show that the synthetic traffic can be used as new training data, improving the detection results of well-known classifiers.

Fig. 1 presents an overview of the framework used to position the different objectives and points of interest of this thesis. We can see that the goal is to explore prediction and detection algorithms for data networks in the five areas mentioned above, with a focus on prediction/detection problems as well as to investigate the generation of synthetic data that follows the probability distribution of features associated with intrusion detection datasets (making use of generative methods).

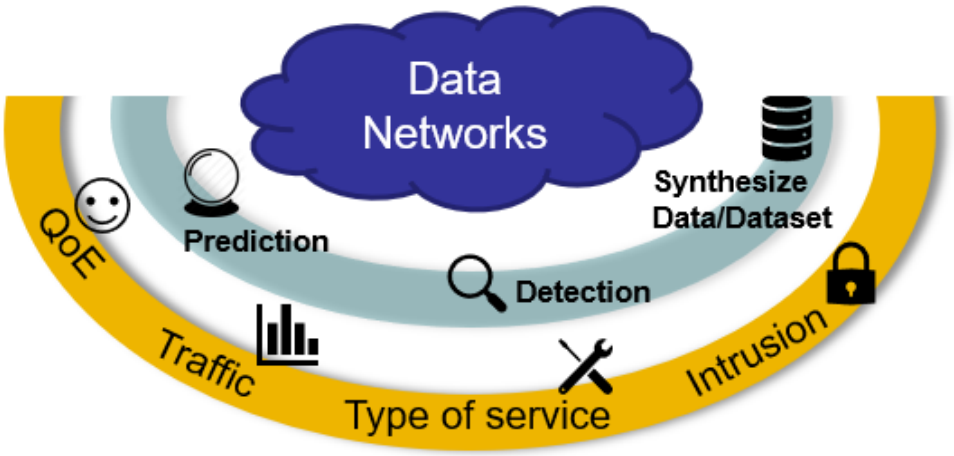


Fig. 1. Framework for the research objectives of this thesis

2. THESIS FRAMEWORK

The modality of this thesis is the compendium of publications in JCR-indexed journals in the telecommunications and data networking field.

In Fig. 1 the objectives and scope of the thesis are presented. In Fig. 2 more details on how the different papers fit into the different areas of study are shown. Each paper fits in a different area.

The first paper: "Review of methods to predict connectivity of IoT wireless devices" has been published in Ad Hoc & Sensor Wireless Networks.

The second paper: "Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things" has been published in IEEEAccess in the Special Section on Big Data Analytics in Internet of Things and Cyber-physical Systems.

The third paper: "Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT" has been published in Sensors.

The fourth paper: "Deep learning model for multimedia Quality of Experience prediction based on network flow packets" has been published in IEEE Communications Magazine with the feature topic of: "Enabling Technologies for Smart Internet of Things".

Finally, the fifth paper: "Variational data generative model for intrusion detection" has been published in Knowledge and Information Systems.

Paper 1 [1] provides a thorough study of the activity of real IoT devices, which parameters affect it and how different algorithms can be used to predict it. The paper proposes a dataset preparation which fits classic time-series algorithms (ARIMA, HMM...) as well as other machine learning algorithms (non-time-series based: Random Forest, Logistic Regression...) which are not usually applied to time-series problems. In particular, as far as we know, it is the first time that a Random Forest algorithm is presented in the literature to deal with a time-series problem as the one treated in the paper.

Paper 2 [2] studies the application of deep learning algorithms to detect the type-of-service of a network flow. It provides a comparison of results with other machine learning methods. It is, as far as we know, the first application of a CNN + RNN model to an NTC problem.

Paper 3 [3] proposes a new solution to intrusion detection in data networks. The proposal is based on a generative model using a variant of a Variational Autoencoder (VAE) whose name is Conditional Variational Autoencoder (CVAE). This variant of a VAE provides many advantages and it is not only able to give better prediction results than other machine learning algorithms, but it is also able to generate new synthetic samples with feature values that have a

probability distribution according to specific types of intrusions. This is, as far as we know, the first application of a CVAE to the intrusion detection field.

Paper 4 [4] presents a new solution based in deep learning networks (a combination of CNN and RNN) to perform QoE detection and short-term prediction of seven QoE anomalies/errors. This is, as far as we know, the first application of a CNN+RNN model to video QoE estimation. In order to perform the estimates, the model uses a small time-series of vectors formed by aggregated information extracted from network packets.

Paper 5 [5] proposes the application of a CVAE to generate synthetic data in networking. The architecture of the model is specifically tuned to generate synthetic data (network traffic data) that can be used by an intrusion detection classifier to improve its training performance.

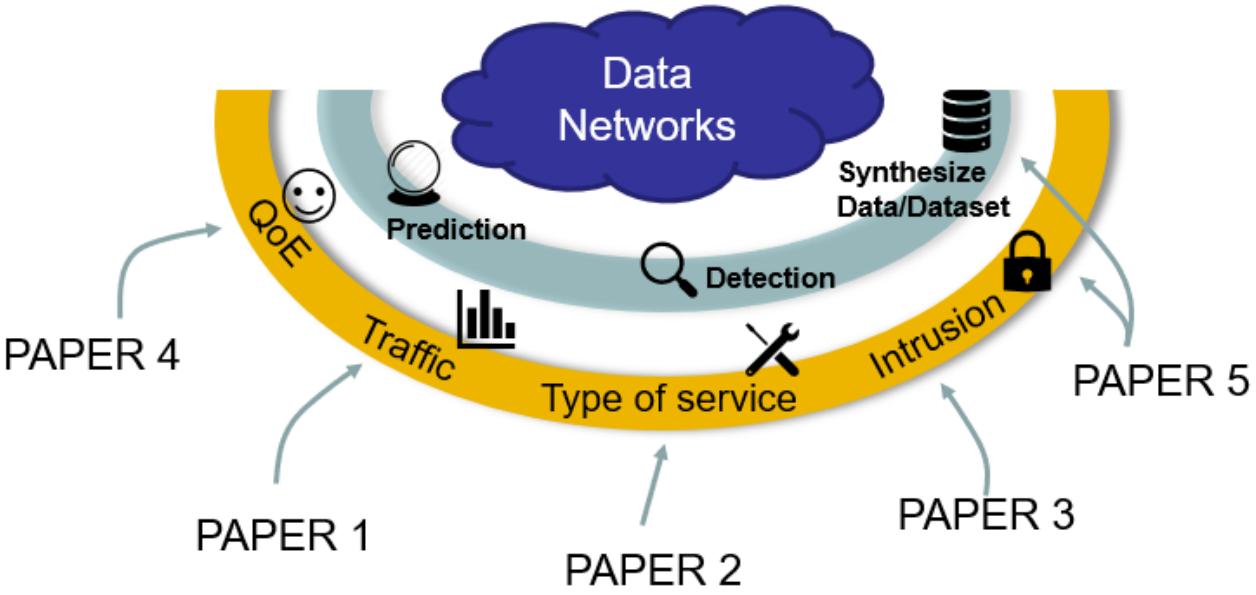


Fig. 2. Relationship between research objectives and papers

3. RESEARCH CONTEXT AND RELATED WORKS REVIEW

The following sections present the research context for this thesis, in two aspects: (1) a general overview of the application of machine learning to data networks and (2) a more specific review of the application of machine learning to the areas of prediction and data generation covered by this thesis.

3.1 Machine learning in data networks – Overview

Machine learning [6][7][8] has been extensively applied to data networking problems. This application was done in many cases under different names inside well-known scientific disciplines as signal processing, information theory, coding theory, etc… examples of this can be observed in the application of linear and non-linear regression, statistical models, compression methods, etc…This applications have been usually limited to classic machine learning techniques, however, many recent advances in machine learning are not applied as fully in networking as in other areas (e.g. image, speech and video processing, natural language processing,…). This can be seen in the most updated reviews on the application of machine learning to networking [6][7][8][9], where the reduced application of the latest models in machine learning is pointed out (e.g. deep learning models, generative models), despite being considered important algorithms with a critical contribution in future research. For example, quoting [6] on the importance of machine learning in the future of data networks:

"……
The latest breakthroughs, including deep learning (DL), transfer learning and generative adversarial networks (GAN), also provide potential research and application directions in an unimaginable fashion.
Dealing with complex problems is one of the most important advantages of machine learning. For some tasks requiring classification, regression and decision making, machine learning may perform close to or even better than human beings. Some examples are facial recognition and game artificial intelligence. Since the network field often sees complex problems that demand efficient solutions, it is promising to bring machine learning algorithms into the network domain to leverage the powerful ML abilities for higher network performance. The incorporation of machine learning into network design and management also provides the possibility of generating new network applications. Actually, ML techniques have been used in the network field for a long time. However, existing studies are limited to the use of traditional ML attributes, such as prediction and classification. The recent development of infrastructures (e.g., computational devices like GPU and TPU, ML libraries like Tensorflow and Scikit-Learn) and distributed data processing frameworks (e.g., Hadoop and Spark) provides a good opportunity to unleash the magic power of machine learning for pursuing the new potential in network systems.
……."

Considering these opportunities and envisaged future contributions [10], the objective of this thesis is to provide advances in the application of the latest machine learning models to networking.

3.1.1 Machine learning in data networks

Machine learning algorithms can learn directly from data without an explicit prior programming task. Machine learning with its ability to learn from data is especially suitable for problems that are too complex to be fully defined or whose definition cannot be made accurately. These are, precisely, the kind of problems that arise in networking.

Many machine learning algorithms can be applied to the diverse problems that originate in data networks, such as: Random Forest [11], Gradient Boosting Machine (GBM) [12], Support Vector Machine (SVM) [13], Logistic Regression, Multinomial Logistic Regression [14], Multilayer Perceptron (MLP) [15], K-Nearest Neighbors (KNN) [14], Principal Component Analysis [14], K-Means [14], Naïve Bayes [14], and many more [7]. These algorithms can be subdivided into four main groups:

- **Supervised**: We start with some training data that contains the ground-truth values (responses) that we will then want to predict [14][16].
    - Regression: The value to predict is continuous
    - Classification: The value to predict is discrete/categorical

  The challenge is to generalize the prediction to test data for which the predicted correct value is not known.

- **Unsupervised**: We start from some training data without values to predict. The goal is to find structure/relationships in the data, without prior information. It can be further subdivided in clustering, dimensionality reduction (latent variables discovery), anomaly/outliers detection, probability density function learning… [14][16].

  The challenge is to find the latent structure of the data.

- **Semi-supervised**: We start with some training data that contains the correct responses plus additional training data without them [17]. The objective is similar to the supervised scenario: to predict the correct value for new test data.

  The challenge is to generalize the prediction to new test data (as in the supervised scenario) while making full use of the additional information provided by the un-labeled data.

- **Reinforcement learning**: The objective is to learn the best sequence of actions that optimize the expected sum of future rewards [18] while interacting with a usually unknown environment that generates the rewards and state transitions that define the dynamics of the interaction.

  The challenge is to learn a policy function that generates the best current action considering an uncorrelated and possibly sparse sequence of rewards without knowing the dynamics of the environment.

An important subset of algorithms employing a cascade of interconnected layers of (usually) non-linear nodes corresponds to the so-called: deep learning algorithms [19]. Deep learning algorithms are used in supervised, unsupervised, semi-supervised and reinforcement learning models. They are usually extremely good in representation learning which is a real advantage to avoid difficult and costly feature engineering. The training is usually done by optimizing a cost function using some form of gradient descent.

The main deep learning algorithms considered in this thesis are: Convolutional Neural Networks (CNN) [20], Recurrent Neural Network (RNN)[21], Long Short-Term Memory

Network (LSTM) [22], Variational Autoencoders (VAE) [23] and Conditional Variational Autoencoders (C-VAE) [24][25]. The following section (section 3.1.2) is dedicated to presenting these models in more detail.

This thesis is mainly focused in **supervised algorithms for classification** and precisely in:

- Classic ML:
  - Random Forest, GBM, SVM, Multinomial Logistic Regression, Gaussian Process and MLP.
- Deep Learning:
  - CNN, LSTM and combinations of both.
  - Generative models: VAE and variants.

In addition to the machine learning models mentioned above, and belonging to the same family of predictive algorithms, we have considered some algorithms that are normally used to make predictions for time-series data. These algorithms are not usually studied as machine learning methods, but rather in the area of statistics and time-series analysis. These algorithms have been the usual resort to make predictions when dealing with time-series data, and we have employed them as part of the methods analyzed to predict future traffic in IoT networks. The time-series algorithms considered are: Hidden Markov Model (HMM) [26], Exponential Smoothing [27], AutoRegressive Integrated Moving-Average (ARIMA) [28] and ARIMA with eXogenous covariates (ARIMAX) [28].

The authors in [6][8]provide two excellent and updated reviews on the application of machine learning in networking. In [6] the application areas of ML to networking are structured as: information cognition (route measurement), traffic prediction, traffic classification, resource management, network adaption (QoE optimization, congestion control, routing strategy), performance prediction (QoE and throughput prediction) and configuration extrapolation (automated network protocol and architecture design). The authors consider that all areas will be impacted by ML with a greater future impact on: resource management, network adaption and automated architecture design.

In [8] similar (but not identical) application areas of ML to networking are identified: traffic prediction, traffic classification, traffic routing, congestion control, resource management, fault management, QoS/QoE management and network security. Traffic prediction is mainly dominated by time-series algorithms (ARIMA, HMM…) with some few applications of non-time series methods based on: MLP, CNN/RNN and SVM. Traffic classification has employed supervised (K-NN, SVM, Adaboost, XGBoost, Decision Trees...) and unsupervised (K-Means, DBSCAN) methods with no reported deep learning works in the study. Traffic routing is driven by reinforcement learning algorithms, mainly Deep Q-Learning. Congestion control is managed with packet loss classification algorithms using classic supervised models (SVM, K-NN, MLP…) and Queue management is the territory of queue length predictors (ordinary linear regression and time-series models) and adaptive mechanisms based on reinforcement learning plus control theory. ML for Resource management is mainly focused on: a) admission control and b) resource allocation; applying prediction and decision-making algorithms (MLP and Q-learning). Fault management is broken down into: predicting the fault, detecting the fault, localizing the root cause of the fault and automated mitigation of the fault;

employing a full range of prediction/classification algorithms (MLP, RNN, decision trees, ensembles...) with a focus on feature selection techniques and anomaly detection based on unsupervised models (K-means, SOM...). QoS/QoE management can use ML for both prediction and adaptation, although all research focuses on prediction using most prediction/classification algorithms (SVM, Random Forest, MLP, K-NN, Naïve Bayes…) without reported works using deep learning. Network security is an intensive area of research where the majority of ML classification algorithms are applied, but in this case, there are more works that employ deep learning algorithms, mainly Deep Belief Networks (DBN), with also many unsupervised learning algorithms used for anomaly detection problems (SOM, one-class SVM…). It is also interesting the three most important points that are mentioned as problematics for a stronger adoption of ML in networking: a) lack of real-world data, b) the need for standard evaluation metrics and c) specific theory and ML techniques for networking (since most of the techniques are developed for other fields).

3.1.2 Deep learning in data networks

Deep learning [19] is a sub-field of machine learning based in neural networks with (usually) many layers. Originally the network was a simple feed-forward neural network with the later addition of new types of networks: CNN, RNN, LSTM, VAE, C-VAE… [6]

A CNN [20] is a specialized feed-forward neural network originally used in image processing but increasingly employed in many other fields. This type of network applies a collection of filters to automatically extract features from the image, creating finally a hierarchical structure of features (representation learning). The weights of the filters are learned directly from the training data. Normally, a CNN incorporates many convolutional layers creating a deep structure. A CNN performs feature engineering automatically, avoiding the lengthy and cumbersome step of doing feature engineering manually.

RNN [21] was initially applied to Natural Language Processing, but similarly to CNN it is currently incorporated to other fields. The main application of RNN is to sequential data with temporal dependencies. An RNN is able to process new data based on previous data. An important problem of RNN has been its difficulty to be trained with long time-dependent data (long time-series). In order to solve this problem a series of RNN variants have been created. The LSTM [22] network is one of these variants, being the most widely used.

An Autoencoder is a type of feed-forward neural network with at least a hidden layer having a dimension smaller than the input and output dimensions. The training of these networks is done using the same samples for the network input and output. The intention of the network is to learn the identity function. From this apparently meaningless operation, we obtain a dimensionality reduction of the input data (the values of the hidden low dimensional layer). This interesting idea is further extended in a VAE, which is based in similar principles where the internal deterministic low dimensional layer is substituted by random values generated according to a parameterized probability distribution. The parameters of this distribution are the values of a previous internal layer. The initial complexity to train such a network is solved using variational principles and stochastic gradient descent [23]. A VAE is incredibly useful to generate synthetic data similar to the data used for training. This synthetic data has the important property of being stochastic in nature but following the same probability distribution of the original data.

Currently, the area of deep learning is probably one of the most active areas of research in machine learning. The ever-increasing trend to apply deep learning to all areas that could require prediction, classification or patterns analysis is not an exception to the data networking field. Nevertheless, this field is not as active as others (medicine, finance, robotics, marketing, media…) in adopting this new technology.

Authors in [9] present a complete review of deep learning for networking considering also future directions. Of all applications considered: wireless sensor networks, network traffic classification, network flow prediction, social networks, mobility prediction, cognitive radio, self-organized networks and routing, the last four are the focus of more work in deep learning, mainly using Deep Belief Networks, RNN, CNN and MLP with several layers. Nevertheless, the inclusion of deep learning in networking remains scarce, as mentioned in [9], quoting: "*While deep learning has received a significant research attention in a number of other domains such as computer vision, speech recognition, robotics, and so forth, its applications in network traffic control systems are relatively recent and garnered rather little attention*.".

Another recent and comprehensive review of deep learning for mobile and wireless networking is provided in [29]. In addition to a detailed review of current works related to deep learning applied to networking, the section on future research perspectives is especially interesting, as it points out the promising areas of future research: (a) Deep Learning for Spatio-Temporal Data Mining, (b) Deep learning for Geometric Data Mining, (c) Deep Unsupervised Learning and (d) Deep Reinforcement Learning for Network Control. Additionally, the lack and difficulty of accessing data sets related to network activity is mentioned as one of the most serious problems in the application of deep learning to networking. This is due to privacy concerns of operators and users, which are completely reasonable, but nevertheless hamper the development and application of deep learning in this area.

In particular, for the application of deep learning to intrusion detection, in [30] is provided an interesting review and taxonomy of deep learning algorithms in this area. They differentiate between the discriminative (e.g. CNN) and generative models (e.g. VAE, Boltzmann Machines, …), emphasizing the importance of autoencoders (especially stacked autoencoders) as feature extractors, which in many cases serve as the first stage to perform the classification of intrusions.

It is also important to appreciate the difficulties that deep learning can have in a strictly regulated area such as networking due to its difficulty to provide an interpretation of results. For example, European Union's General Data Protection Regulation require such an interpretability of the results when the ML model is used to make decisions without human intervention. This problem has produced an interesting research activity to facilitate the interpretation of the results provided by a deep learning network. This "black box" problem of deep learning models is related to the problem of adversarial inputs (slightly modified inputs that cause an intentional change in the results) and the need to provide credibility to the decisions made by the network. In this line, the work in [31] analyzes this problem and presents a possible solution to estimate the nonconformity (and interpretability) of results, by finding a subset of training samples similar in cosine distance to the results produced by all layers of the network; they apply the k-Nearest Neighbors algorithm to identify the subset of similar inputs which are the basis for facilitating a later interpretation of the results.

3.1.3 Generative models in data networks

The generation of synthetic data is usually made with simulation or generative techniques. With a simulation we try to create an imitation of the real environment that produces the original data. The generation of synthetic data with simulations [32] is not considered in this thesis. The second method to create synthetic data is to use a generative model. A generative model learns the probability distribution of the data. This allows making predictions but, in addition, it also enables to generate new data and to impute missing data (as it is done in [3] as part of the thesis), this is because a generative model allows taking samples of the probability distribution of the data.

We can consider two types of models to accomplish prediction in a supervised scenario: discriminative and generative models. In a supervised setting we have a set of features (X) and a set of labels or values (Y) to predict. In a discriminative model we try to learn directly $P(Y/X)$ and prediction is made by choosing the value of Y that maximises it. In this case, the intention is to learn the decision boundary between values/classes. In a generative model we try to learn $P(X, Y)$ and prediction is made by choosing the value of Y that maximises $P(X, Y)$ given an X. The intention is now to learn the probability distributions of the classes and of the features conditioned on the classes, or alternatively, the joint probability distribution of both features and classes. That is, in the discriminative case we try to find: $arg_Y max(P(Y/X))$. Meanwhile, in the generative case we try to find: $arg_Y max\left(P\left(X/Y\right) * P(Y)\right) = arg_Y max(P(X,Y))$

An important consequence is that once $P(X, Y)$ is known, we can take samples of this joint probability distribution. This is the mechanism to generate new data which is like real one or to perform data augmentation or imputation of missing values of features.

When a generative model is combined with a dimensionality reduction algorithm, we can have an interesting outcome that allows obtaining a sparse latent representation of the data. This is achieved, for example, by means of a Variational Autoencoder (VAE). In [3][5], we use a VAE to obtain all these nice properties and, at the same time, it is much easier to train than other alternative methods (e.g. stacked autoencoders). Generative models can be also used for classification [3] when labelled data is applied to a trained generative model.

In the area of networking, there are some examples of applications of generative models, being the most used algorithms: Deep Belief Networks, Naïve Bayes and Variational Autoencoders. Authors in [33] provide a review of the current state-of-the-art algorithms for generative models, among which are: Variational Autoencoders and Generative Adversarial Networks.

3.1.4 Machine learning in IoT networks

Even when the machine learning architectures and techniques presented in this thesis are applicable to generic traffic for any data network, the experiments to prove their effectiveness has been mainly done with traffic associated with IoT networks. IoT traffic poses a challenge to current network management and monitoring systems, due to the large number and heterogeneity of the connected devices. The difficulties created by the network traffic in IoT networks have been the reason to choose these networks because the solutions provided are more demanding.

The Internet of Things (IoT) has been defined in Recommendation ITU-T Y.2060 (06/2012) as a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies. It is a network of physical devices embedded in all kind of equipments that autonomously transfer information and operational commands between them or with some centralized system.

A complete review of machine learning algorithms applied to IoT problems is presented in [7]. Intrusion detection [34], traffic prediction [35][36][37], characterization and classification of traffic [38], and estimation of video QoE [39][40][41][42], are critical issues in IoT networks. Below is a brief analysis of the importance of NTAP for IoT in the specific areas considered in this thesis:

- Considering **traffic prediction**, from the point of view of an IoT Service Operator (namely a telecommunications operator) it is extremely useful to know in advance the probability distribution of wireless devices connectivity. It is important to anticipate the likelihood that a wireless device will send information over a certain time period, in order to: 1) anticipate the business impact, 2) accommodate the maintenance activity periods to reduce the impact of possible connectivity interruptions, 3) prepare the infrastructure required to reduce the risk of interruption of highly important services and associated devices. In the first work of this thesis [1] is provided a detailed study of the different prediction models available for time-series data and how to modify the training dataset to apply classical machine learning models (random forest, logistic regression, etc…) for time-series prediction. In this work we show that, with the proposed data pre-processing, we can obtain excellent results with a logistic regression or random forest models, which are almost as good as the results obtained with the best time-series model (ARIMAX), but, with an important reduction on the required processing time.
- In relation to **intrusion detection**, a Network Intrusion Detection System (NIDS) is a system which detects intrusive, malicious activities in a host or host's network. The importance of NIDS is growing as the heterogeneity, volume and value of network data continue to increase. This is especially important for current Internet of Things (IoT) networks, which carry mission-critical data for business services. Intrusion detection must deal with highly noisy and unbalanced datasets for which a classifier based on generative models may be more appropriate. The conditional VAE presented in [3], as part of this thesis work, provides a solution based on generative models that shows better classification results that those obtained with classic solutions: Random Forest, SVM, Logistic Regression and MLP.

- Similarly, the **classification of traffic** (type-of-service identification) is of great importance in IoT networks. A Network Traffic Classifier (NTC) is an important part of current network management and administration systems. This classifier infers the service/application (e.g. HTTP, SIP…) being used by a network flow. This information is particularly important for Quality of Service (QoS) management, since the service used has a direct relationship with QoS requirements and user contracts/expectations. Network traffic identification is crucial for implementing effective management of network policy and resources in IoT networks, since the network may need to react differently depending on traffic profile information. The work in this thesis [2] provides a new technique for NTC that considers these problematics and the need to improve the accuracy of the classifiers.
- **Estimation of video QoE** is important in current video transmission systems and its importance will grow with the new capabilities provides by the new network architectures (e.g. edge computing, cloud computing…), with more flexible network management systems which can make better use of quality estimates as perceived by the user of the services. The possibility of making a direct estimation of QoE from the network packets opens up the prospect of real-time quality of service (QoS) estimation, which is critical for the modern services infrastructure.

In Fig 3. is presented a high-level view of the services available for the new IoT applications based on edge computing architectures [41]. Edge computing is a way to streamline the flow of traffic between cloud computing services and particular devices (e.g. IoT) and provide real-time local data analysis at the edge of the network, near the source of the data. This diagram shows the distribution of functions and services of modern networks architectures. The four areas, mentioned in previous sections, where machine learning can be applied to prediction and traffic analysis, can be allocated to the middle layer in Fig 3. This fact allows expanding the processing and distribution capabilities of new network services and is one of the main reasons for the expected future importance of machine learning in IoT networks.

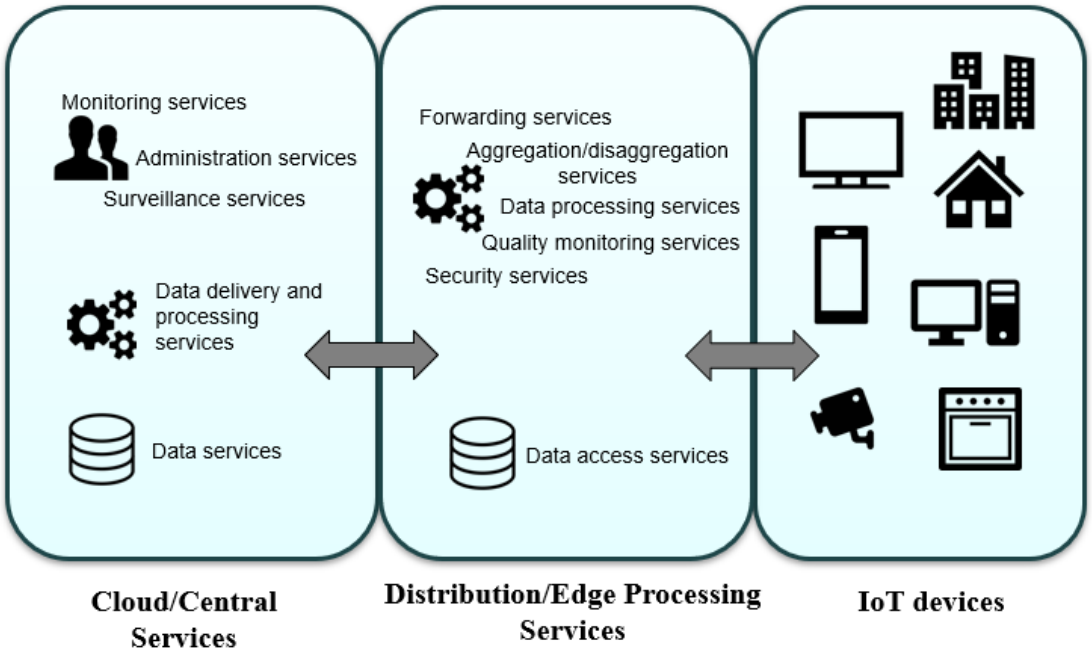


Fig 3. High level diagram of distribution and processing services for IoT applications.

3.2 Specific application areas

It is interesting to mention possible applications of prediction and detection in the field of data networking. Here we present a summary:

- Customer churn prediction
- Customer experience, Quality of Experience (QoE) ***
- Recommender systems
- Congestion prediction and mitigation **
- Network traffic prediction ***
- Performance and failure prediction at device, network and service levels
- Improve customer care, marketing and pricing
- Fraud mitigation
- Improve network operations **
- Type of traffic identification ***
- Intrusion detection ***
- Social media analysis
- Customer behaviour
- Predictive maintenance *
- Support for new networking architectures (e.g. SDN, edge-computing…) **

In the previous list, near each application, there is a series of asterisks that represent the amount of coverage of this application in the research presented in this thesis. Three asterisks mean that we completely cover the corresponding application, two asterisks that we partially cover it, one asterisk a small coverage and no asterisk means there is no coverage.

Type of traffic identification and type of traffic prediction, which are two of the application areas fully covered by this thesis, are identified in a recent IEEE Network Survey [6] as part of the most recent breakthroughs in the application of deep learning and other machine learning techniques to data networks. QoE estimation is also fully covered by this thesis. Finally, the thesis broadly covers intrusion detection from different angles: a) to improve detection and b) to generate synthetic data that can be used to improve detection.

One of the more desirable functions for any data network is the ability to perform accurate and robust detection/prediction of network characteristics which have an impact in the network operations and management, such as:

(1) the future traffic and/or activity in the network

(2) the type-of-service used by a network flow

(3) the presence of security intrusions or malicious activities in the network

(4) the quality of experience (QoE) of a customer using content transmitted by the network (multimedia content).

The first point is important because a data network must handle many diverse service requirements coming from their associated devices; therefore, any knowledge about future traffic behaviour is important to anticipate best resources allocation and possible network reconfigurations [35][36][37][43][44].

The second point is crucial in order to implement effective management of network policy and resources, since the network must react differently according to the service profile for each network connection [45][46][47]. The detection of the service that is being used by a network flow is known in the literature as Network Traffic Classification (NTC).

It is clear the importance of the third point for any network operation, since the ability to detect intrusive, malicious activities or policy violations in a data network is critical due to the complex, sensitive and ever-increasing economic importance of modern network services [34][48].

The fourth point is relevant as the demand for video services increases in parallel with the storage and processing capabilities of these services by the network itself (edge computing, cloud services…). It is now possible to host highly demanding video processing services in the network, which allows to offer new network capacities based on automatic and intelligent analysis of video transmissions and QoE-aware network management and video traffic prioritization and scheduling [39][40][41][42]. Hence the importance of more robust and accurate QoE predictors that can make better use of the new available platforms (e.g. GPUs). In in this thesis is proposed a new QoE predictor which is based in a deep learning model that is especially suitable for these new platforms (e.g. GPU) and that provides better classification results that more classic state-of-the-art-art machine learning algorithms.

Prediction and detection in the four areas considered (traffic estimation, classification of type of traffic, intrusion detection and QoE estimation) present many challenges to a classification algorithm: scarce data, highly unbalanced datasets with a few labels having most samples, noisy data, complex and numerous features, highly correlated features and multi-class classification with usually many values. This is the reason to explore new algorithms, such as: (a) generative algorithms (variational autoencoders) that can handle noisy and complex features within a stochastic framework, and (b) classification algorithms based on convolutional and recurrent neural networks. In the latter case, our hope was that their excellent representational learning capabilities could be extended to these new areas, taking into account their good results in image, voice and text processing, thus avoiding the need for complex feature-engineering that would otherwise be necessary.

The datasets and computing resources available were additional aspects considered when selecting the application areas covered by this research work. The use of machine learning techniques in other areas of application generally requires a large infrastructure due to the volume and complexity of the data (e.g., social network analysis, customer behaviour ...). Another reason for selecting these areas was that they are more technical and less involved with the client or economic aspects of the services, which are generally more problematic due to confidentiality and commercial issues. An exception to this last point is the work done to estimate QoE that is directly related to the user experience. In this case, we created an experimental setup with real individuals who evaluated several video transmissions under different network conditions.

3.2.1 Intrusion detection

Intrusion Detection Systems (IDS) [48][49][50][51] are an important element of the entire Security Ecosystem (SE) consisting of devices, applications, systems, procedures and personnel dedicated to preventing, detecting and avoiding intrusive, malicious activities or policy violations on a host or hosts network. These systems are deployed at different levels, from the highest levels of security analysts/administrators and Security Information and Event Management Systems (SIEM) to the lowest levels of firewalls, antivirus and intrusion detection and prevention systems [52][53]. The lower levels identify and report on the threats and can provide some mechanism for automatic actions (prevention systems), while the higher levels integrate, coordinate, prioritize, decide and launch the actions to be taken.

SE components that detect, report and block threats:

- Firewalls: A firewall allows or blocks outgoing or incoming traffic to an internal network. It is a perimeter security protection. They can work at the packet level, at the connection flow level or at application level, depending on which point of the network protocol hierarchy they operate. To identify threats, they usually look for specific content or signatures in the data (signature-based). It is a first line of defense, but it does not protect against internal attacks within the perimeter protected by the firewall.
- Antivirus/antispyware: They are software installed in the host to alert and protect against virus and malware. They can be based on file scanning searching for defined bytes signatures of the virus. This is a signature-based approach. Another approach is to look for virus actions (behaviour-based approach), monitoring system events and searching for specific patterns or event correlations.
- IDS: Intrusion detection systems (IDS) identify intrusions inside the security perimeter (e.g. established by a firewall). In a first classification, they can be differentiated into host-based IDS (HIDS) and network-based IDS (NIDS), depending on whether they detect threats at the network level or are deployed on a particular host, detecting intrusions only for that host. It is also possible to classify IDS by different detection approaches as: signature-based detection and anomaly-based detection. Signature-based detection methods use a database of previously identified bad patterns to identify and report an attack, while anomaly-based (aka behaviour-based) [54][55] detection uses a model to classify (label) traffic as good or bad, based mainly on supervised or unsupervised machine learning methods. One characteristic of anomaly-based methods is the need to deal with unbalanced data. This happens because intrusions in a system are usually an exception, difficult to separate from the usually more abundant normal traffic. Working with unbalanced data is often a challenge for both the prediction algorithms and performance metrics used to evaluate systems.

  All the works considered for this thesis are NIDS with anomaly-based models

- Intrusion prevention systems: They are similar to IDSs but with the capacity to react to an intrusion with an automatic response (e.g. automatic reconfiguration of a network element)

SE components that integrate information and coordinate the response to threat events:

- SIEM: The function of a SIEM [52] is to aggregate security events, identify security threats and actuate by alerting security personnel and, in some cases, launching automatic commands on network elements. It is responsible for logging the necessary information about security events, including contextual information required by the security analyst to decide the best action. It will also log the information requested by legal or forensic requirements. A SIEM helps identify the relationship between events

using rule-based or correlation methods. They can include capabilities to analyze user behaviors and implement complex automatic response flows.

SIEMs obtain security events by deploying agents in different elements of the network infrastructure hierarchy: hosts, servers, network elements…, and, in different elements of the security infrastructure: firewalls, NIDS, HIDS…

An additional function provided by the SIEM is an integrated visualization function with the ability to help in the consolidated visualization of threats. The visualization of security events [56][57] is a complex issue and it is essential for security analysts to be able to manage the required information which, due to its volume and rapid change, could otherwise be unmanageable.

As a summary, a SIEM collects and analyzes security events from different sources, stores them in a centralized location, correlates events and generates alerts and reports based on this information.

- Security analysts and administrators: They are the final users of the different elements of the security ecosystem, with the final responsibility for the identification and response to security threats. They operate in the Security Operations Center (SOC) [52].

Security attacks in general can be classified into eight main categories [51]:

- Physical attacks: They involve physical damage to computers or network hardware.
- Infection: This category of attacks aims to infect the target system through tampering or by installing infected files in the system (e.g. Viruses, Worms, Trojans).
- Exploding: These attacks seek to overload/overflow the target system (e.g. Buffer Overflow)
- Probe: These attacks collect information about the target system (e.g. Sniffing, Port Mapping Security Scanning).
- Cheat: They access the system with fake identities (e.g. IP Spoofing, MAC Spoofing, DNS Spoofing, Session Hijacking, XSS (Cross Site Script) Attacks, Hidden Area Operation, and Input Parameter Cheating)
- Traverse: This category of attacks uses all possible ways to match the system credentials to access the system (e.g. Brute Force, Dictionary Attacks, Doorknob Attacks).
- Concurrency: They alter the availability of the system by sending massive requests that the system cannot handle (e.g. Flooding, DoS, DDoS)
- Others: These are attacks on systems that are not configured or maintained properly and that have a known vulnerability/weakness that compromises them.

In addition, security attacks can be classified as passive and active [51]. Passive attacks only collect information (host or network traffic). Active attacks actuate on the attacked system. Active attacks are classified into four categories according to the Defense Advanced Research Projects Agency (DARPA):

- DoS: Denial of Service Attacks are designed to make computer or memory resources too busy or too full to handle legitimate network requests and, therefore, deny users access to a machine (e.g. apache2, smurf, neptune, dosnuke, land, pod, back, teardrop, tcpreset, syslogd, crashiis, arppoison, mailbomb, selfping, processtable, udpstorm, warezclient)

- Probe: These attacks scan the network/hosts to gather useful information about the hosts with the intention of launching attacks (e.g. portsweep, ipsweep, queso, satan, msscan, ntinfoscan, lsdomain, illegal-sniffer)
- R2L: In this type of attacks, a user without an account in the system obtains local access (e.g. dict, netcat, sendmail, imap, ncftp, xlock, xsnoop, sshtrojan, framespoof, ppmacro, guest, netbus, snmpget, ftpwrite, httptunnel, phf, named)
- U2R: In this type of attacks, the user who has the privileges of a normal local account tries to obtain superuser privileges (e.g. sechole, xterm, eject, ps, nukepw, secret, perl, yaga, fdformat, ffbconfig, casesen, ntfsdos, ppmacro, loadmodule, sqlattack)

It is also important to mention the fundamental role played by security logs, which are critical at all security management levels. Security logs are the main entry point of information for security threats, having their own ecosystem of functions [58]: acquisition, filtering, normalization, collection management, storage, analysis and long-term storage. IDSs are an important producer of security logs; IDSs can report intrusions in real-time or off-line by security logs. Likewise, IDSs are consumers of security logs. Anomaly-based IDSs use security logs to train the classification algorithms and may need security logs as contextual information while they are in operation.

Intrusion detection based on machine learning can be supported by supervised or unsupervised methods. Supervised methods employ the usual models: MLP, SVM, Logistic Regression… [48]. Unsupervised methods can be set-up in different ways [59], adopting different approaches: probabilistic methods, clustering methods or deviation methods. In probabilistic methods, we characterize the probability distribution of normal data and define as an anomaly any data with a given probability lower than a threshold. In clustering methods, we cluster the data and categorize as an anomaly any data too far away from the desired normal data cluster. In deviation methods, we define a generative model able to reconstruct the normal data, in this setting we consider as an anomaly any data that is reconstructed with an error higher than a threshold.

In [3] we present an anomaly-based supervised machine learning method (based on a C-VAE). We will use a deviation-based approach, but instead of designating a threshold to define an intrusion, we will use a discriminative framework that will allow us to classify a specific traffic sample with the intrusion label that achieves less reconstruction error.

In relation with anomaly-based NIDS, there are several publicly available datasets containing network traffic with different types of anomalies (KDD99, NSL-KDD, UGR16, CAIDA, AWID, ADFA…). It is very difficult to compare the results obtained by different works on intrusion detection, since the datasets are usually different and even when they are similar, the test sets used to present the results are usually different from the one proposed by the creators of the datasets.

There is no report, as far as we know, on the use of a C-VAE for classification in the field of network intrusion detection. There are works that present results applying other deep learning models in this field. In [60] a neural network is used in a simulated IoT network. The work in [61] presents a classifier which detects intrusions in an in-vehicle Controller Area Network (CAN), using a deep neural network pre-trained with a Deep Belief Network (DBN). The

authors of [62] use a stacked autoencoder to detect multi-label attacks in an IEEE 802.11 network. In [63] is implemented an intrusion classifier combining spectral clustering and deep neural networks in an ensemble algorithm. An and Cho [64] presents a classifier solution using a VAE in the intrusion detection field.

The following table presents a summary of the main works related to the research carried out for this thesis (anomaly-based NIDS). It provides a reference to the document, the data set used and the scope of the work.

Objective/Area	Ref.	Dataset	Scope
Intrusion detection	[64]	KDD99	- Classifier solution using a VAE in the intrusion detection field, but it is a VAE (not CVAE) with a different architecture to the one presented in this thesis.
	[65]	Yahoo S5 time-series dataset	- They use a recurrent neural network (RNN) with a CVAE to perform anomaly detection on a big time-series dataset. It is applied to generic multivariate time-series. The architecture is different to the one presented in this thesis, and the results are not related to NIDS.
	[66]	Multivariate time-series from robot movements	- It is employed an RNN with a VAE to perform anomaly detection on multivariate time-series coming from a robot. Data and results are not applicable to NIDS.
	[24]	MNIST	- The authors apply a CVAE to a semi-supervised image classification problem. Data and results are not applicable to NIDS.
	[60]	Simulated IoT network	- A neural network is used for detecting DoS attacks in a simulated IoT network, reporting an accuracy of 99.4%.
	[61]	In-vehicle Controller Area Network (CAN)	- This work presents a classifier which detects intrusions in an in-vehicle Controller Area Network (CAN), using a deep neural network (DNN) pre-trained with a Deep Belief Network (DBN). the DNN is trained with probability-based feature vectors that are extracted from the in-vehicular network packets. For a given packet, the DNN provides the probability of each class discriminating normal and attack packets, and, thus the sensor can identify any malicious attack to the vehicle.
	[62]	AWID dataset focused on intrusion detection	- The authors of this work use a stacked autoencoder to detect multilabel (4-class) attacks in an IEEE 802.11 network with an overall accuracy of 98.6%. They use a sequence of sparse auto-encoders but they do not use variational autoencoders.
	[63]	NSL-KDD	- They implement an intrusion classifier combining spectral clustering and deep neural networks in an ensemble algorithm. They used the NSL-KDD dataset in different configurations, reporting an overall accuracy of 72.64% for a similar NSL-KDD configuration to the one presented in this paper.

[67]	NSL-KDD	- It explains why and how the NSL-KDD data set was created. They provide results of applying several methods to the NSL-KDD data. The best accuracy reported is 82.02% with Naive Bayes Tree using Weka. They use the full NSL_KDD dataset for training and testing, for the 2-labels prediction scenario.
[68]	NSL-KDD	- It is applied a multilayer perceptron (MLP) with three layers to the NSL-KDD dataset, they achieved an accuracy of 79.9% for test data, for a 5-labels intrusion scenario. For a 2-labels (normal vs. anomaly) scenario they provided an accuracy of 81.2% for test data.
[69]	NSL-KDD	- The authors report, for a 2-labels scenario and using self-organizing maps (SOM), a recall of 75.49% on NSL-KDD test data.
[70]	NSL-KDD	- The authors report, employing AdaBoost with naive Bayes as weak learners, an F1 of 99.3% for a 23-labels scenario and an F1 of 98% for a 5-labels scenario; to obtain these figures they used 62,984 records for training (50% of NSL-KDD), where 53% are normal records and the remaining 47% are distributed among the different attack types; test results are based on 10-fold cross-validation over the training data, not on the test set.
[71]	Real data of private network	- This work performs classification using a generative model based also in a Hidden Markov Model. This work reports a precision of 93.2% using their own dataset.
[72]	NSL-KDD	- The authors resorted to a deep belief network applied to the NSL-KDD dataset to do intrusion detection. They reported a detection accuracy of 97.5% using just 40% of the training data, but it is unclear what test dataset is used.
[73]	1998 DARPA dataset (intrusion detection)	- They employed continuous time bayesian networks as detection algorithm, using the 1998 DARPA dataset. They achieved good results on the 2-labels scenario; the metric provided is a ROC curve diagram, but no performance figures are given.
[74]	KDD99, NSL-KDD, Real traffic	- This is a reference paper. It presents a survey of works related to machine learning architectures applied to NIDS, including generative models.
[48]	NSL-KDD	- This paper explains the reasons for creating the NSL-KDD dataset. They give results for several algorithms. The best accuracy reported is 82.02% with naive Bayes tree using Weka. They use the full NSL_KDD dataset for training and testing, for the 2-labels scenario.
[75]	NSL-KDD	- It proposes an approach to detect attacks using a deep auto encoder to perform dimensionality reduction before performing the classification of attacks. The algorithm used for classification is not mentioned. Using the deep autoencoder, the final accuracy on test data is 95.06%, with an unclear test dataset. The dimensionality reduction with the

			autoencoder provides better final accuracy than PCA, factor analysis and Kernel/PCA.
	[76]	KDD99	- It applies Restricted Boltzmann Machines (RBM) in anomaly detection using the KDD99 dataset, obtaining an accuracy of 94% on the training dataset and 83% on the test dataset.
	[77]	KDD99	- A Deep Belief Networks (DBN) is used to train a classifier to detect intrusions, comparing the performance to SVM and ANN. DBN obtains the best accuracy of 93.49% which is achieved with a deep DBN of 122-150-90-50-5 (nodes per layer) plus a final backpropagation 2-layer neural network ended with a softmax (classification) layer.
	[78]	NSL-KDD	- For the 2-labels scenario, and using Naive Bayes with several feature engineering methods, they report an accuracy of 96.5% but the test set used is unclear.
	[79]	NSL-KDD	- They report an accuracy of 99.1% using several methods (SVM, NB...) with a previously performed dimensionality reduction on the features, but again, it is not clear the test set used, and the metrics are given on subsets of the anomaly types.
	[80]	NSL-KDD	- they report an accuracy of 99.9% with AdaBoost and a selection of features using a wrapper model; they use a subset of the NSL-KDD dataset for training and an unclear test set
	[81]	NSL-KDD	- For the 2-labels scenario and using AdaBoost with weak learners being simple decision stumps, they report a detection rate of 90% on test data.
	[82]	KDD99	- The authors propose a least squares support vector machines (LS-SVM) based intrusion detection model, using kernel space approximation through greedy searching. They obtain a best detection rate of 0.98
	[83]	KDD99	- This paper proposes using optimized Regularized Least Square (RLS) classification combined with k-means clustering plus kernel approximation techniques. It obtains a best accuracy of 0.986 for DoS label detection in a one vs. rest detection mode.

Table 1. Intrusion detection - related works

3.2.2 Traffic prediction

Traffic prediction is a classic problem in networking which has been usually tackle with time-series prediction models [8][84] such as: Hidden Markov Model, Exponential Smoothing, AutoRegressive Integrated Moving-Average (ARIMA), ARIMA with eXogenous covariates (ARIMAX), etc… [85][86]

The problem of traffic prediction consists in forecasting the traffic volume or connectivity status of a communication flow in the short, medium- or long-term future.

Besides the above-mentioned time-series models the usual ML models applied to this problem have been: Neural Networks, SVM and Recurrent Neural Networks-LSTM (more recently) [8][86]

All the experiments in this area are performed with proprietary data as it is the case for the research performed in this thesis.

The prediction of network traffic using deep learning does not have much literature. In the field of the prediction of network activity/traffic, the use of other machine learning methods is more frequent, mainly time-series methods [87][88][89]. Considering some of the few examples of application of deep learning to traffic forecasting: in [90] a deep learning network (CNN, RNN) is used for mobile traffic forecasting using spatiotemporal features; and for a similar problem, in [91] a combination of LSTM and Stacked Autoencoder networks is applied.

The following table presents a summary of the main works related to the research carried out for this thesis. It provides a reference to the document, the data set used and the scope of the work.

Objective/Area	Ref.	Dataset	Scope
Traffic prediction **- Supervised**	[43]	Simulated network data	- Prediction using logistic regression of network reliability for wireless sensor networks, using two parameters to make the prediction number of sensors nodes and transmission range. It is a binary prediction but of a very different nature to this thesis works.
	[44]	Real WLAN traffic traces	- Using SVM to forecast traffic in WLANs. They study the issues of one-step-ahead prediction and multi-step-ahead prediction - Prediction of continuous values (with MSE and NMSE) which makes them impossible to compare with the results of this thesis work (discrete prediction). - SVM presents better results than MLP, ARIMA and Fractional ARIMA. They provide the Mean Square Error for different time-steps predictions.

	[37]	Real wireless network with 66 base stations.	- This work presents two applications for wireless networks traffic forecasting, the prediction of the moment when a specified Base Station (BS) will saturate (long term prediction) and the prediction of traffic anomalies (short term prediction). - Comparison of MLP and ARIMA. For short term predictions MLP is better, for long term predictions ARIMA is better. - Prediction of continuous values (with MAE, MAPE and SMAPE) which makes them impossible to compare with the results of this thesis work (discrete prediction).
	[92]	GEANT backbone networks	- Application of several RNN architectures (LSTM and GRU) to predict network traffic.
	[93]	Several WLAN traffic traces of: 1) Network Research Laboratory at Tianjin University, 2) the Mobile Computing Group at Stanford University and 3) the intranet traffic on the ACM SIGCOMM'01 conference held in the U.C. San Diego in August 2001	- They show that network traffic prediction using SVM outperforms ARIMA, MLP and fractional ARIMA (FARIMA) for both one-step-ahead and multi-step-ahead predictions.
	[90]	CDR recording of Telecom Italia in the city of Milan	- They propose several CNN-RNN architectures as an alternative to ARIMA models, obtaining a forecasting accuracy of 70-80%
	[91]	Big proprietary dataset from China Mobile	- They present a hybrid deep learning model for spatiotemporal prediction, with an autoencoder for spatial modeling and LSTMs for temporal modeling. They improve the results obtained with ARIMA.
Traffic prediction **- Time series**	[36]	GSM traces of China Mobile	- Seasonal ARIMA model to predict wireless traffic workload. The relative error between forecasting values and actual values are all less than 0.02.
	[35]	Real wireless network	- Study of time-series traffic forecasting algorithms for IEEEE802.11 network. - Short term traffic prediction in a large wireless infrastructure. - Prediction of continuous values (with median absolute and relative prediction errors) which makes them impossible to compare with the results of this thesis work (discrete prediction). - Comparison of several Moving Average and ARIMA variants. The best forecasting performance is obtained with exponentially weighted moving average (EMA).
	[94]	Real proprietary network traffic and simulated one	- Network traffic volume prediction with an HMM model
	[87]	Proprietary WiMAX traffic traces	- Authors present a comparison between ARIMA, MLP and Stationary Wavelet Transform, showing that MLP provides better results for volume of

			traffic short term predictions and ARIMA for long-term predictions.
	[88]	IEEE802.11 traffic at the University of North Carolina at Chapel Hill	- They evaluate a series of variants of ARIMA, Moving Average (MA) and Exponentially Weighted Moving Average (EMA) methods. They obtain best results for the MA methods.
	[89]	Real traffic traces measured from the GSM network of China Mobile of Tianjin.	- They present the results of applying a seasonal ARIMA model with a relative error of 0.02. There is no comparative with other models.

Table 2. Traffic prediction - related works

3.2.3 Type of traffic prediction (traffic classification)

Type of traffic prediction aka Network Traffic Classification (NTC) is an important part of current network management and administration systems. An NTC infers the service/application (e.g. FTP, Radius, LDAP...) being used by a network flow. This information is important for network management and Quality of Service (QoS), as the service used has a direct relationship with QoS requirements and user contracts/expectations.

Network traffic identification is crucial for implementing effective management of network policy and resources in data networks, as the network needs to react differently depending on traffic profile information.

There are several approaches to NTC: port-based, payload-based, and flow statistics-based [95][96]. Port-based methods make use of port information for service identification. These methods are not reliable as many services do not use well-known ports or even use the ports used by other applications.

Payload-based approaches the problem by deep packet inspection (DPI) of the payload carried out by the communication flow. These methods look for well-known patterns inside the packets. They currently provide the best possible detection rates but with some associated costs and difficulties: the cost of relying on an up-to-date database of patterns (which must be maintained) and the difficulty to be able to access the raw payload. Currently, an increasing proportion of transmitted data is being encrypted or needs to assure user privacy policies, which is a real problem to payload-based methods.

Finally, flow statistics-based methods rely on information that can be obtained from packets header (e.g. bytes transmitted, packets interarrival times, TCP window size,). They rely on packet header high-level information which makes them a better option to deal with non-available payloads or dynamic ports. These methods usually rely on machine learning techniques to perform service prediction [95].

The works presented here are based on flow statistics-based methods.

There are many datasets available to carry out experiments for NTC (Moore, WIDE…). However, most of the experiments are done with proprietary traffic as it is the case for the research performed in this thesis.

If we focus on works related with deep learning models, this thesis provides the first study, as far as we know, of a CNN+RNN model applied to NTC. There are many works that apply neural networks to NTC, but the network models employed are variants of MLP classifiers. In [97] they propose a multi-layer perceptron (MLP) with one hidden layer. An ensemble of MLP classifiers is applied in [98]. In [99] an MLP with a particle swarm optimization algorithm is employed. Zhou et al. [100] apply an MLP with 3 hidden layers. A Parallel Neural Network Classifier Architecture is used in [101], it is made up of parallel blocks of radial basis function neural networks.

The following table presents a summary of the main works related to the research carried out for this thesis. It provides a reference to the document, the data set used and the scope of the work.

Objective/Area	Ref.	Dataset	Scope
Type of traffic classification	[97]	Internet traffic manually classified [102]	- They propose a multi-layer perceptron (MLP) with one hidden layer, but it is actually adopted as the internal architecture to apply a fully Bayesian analysis. The best one vs. rest accuracy, using 246 features, for 10 grouped labels is 99.8%, and a macro averaged accuracy of 99.3% (10 labels).
	[98]	TCP traces of backbone router of the University of Jinan	- Classification with an ensemble of MLP classifiers with error-correcting output codes, achieving an average overall accuracy (for 5 labels) of 93.8%.
	[99]	Auckland IV.: public available packet trace.	- An MLP with a particle swarm optimization algorithm is employed to classify 6 labels with a best one vs. rest accuracy of 96.95%.
	[103]	Moore dataset [104]	- They use an MLP with 3 hidden layers and different numbers of hidden neurons, showing an overall accuracy greater than 96%, for a grouping of labels in 10 classes, resulting in a final class distribution very unbalanced (a frequency of almost 90% for highest frequency class), no F1 score is provided.
	[100]	Moore dataset	- They apply an MLP with 3 hidden layer combined with a fast correlation-based feature selection.
	[101]	Data collected at the Florida Institute of Technology	- A Parallel Neural Network Classifier Architecture is used. It is made up of parallel blocks of radial basis function neural networks. To train the network is employed a negative reinforcement learning algorithm. They classify 6 labels reporting a realistic overall accuracy of 95%, no F1 score is provided.
	[105]	Traces collected at two backbone and two edge links located in the U.S., Japan, and Korea	- This work proposes an entropy-based minimum description length discretization of features as a preprocessing step to several algorithms: C4.5, Naïve Bayes, SVM and kNN. Claiming an enhanced performance of the algorithms, achieving a one vs. rest accuracy of 93.2%- 98% for 11 grouped labels.
	[106]	Proprietary network traffic captured with WireShark	- Authors apply different machine learning techniques to NTC (C4.5, Support Vector Machine, Naïve Bayes) reporting an average accuracy of less than 80% using 23 features and detecting only five services (www, dns, ftp, p2p, and telnet)
	[107]	Proprietary dataset	- They employ an enhanced random forest with 29 selected features. They group the services in 12 classes, providing only one vs. rest metrics (not aggregated). Having F1 scores in the interval 0.3-0.95, with only 3 classes higher than 0.96.
	[108]	WIDE backbone dataset	- This work includes flows correlation in a semi-supervised model providing overall accuracy of less than 85% and a one vs. rest F1 score, for 10 labels,

			of less than 0.9 (except two labels with 0.95 and 1). They report having better results than other works using C4.5, kNN, Naïve Bayes, Bayesian Networks and Erman´s semi-supervised methods.
	[109]	Moore dataset [104]	- A Directed Acyclic Graph-Support Vector Machine is proposed in this work, attaining an average accuracy of 95.5%. The method is applied to a one-to-one combination of classes
	[110]	UNB ISCX Network Traffic dataset [111]	- They study the application of several algorithms: J48, Random Forest, Bayes Net, and kNN to UNB ISCX Network Traffic dataset, with 14 classes and 12 features, reporting a best classification accuracy of 93.94% for the kNN algorithm.
	[112]	Moore dataset [104]	- This work presents a variant of decision tree algorithm C4.5 working on the Hadoop platform. They classify 12 labels giving a one vs. rest accuracy in the interval 60-90% for all the labels with only two labels with a value higher than 90%.
	[113]	Traces from the Internet Link of the University of Calgary	- This work presents Erman´s semi-supervised method. This method consists in clustering the flows using K-Means or some alternative clustering method and then mapping the clusters centroids to traffic types using Euclidean distance. An accuracy greater than 90% is reported.

Table 3. Type of traffic prediction - related works

3.2.4 QoE estimation

QoE is defined by ITU-T as "the overall acceptability of an application or service, as perceived subjectively by the end user". The ability to evaluate the QoE in a communication system, and especially in a system involved in video transmission, is critical. One of the main objectives of modern network management systems is to monitor and guarantee end-user Quality of Experience (QoE), hence the importance of an accurate QoE monitoring system.

The usual way to evaluate QoE is either to carry out experiments with individuals as testers or to calculate it indirectly from Quality of Service (QoS) network parameters (jitter, delay, packet loss....) [42][114][115].

Another approach, recently being actively explored is applying machine learning (ML) to video QoE estimation. The resulting QoE detector must predict a QoE score directly from information contained in the transmitted videos, the network packets or end-user recorded events (e.g. related web activity). This approach is the one taken for the research performed as part of this thesis which provides a video QoE detector from network packets information using deep learning models. This is also the most advanced and precise approach [42] that shifts the focus of video quality assessment from QoS (system oriented) to QoE (user oriented).

The datasets used to perform experiments in QoE are mainly proprietary, as has been the case for the dataset used for the research carried out for this thesis.

As far as we know, there are no previous works presenting the application of a CNN+RNN model to video QoE estimation, hence we believe that the research presented in this thesis [4] is original in this regard.

The following table presents a summary of the main works related to the research carried out for this thesis. It provides a reference to the document, the data set used and the scope of the work.

Objective/Area	Ref.	Dataset	Scope
QoE estimation	[39]	System prototype in an OpenStack based virtualization environment	- They design a three-tier edge computing system architecture to elastically adjust computing capacity and dynamically route data to proper edge servers for real-time surveillance applications. - It demonstrates the reconfiguration capabilities of current network services.
	[40]	N/A	- This work highlights some of the potentials and prospects of edge computing for interactive media. - It presents the importance of QoE estimate and control in multimedia applications
	[41]	N/A	- It provides an overview of real-time video analytics applications that are (or will soon be) performed by

		the network.
[42]	N/A	- It gives a comprehensive survey of the evolution of video quality assessment methods, analysing their characteristics, advantages, and drawbacks. - It also introduces QoE-based video applications and, identifies the future research directions of QoE-oriented video quality assessment.
[114][115]	Emulation of Internet Service Provider (ISP) network, at Polytechnic University of Valencia, Spain.	- This works proposes an analytical expression for video QoE calculation based on several parameters: jitter, delay, bandwidth, loss packets and zapping time for IPTV video transmissions - They provide an automatic Video Quality Assessment (VQA) based on the identification and processing of parameters extracted from the video. - They present a QoE management system to guarantee enough IPTV QoE to the customer independently of its type of connection (wired or wireless). The system calculates the user's QoE and notifies which networks are available and have higher QoE. - QoE estimates are based in a mathematical expression connecting several network measurements. It is not based in a machine learning algorithm.
[116]	N/A	- They produce a theoretical discussion on how to use QoS parameters (e.g. delay, jitter…) to predict QoE using a dataset built from subjective end-user scores, and applying machine learning algorithms based on Support Vector Machine (SVM) and Decision Trees.
[117]	N/A	- This work provides a survey of machine learning techniques used to capture the relationship between QoS parameters and QoE scores. - They apply most of the common machine learning algorithms (Linear Discriminant Analysis, Random Forest (RF), SVM, Naïve Bayes, K-Nearest Neighbors) to the automatic identification of QoE from QoS network parameters.
[118]	Dataset based on 40 million video viewing sessions on conviva.com's affiliate content providers' websites.	- It focuses on Content Delivery Networks (CDN). - It gives a review of the reasons why developing an objective method of quality assessment based on video transmission parameters is extremely difficult due to the complex relationships between these parameters, the user's perception and even the nature of the content. - They apply machine learning algorithms (Decision Trees, Naïve Bayes and Logistic Regression) to predict the QoE based on transmission parameters (bitrates, latency...) and end-user engagement attributes (playtime, number of visits...).
[119]	LIVE-Netflix Video QoE Database	- They perform prediction of streaming video QoE applying several regression models such as Ridge and Lasso Regression, and ensemble methods such as Random Forest (RF), Gradient Boosting (GB) and Extra Trees (ET).

Table 4. QoE estimation - related works

3.2.5 Synthetic data generation

The main principle behind all ML models is that they learn from data instead of learning in an imperative way based on predefined rules (programming paradigm). Hence, the importance of having large representative datasets. Large datasets are important since the objective is to be able to create algorithms that can generalize to data outside of the data used for training, hence the need of a representative dataset. A dataset is representative if it includes samples that represent all possible behaviors that we try to model with our algorithm, and avoids non-representative samples (noise). Since the behaviour of systems is often complex, their representative datasets are usually large. When we have problems acquiring a representative dataset due to cost, time, privacy or technical difficulties, and we end up with small datasets or datasets that do not include sufficient samples of under-represented behaviours, then we need to consider the use of synthetic data.

In order to create a dataset that can be used for model training, we can have three alternatives [120]:

- Real data: data generated by the normal generation environment associated with the data and that we try to model with our ML algorithm
- Semi-synthetic data: data generated by an artificial generation environment that tries to be similar to the normal generation environment of the data. In this case the intention is to reproduce virtual entities (e.g. users, systems...) with a behaviour similar to the real one, with the intention that the data produced by the simulated environment is similar and representative of the real one. The simulation can be based on physical entities (e.g. network, switches, computers...) or simulated by software processes.
- Synthetic data: data synthetically created without using a simulated generation environment. This data is created trying to be similar to the real data (e.g. correlation, probability distribution, patterns...). In this case, we synthetize the data directly instead of obtaining it by simulating the data generation environment.

There are pros and cons for all three alternatives [120]. Of course, the best option is to have a real and representative dataset. Since this is not always an option, the next best option is to create semi-synthetic data that simulates the data generation process in a realistic way. But, in many cases, due to cost, time, or technical difficulties, the only available option is to create synthetic data. This latter option can be problematic, since the generated data can be noisy and not representative of the original data, therefore, it is important to articulate good methods to generate synthetic data when all other possibilities are not feasible. Synthetic data should resemble the actual data, but with the variability required to not be an exact copy of the original data.

Intrusion detection is an area particularly interesting for the generation of synthetic data. Acquiring a representative dataset can be costly and time consuming even with a simulated environment. In addition, the intrusion detection datasets are strongly biased to normal traffic, being difficult to access traffic associated with intrusion events. Regarding unbalanced datasets there are well-known over-sampling algorithms (SMOTE, ADASYN,..)[5] that create synthetic data for the under-represented classes. The main idea of these algorithms is to create new samples close (under a defined distance measure) to existing samples that belong to some specific minority class, therefore, it is important how the “distance” function is defined, which is not an easy task as demonstrated by the many variants of the SMOTE algorithm.

Another possibility to create synthetic data is provided by generative models that learn the latent joint probability distribution of the data. This allows the subsequent sampling of the joint probability distribution, creating synthetic data with a joint probability distribution similar to that of the original data. This is an alternative way to generate synthetic data, and it is the one that is followed in this thesis [5] using a variational autoencoder. Authors in [121] present a work of a similar nature, where a generative model is constructed to capture the joint probability distribution of the data. In this case, the data to be synthetized is relational data (contained in a database). The joint probability distribution for the complete dataset is obtained through a complex process that identifies the probability distribution of each column in the database, followed by an estimate of the covariance between columns using a Gaussian Copula. The covariance estimate is extended to related tables. To synthetize new data, they sample through the resulting (and complex) joint probability distribution. A similar approach to synthetize data with different generative models has also been applied to generate images [10][35][38] and text [36][43].

When generating synthetic data there are two scenarios: (a) to create synthetic samples with all their features [5], or, (b) to complete partially-filled samples where the values of some features are known but other are missing, in this case the synthesis is reduced to the missing features, with the important constraint of synthetizing the missing features conditioned on the values of the known ones [3].

There are several works related to the creation of semi-synthetic data for intrusion detection: In [122] the authors propose a modular synthetic dataset generation framework for web applications, together with a monitoring environment to collect data at multiple protocol layers (e.g. TCP, database queries, system calls...). They can create different types of attacks or reuse existing ones by adopting the Metasploit Framework within their own simulation environment, which they call Wind Tunnel. The approach corresponds to a semi-synthetic model. The work in [123] proposes a simulated environment to create intrusion data for a vehicular adhoc network (VANET). They present an experiment using a network simulator with 10 simulated scenarios of mobility of VANET hosts and 5 types of emulated security threats with the capacity to define the total number of vehicles and the number of malicious hosts in the VANET. In [124] a generator architecture (semi-synthetic approach) is proposed for datasets of system calls used for host intrusion detection systems (HIDS). The generator architecture is generic, but it is demonstrated using Ubuntu Linux and Mozilla Firefox as the profiled application. Authors in [125] implement a software simulated environment to create high-level human threats produced by malicious employees/agents inside an organization. They create a complex high-level simulated environment including aspects such as human behaviour, relationship and communications models within the organization. They create synthetic datasets corresponding to complex threats scenarios associated with personal dynamics within the organization

Synthetic data generation is an interesting research area that will surely be further explored with the arrival of new algorithms (e.g. variational autoencoders and generative adversarial networks). The following table presents a summary of the main works related to the research carried out for this thesis. It provides a reference to the document, the data set used and the scope of the work. In this case, only similar works (for synthetic data) are presented in a general sense and coming from different fields.

Objective/Area	Ref.	Dataset	Scope
Synthesize data	[126]	Data collected from sensors deployed in the Intel Berkeley Research Laboratory	- They propose a method to recover missing (incomplete) data from sensors in IoT networks using data obtained from related sensors. The method used is based on a probabilistic matrix factorization and it is more applicable to the recovery of continuous features
	[127]	MNIST and Cocaine-Opioid and Alcohol-Cannabis datasets (NIH-funded project)	- Reconstruction of missing data for multimodal datasets. The proposed model is based on a combination of a denoising autoencoder and a variant of a generative adversarial network. It obtains better results than alternatives models such as: matrix factorization, multimodal autoencoder, pix2pix and CycleGAN - This work can be considered aligned (but not strictly similar) with the present thesis work, but it requires a training process and a network both more complex.
	[128]	MNIST	- Reconstruction of missing parts of digits of the MNIST dataset using a VAE and a variant of principal component analysis (PCA). The model based on VAE provides the best reconstruction of the missing parts. It does not employ a conditional VAE.
	[129]	MINIST and Frey Face datasets	- First application of VAE to image generation
	[130]	MINIST, CIFAR-10 and Toronto Face Database.	- First application of Generative Adversarial Networks to image generation
	[131]	QASent and WikiQA datasets.	- Text generation variational autoencoder, conditioned on an input text.
	[132]	Yahoo Answer and Yelp15 review datasets	- Text generation with a VAE and a dilated CNN as the decoder.
	[121]	Biodegradability, Mutagenesis, Airbnb, Rossmann and Telstra. All open-source datasets.	- Generative model for relational data in general. They fit the probability distribution for the columns data using the Kolmogorov-Smirnov test as a measure of goodness of fit to some predefined distributions. They use a Gaussian Copula to model the covariance between different columns.

Table 5. Synthetic data generation - related works

4. RESEARCH SCOPE

As a summary, in Table 6 is presented the research scope, considering the objectives, areas of application and methods and algorithms applied.

Objective	Area	Method	Algorithms
Synthesize data	Intrusion/security	Generative/Unsupervised	Conditional VAE
Synthesize data	Intrusion/security	Unsupervised/Supervised	SMOTE, ADASYN
Detection	Intrusion/security	Generative/Unsupervised	Conditional VAE
Detection	Intrusion/security	Supervised	Random Forest, Linear SVM, Logistic Regression, MLP
Detection/Prediction	Type of traffic classification	Supervised	CNN, LSTM
Prediction	Traffic estimation	Supervised (based in transformed cross-sectional data)	Random Forest, Logistic Regression, Bayesian Logistic Regression, GBM
Prediction	Traffic estimation	Time-series (based in original time-series data)	Exponential Smoothing, HMM, ARIMA, ARIMAX
Prediction	Quality of Experience estimation	Supervised	CNN, LSTM, Gaussian Process

Table 6. Scope of this research

In Table 6 are presented all the algorithms that have been used for this work, some of these algorithms have been used to carry on analysis and comparative assessments between different models and others are new proposals/architectures/models which are part of the contributions of the thesis,

It is interesting to have a comparative summary between the broad categories of methods related with machine learning, deep learning and generative models. This is a broad comparative that position each group by their main differences, even when they are actually interconnected (Fig 4).

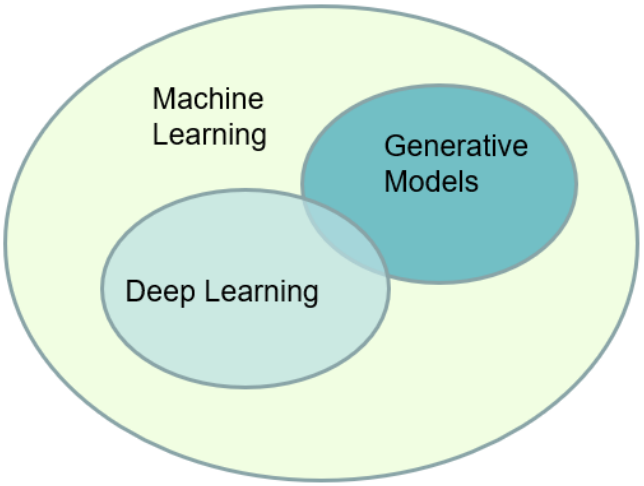


Fig 4. Relationship between machine learning, deep learning and generative models

Machine learning:

Pros

- Solid and well-known methods
- Used in many applications
- Easy interpretability of results for some models (e.g. decision trees)

Cons

- Requires extensive feature engineering
- Many models available but not integrable during the learning phase. Ensemble methods are the way to integrate them, but only at the results level.
- Difficult interpretability of results for some models (e.g. random forest)

Deep learning:

Pros

- State of the art results for some problems
- Requires less feature engineering
- Good for representation learning and to create data embeddings
- Framework in which many models can be integrated (in a Lego-way) as long as the resulting loss and activation functions are differentiable.

Cons

- Generally, requires large datasets
- Many hyper-parameters and difficult tuning
- High computational resources demanded for training
- Lack of interpretability (difficult justification of algorithm decisions)

Generative models:

Pros

- Based on learning the probability distribution of the data (features and labels jointly)
- Allows the sampling of the probability distribution to generate new data or imputation of missing features.
- Possibility to learn a latent representation of the data simultaneously (e.g. VAE)

- Better classification performance compared to discriminative models for small datasets, since the constraints imposed by the generative assumptions needed to build the model work against over-fitting

Cons

- Poor classification performance compared to discriminative models for medium/large datasets
- Requires large datasets to build correct models.
- More difficult to train (usually more parameters to tune)

This comparative is a generic introduction to the specific architectures proposed in this thesis. In the following sections are presented in detail the new architectures/models proposed in this thesis and how they are applied to the areas listed in Table 6.

4.1 Combination of convolutional and recurrent neural networks

The novel applications of deep learning proposed in this thesis, for QoE and NTC prediction, are based on an architecture formed by combining CNN and RNN networks.

This architecture combines two interesting properties that have demonstrated to be important to improve the classification capabilities of the resulting classifiers:

- The initial CNN layers are able to perform automatic knowledge representation. CNNs were initially applied to image processing, as a biologically inspired model to perform image classification, where feature engineering was done automatically by the network thanks to the action of a kernel (filter) which extracts location invariant patterns from the image. Chaining several CNNs allows extracting complex features automatically.
- A subsequent block formed by RNN layers allows to capture the time dependent information contained in the processed data. The RNN layers are specially indicated to process sequential information and have been applied mainly to textual and time-series data.

The principles for the application of this combined architecture to network flow packets, are:

- To transform the flow of data packets to an image-like data structure that can be processed by a CNN. We consider the matrix formed by the time-series of feature vectors as an image. Image pixels are locally correlated; similarly, feature vectors associated with consecutive time slots present a correlated local behavior, which allows us to adopt this analogy.
- The data structure generated by the CNN block must be transitioned to a new data structure for the LSTM block. In this transition, it is important that the time dimension of the tensor produced by the CNN block is maintained and transformed to the time dimension of the input tensor of the LSTM block. This transition is performed by reshaping and permuting the dimensions of the tensors.
- To perform a fine adjustment of the CNN and LSTM layers and their parameters. Adding more layers do not produce necessarily an improvement in classification performance.

Fig 5 presents the combined CNN-RNN architecture. Of the different options available for the RNN block (LSTM, GRU...), we have chosen the LSTM model because its slight additional complexity (in comparison, for example, with GRU) is compensated by its better results.

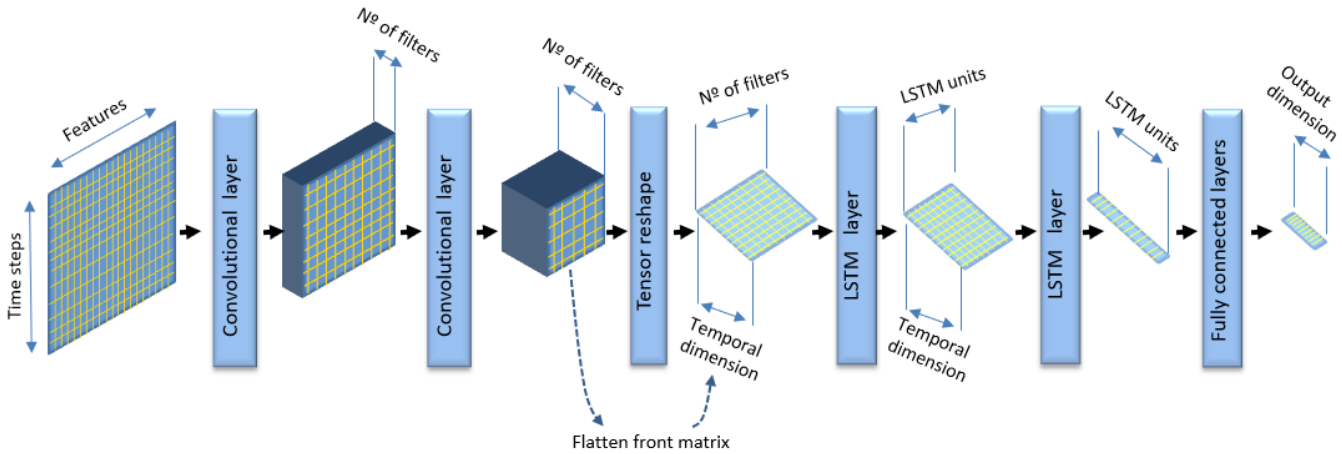


Fig 5. Combination of two-layers CNN and two-layers RNN model. Architecture of the best CNN-LSTM network applied to QoE prediction

The architecture presented in Fig 5 corresponds to the best one obtained in [4] for QoE prediction. Figs 6-8 present other possible configurations with different number of CNN and LSTM layers. Fig 8 shows the architecture that provides best results for the research performed on NTC prediction [2]. We can see that the number of layers has an impact on the results obtained and that increasing this number does not always improve them.

In addition to the layers shown in Figs 5-8, other layers have been considered for this architecture: Max Pooling, Batch Normalization and Drop-out layers [2][4]. Max pooling provides a further dimensionality reduction in the generated tensors apart from additional properties of invariance. Batch normalization improves the stability of the learning process and, sometimes, makes it faster. Regularization using drop-out is a very effective way to avoid over-fitting for neural network architectures. These additional layers were tested in the experiments conducted for the thesis, but were not included in the final models, with the exception of the drop-out layer that was used to improve generalization (avoid overfitting).

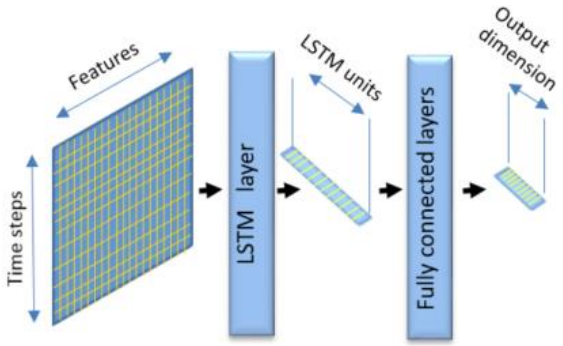


Fig. 6. Simple one-layer RNN model

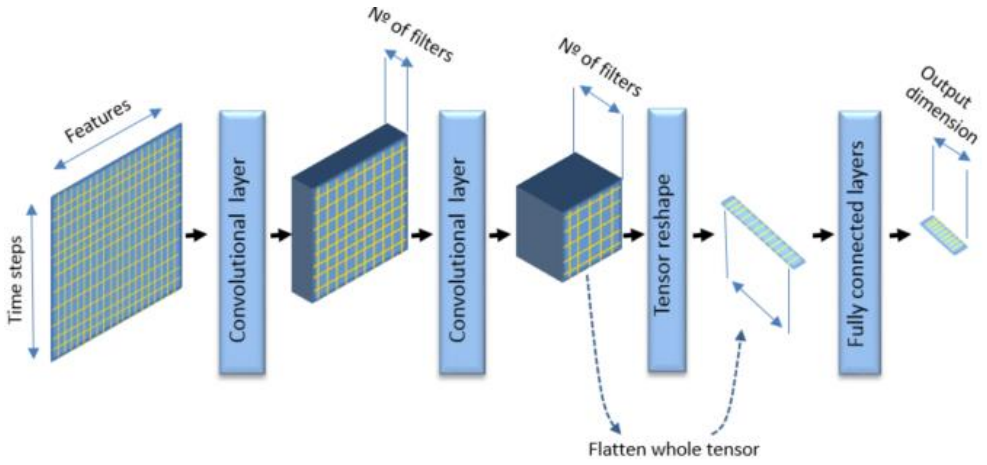


Fig. 7. Simple two-layers CNN model

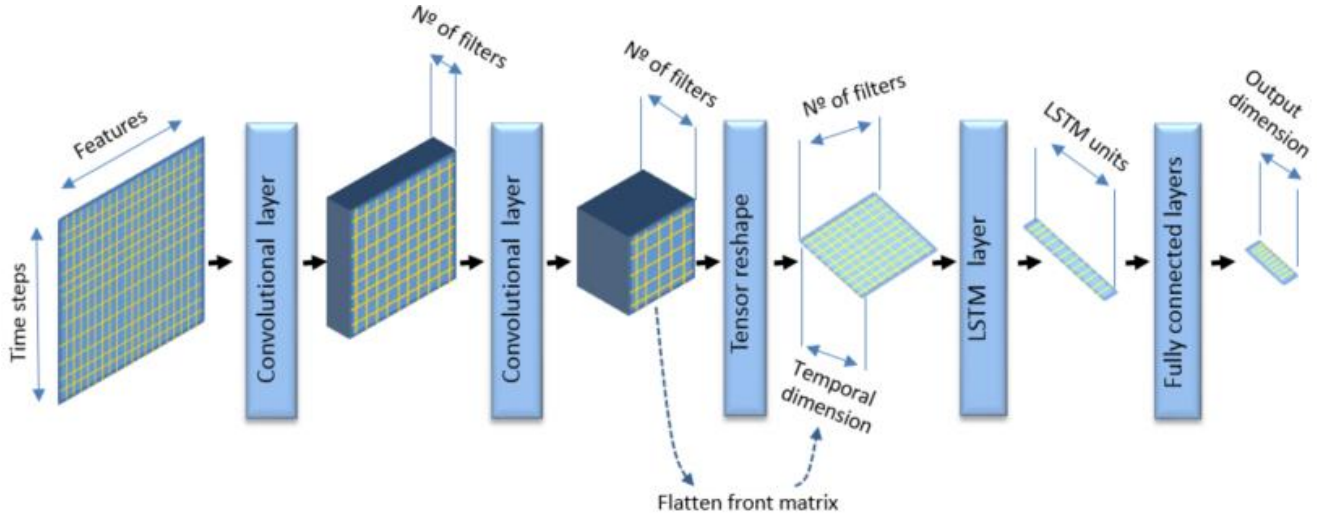


Fig. 8. Combination of two-layers CNN and single-layer RNN model. Architecture of the best CNN-LSTM network applied to NTC prediction.

The preparation of data was critical to achieve the final results in [2][4], this preparation consisted, in both cases, in transforming a sequence of information extracted from network packets into 2D-matrix-like structures that could be processed by a CNN layer.
In Fig. 9 the data arrangement for [2] is presented, where specific parts of the packets header are arranged in sequence, resulting in a 2D matrix structure. Similarly, in Fig 10 is presented the data preparation for [4], which is more complex because in this case we do not make a single prediction, but a whole set of predictions for the two-time-ahead forecasting of the 7 QoE errors that are examined.

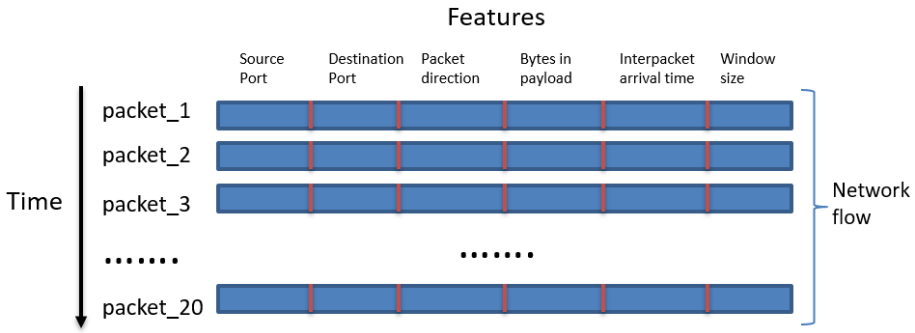


Fig. 9. Composition of a network flow (NTC prediction)

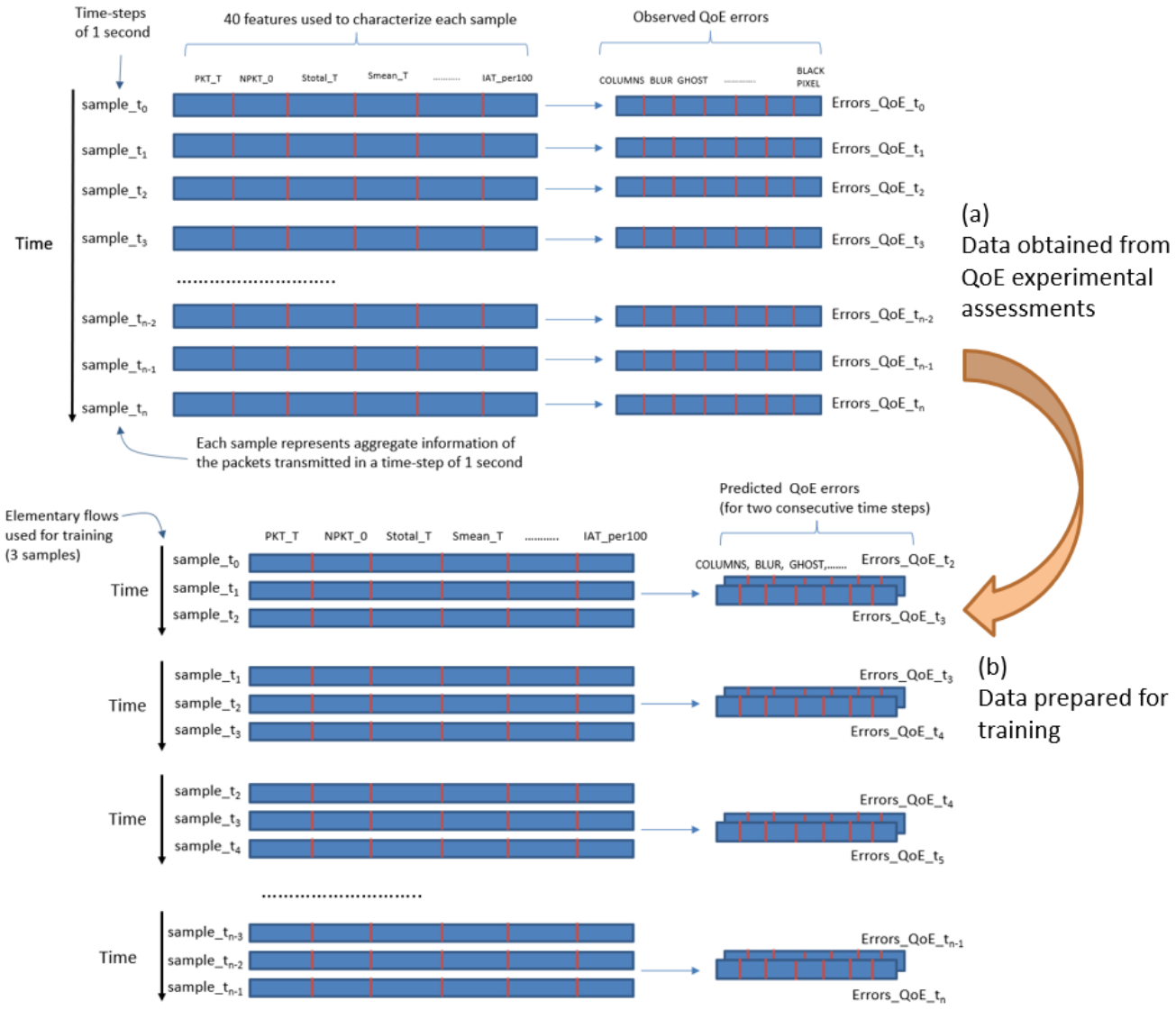


Fig 10. Features transformation for QoE prediction.

4.2 Gaussian processes

It is a difficult task to apply deep learning models to problems with scarce training data. In this case, it would be interesting to try additionally other ML methods that make full use of all available data. Bayesian models are particularly suitable for this situation and, in particular, the models based on Gaussian Processes (GP) [133]. This is the reason to test the suitability to combine a deep learning architecture with a GP model. This has been done in [4], where a small amount of training data was available to train the QoE classifier.

GP models are generally applied to regression problems, but they can also be used for classification. A GP Classifier [134] is based on the so-called Laplace approximation [134], which tries to approximate with a Gaussian function a non-Gaussian posterior formed by applying a logistic link function to the output of an intermediate latent function. The resulting squashed outcomes produced by the link function are associated to classification probabilities. The kernel [133] chosen has been a Radial Basis Function (RBF) kernel with two adjustable parameters: lengthscale and variance. To train the GP Classifier consists on tuning these two parameters plus an additional noise variance parameter associated with the likelihood of the model.

When applying the GP Classifier in [4] we have used the architecture in Fig 5 as our initial network, then using one of the last layers of this network (already trained) as the input to the GP Classier (Fig. 11). With this configuration, we have trained the GP Classifier by adjusting the lengthscale and variance parameters of the RBF kernel used in this case.

Considering the difficulties to apply a multi-label GP classifier [134], in [4] we have applied 14 independent GP binary classifiers: one per label (7) and prediction time-step (2). The GP classifier is a non-parametric algorithm that makes full use of the data available, which is precisely the reason to try this model in [4], since to train the QoE classifier we started with a reduced amount of training data.

Nevertheless, the increase in performance obtained with the addition of the GP classifier is quite small and is not worthy of consideration, especially given the additional memory and time processing required by adding the GP classifier. In particular, in [4] we obtain and increase in the aggregated F1 measure (for all labels and time-ahead steps) from 0.6965 to 0.6987 (0.3% increase).

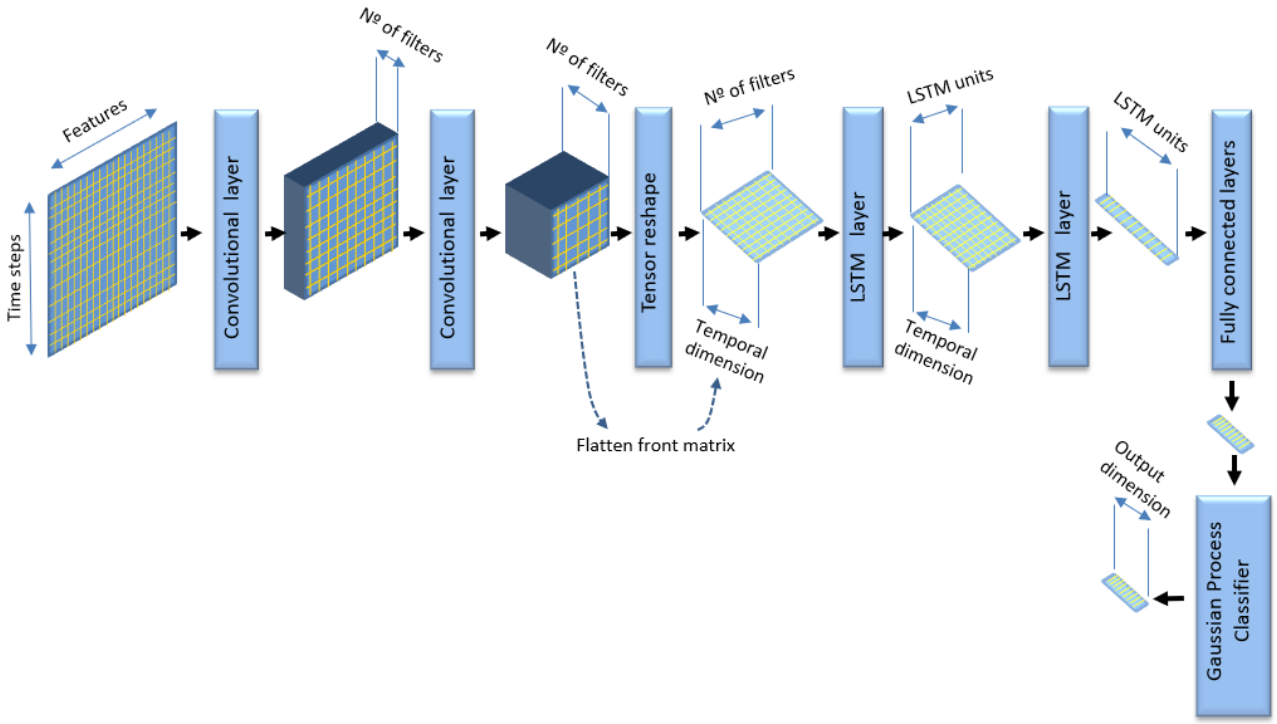


Fig 11. Application of gaussian processes as a final layer for the combined CNN-RNN model (QoE prediction)

4.3 Conditional variational autoencoders for classification

It is paradoxical to see the little attention that generative models have attracted in networking [8], considering their successful application in other areas, mainly with the advent of the variational autoencoder (VAE)[23] and generative adversarial networks (GAN)[135] algorithms. This is the reason to consider the application of one of these models (VAE) for this thesis.

A generative model can be used to create new synthetic data, to correct/improve available data or as a mean to build classifiers/regressors. In this section we will cover the use of a VAE as a classifier [3] and in the following section how to use it to synthesize and correct data [5].

As noted in section 3.2.1 the application of a supervised model for classification can be done in three ways applying different methods: probabilistic methods, clustering methods or deviation methods. Deviation methods define a generative model that can reconstruct normal data, and any data that is reconstructed with an error greater than a threshold is considered an anomaly. Deviation methods are those used when applying a VAE for classification.

When using a VAE to build a classifier it is necessary to create as many models as there are distinct label values, each model requiring a specific training step (one vs. rest). Each training step employs, as training data, only the specific samples associated with the label learned, one at a time. At test time, the comparison of the errors obtained when re-generating the features of the data using the different models will indicate which is the correct label (the one associated with the model that produces the least error).

The novel approach in this thesis has been to use a Conditional VAE (CVAE) [24][25] instead of a VAE to perform classification. This is, as far as we know, the first time that a CVAE is used for this topic.

The main difference between a CVAE and a VAE is presented in Fig 12. In a CVAE we employ the label (in our case, the intrusion label) as an additional input to the decoder at training time. This additional information allows the network to learn the association of features to labels. This also allows the training of a single network to be sufficient to perform the classification, while a VAE needs as many different networks as the number of label values. At test time, we simply have to run the network forward with the features and each possible label value (one run per label value) and to compare the reconstruction errors obtained for each run (Fig 13). Therefore, a CVAE needs to create a single model with a single training step, employing all training data irrespective of their associated labels. This is why a classifier based on a CVAE is a better option in terms of computation time and solution complexity. Furthermore, it provides better classification results than other familiar classifiers (random forest, support vector machines, logistic regression, multilayer perceptron), as shown in [3].

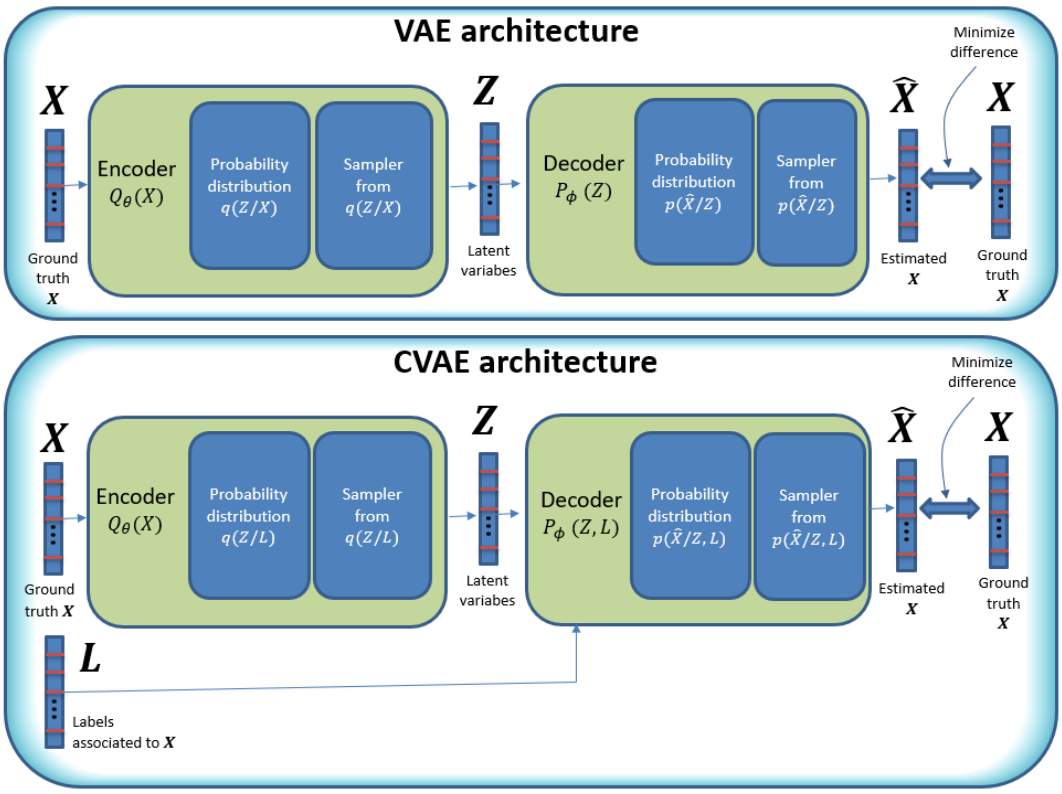
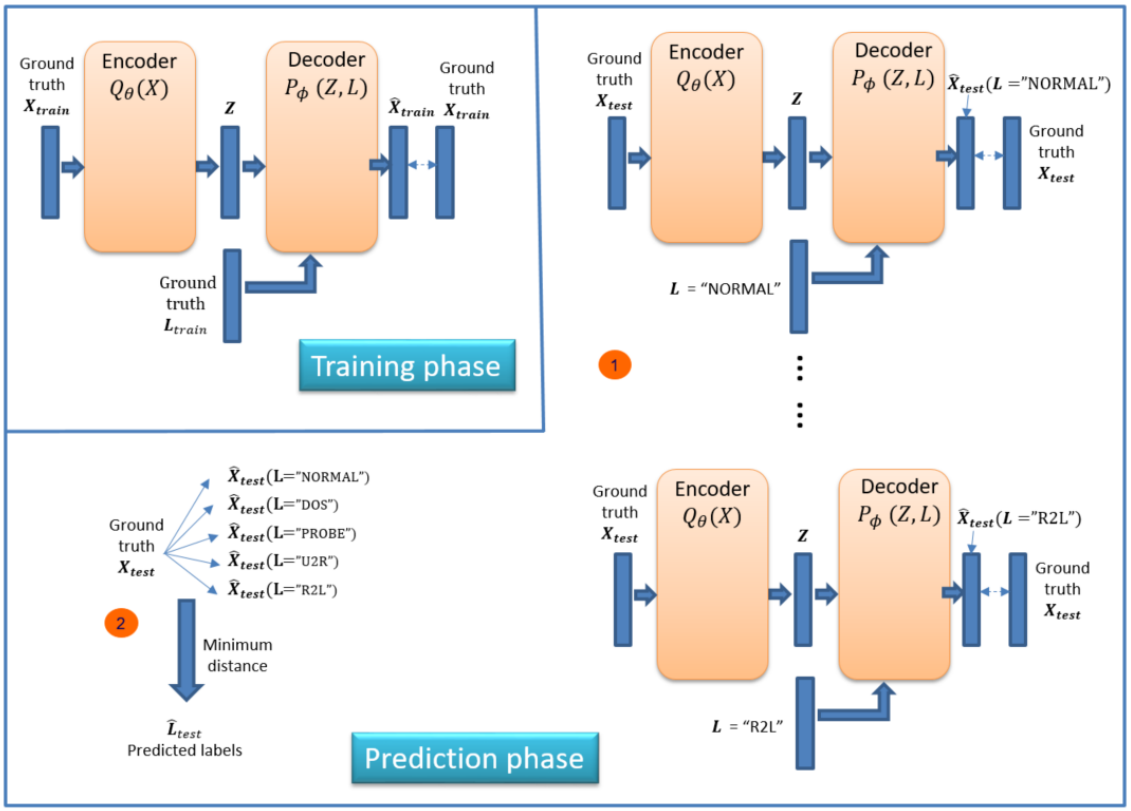


Figure 12. Comparison of CVAE with a typical VAE architecture.



.

Figure 13. Classification framework.

In [3] is presented in detail the CVAE architecture for classification. It is also important to consider the different alternatives for the probability distributions of the latent (intermediate) layer and the final layer. In the experiments carried out in [3] we have considered a Gaussian distribution for the latent layer and a Bernoulli distribution for the last layer. In the next section it will be shown that for [5] we explored other possible configurations for these distributions.

4.4 Conditional variational autoencoders for practical data synthesis

As discussed earlier, a generative model can be used to create synthetic data following the probability distribution of some real features, either continuous or discrete. We propose for this thesis to employ a new generative model based on a CVAE to synthetize new data [5] and to reconstruct missing data [3].

The new method (based on CVAE) offers operational advantages (speed, simplicity) and quality in the synthesized data (similar probability distributions and improvements in their properties) compared to the usual over-sampling algorithms (e.g. SMOTE).

The main difference between using a CVAE and SMOTE (and its variants) is that a CVAE is based in a latent probability distribution learned from data, instead of being based in a predefined 'distance' function. A CVAE does not need to assume any 'distance' function, or to impose rules on the importance of proximity to majority class samples, which would be additional hyper-parameters to explore.

In order to achieve the best possible generation results, we have explored different probability distributions for the latent and final layers and different loss functions used to train the network.
Fig 14 presents the best architecture [5] obtained with a Gaussian distribution for the latent layer and a Bernoulli distribution for the last layer, with a loss function formed by adding the log-loss of the probability distributions for the final layer with the Kullback-Leibler divergence between a standard normal prior and the Normal probability distribution parametrized by the vector of means and variances produced by the latent layer. In Fig 14 are shown the different components of the loss function. Many other configurations were tried before arriving to this selected architecture, in particular: 1) mean square error (RMSE) instead of the log-loss for the loss function, 2) different loss functions for the continuous and discrete features, 3) to use the label as the input to the encoder, instead of the features (X).

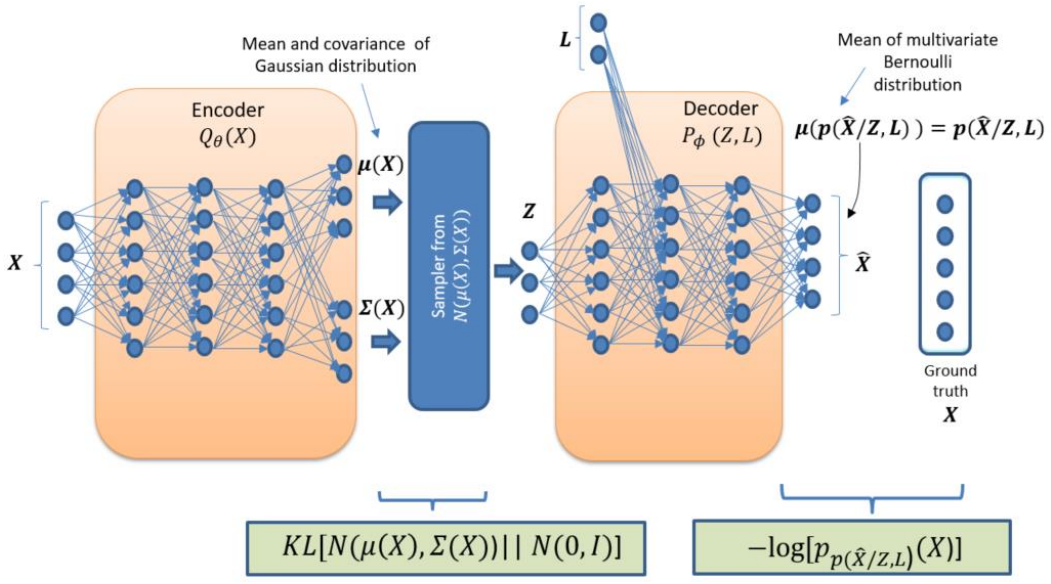


Fig 14. CVAE model including the labels in the decoder with Gaussian and Bernoulli distributions. Training phase.

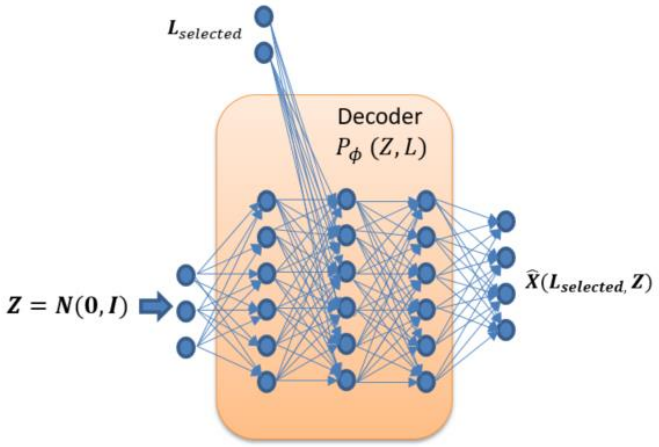


Figure 15. CVAE model including the labels in the decoder with Gaussian and Bernoulli distributions. Generation phase.

Fig 15 shows a CVAE at test time (generation phase) [5]. Once the CVAE network is trained, we no longer require the encoder block. In order to generate new synthetic data, we apply random inputs (following a standard normal distribution) to the decoder plus the label value (one-hot encoded) as an additional input. This additional input is concatenated to the nodes of an intermediate layer of the decoder block. The decoder output will be our new synthetic data. This synthetic data will follow the probability distribution of the features of the real data associated to the label value provided at generation time.

In [5] other additional challenges were found, in addition to the adequate generation of synthetic data, since evaluating that synthetic multivariate data follows the same probability distribution of real data turned out to be a difficult task for which we had to develop several approaches: (1) extended histograms of the original and synthesized features; and (2) the analysis of classification results obtained from the application of original and synthesized data to several classification algorithms. The second point is especially important, because we assumed the hypothesis that two datasets are similar if they deliver similar prediction metrics when several classifiers use their data indistinctly, therefore, we try to show that we have similar classification accuracies when we do predictions with the original or the synthetic datasets without distinction, by using the predictions obtained with several ML classifiers: Random Forest, Logistic Regression, SVM and Multilayer Perceptron (MLP).

The paradigm followed to reconstruct missing data [3] has been different. In this case we have followed a similar approach to the one shown in Fig 13 and it is presented in detail in [3]. In this case, we have used a discriminative approach, and, for each missing feature, we selected the value that produces the smallest reconstruction error considering all the features.

4.5 Machine learning for time-series prediction

As described in [52], the majority of security data and most of networking data fall into the category of time-series data. Time-series consist of event data with a defined temporal order i.e. they have a sequence structure. The time dimension can be continuous or discrete, similar to the event data that can also be formed by continuous or discrete values of vectors or scalars. In the case of event data formed by vectors, the vectors may contain only continuous variables, only discrete variables or a mixture of continuous and discrete variables.

The datasets used for ML in networking are often made up of time-series data. The time-series can be associated with the predictors, the expected outcomes or both. The data generation and preparation processes are responsible for the necessary discretization, aggregation and scaling of the different variables, whether they are predictor variables (features) or predicted variables (outcomes).

The discretization and aggregation stage is particularly important in the data preparation process. For example, in [1] we propose a k-ahead forecasting problem where the inputs are time-series of categorical values (on/off connectivity of mobile devices aggregated in time-slots of 1-hour) and the outcomes are also a time-series of categorical values. In [2] the inputs are time-series formed by a sequence of up to 20 packets. For each packet, six features are extracted (source port, destination port, the number of bytes in packet payload, …). That is, the inputs are time-series of vector values (categorical and continuous) and the outcome is a single categorical value (scalar) that corresponds to the traffic type of the network flow. In [4] the inputs are time-series composed of a sequence of 3 vectors consisting of 40 continuous features. The features correspond to aggregate information taken in time-slots of 1-sec from network packets captured in PCAP format. The outcome is also a sequence of 2 vectors associated with the presence/absence (categorical) of 7 QoE errors for the next 2 time-steps. That is, in this case we have two time series of vectors with continuous and categorical data for predictors and outcomes, respectively.

Considering traffic prediction, in [1] we have provided a review of the application of time-series and non-time-series models to the prediction of traffic of an IoT mobile network, using real data from a main Telco Operator in Spain. In this study we have evaluated the following time-series methods: Hidden Markov Model (HMM), Exponential Smoothing, ARIMA and ARIMAX. And the following non-time-series methods: Logistic Regression, Random Forest, Gradient Boosting Method (GBM) and Bayesian Logistic Regression. Being the first time, as far as we know, that the ARIMAX and Random Forest models are applied for traffic prediction. The study provides new insights comparing the results for time-series and non-time-series methods, applying the methods to a large number of devices with very different connectivity behaviors. In particular, it is interesting the data transformation that was required to obtain a single data structure that could be used by time-series and non-time series models, since, in principle, both types of models require different data structures (Fig. 16).

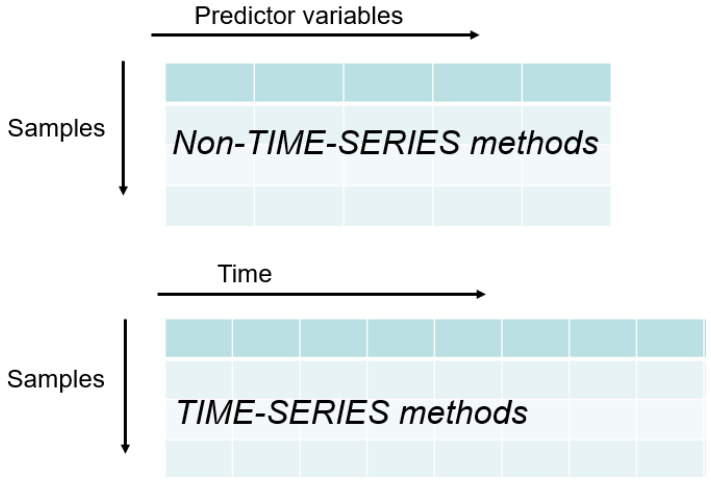


Fig 16. Different data structures for time-series and non-time-series models.

5. CONTRIBUTIONS AND LESSONS LEARNED

5.1 Contributions

The following table presents the main contributions provided by this thesis.

Objective/Area	Contributions
Intrusion detection	- Proposal of a new model (ID-CVAE), which is essentially an unsupervised technique trained in a supervised manner thanks to the use of class labels during training. - First application, as far as we know, of a conditional VAE to perform intrusion detection. - ID-CVAE integrates the intrusion label in the decoder layer, which results in a less complex model than an equivalent model that exclusively uses a VAE and with a better detection performance. - For ID-CVAE, the classification process only requires one single training stage followed by as many test stages as distinct values we try to predict. A VAE would require as many training and test stages as there are distinct values label values. Considering that the training phase is the most costly, we can see the improvement in performance obtained by using a conditional VAE (CVAE) - With ID-CVAE for classification we obtain an accuracy over 80% for the NSL-KDD 5 labels scenario, which is better than the values obtained from Random Forest, Linear SVM, Logistic Regression and MLP
Type of traffic classification	- The natural domain for a CNN, which is image processing, is expanded to Network Traffic Classification (NTC) in an easy and natural way. It demonstrates that a CNN can be successfully applied to NTC classification, giving an easy way to extend the image-processing paradigm of CNN to a vector time-series data (in a similar way to previous extensions to text and audio processing). - First application, as far as we know, of a CNN+RNN model to an NTC problem. - It is shown that a RNN combined with a CNN provides better detection results than alternative algorithms without requiring any feature engineering, usual when applying other models. - A robust model that gives excellent F1 detection scores under a highly unbalanced dataset, with over 100 different classification labels is provided. It works with a very small number of features and does not require feature engineering. The model is trained with high-level header-based data extracted from the packets. It is not required to rely on IP addresses or payload data, which are probably confidential or encrypted.
Traffic prediction	- Results of applying machine learning techniques to forecast the on-off activity state of IoT mobile devices, using time-series and no-time-series methods, are presented. Data from real IoT mobile devices is employed. It provides new insights comparing the results for time-series and non-time-series methods, applying the methods to a large number of devices with very different connectivity behaviours. - Data pre-processing to present the data in a form that could be used by both time-series and non-time-series methods. - Test results were achieved with a specifically developed cross-validation process, applied to both, time-series and non-time-series methods. - Novel results obtained from the application of random forest. - Previous works on this subject have focused on the prediction of data volume transmitted (which is a continuous variable), however this paper focuses on

	predicting on/off connectivity (a discrete variable), which constitutes a new starting point and a different issue itself, providing novel results. - It is proven that a mixed method (ARIMAX) provides the best accuracy (93%) but requires a huge training time. ARIMA and some non-time-series methods (logistic regression and random forest) with accuracy over 90% also provide very good performances. Considering the higher computational requirements for ARIMAX compared to logistic regression, random forest and ARIMA; the latter methods would be a better choice for a production environment (as they provide similar practical accuracy with less computing time).
QoE estimation	- It is demonstrated that a CNN can be applied to a time-series of samples (formed by aggregated information from network packets) to predict QoE for video transmission. - The proposed model can be integrated into a network management system to monitor network quality (as observed by the end-user), which is an essential part of a self-adapting network (e.g. SDN, edge computing...). The model is applicable to a real-time environment (in time-steps of 1-second) and is able to predict video QoE for current and near-future video transmissions. - The best proposed model includes a combination of CNN and RNN networks, being the CNN network the most critical piece. This is somewhat surprising given the time-series nature of the data that was formed by adding 3 samples of elementary flows (which consist of aggregated information from networks packets taken in a 1 second period). - Excellent prediction performance for not extremely unbalanced labels with a small dataset
Synthesize training data to improve classification	- First application, as far as we know, of a conditional VAE to generate fully synthetic network traffic data - Application of the synthetic data to an intrusion detection problem, which shows that, when training an ML algorithm with the new synthetic data, the detection results obtain a substantial improvement. This improvement is greater with the synthetic data generated by the proposed method in comparison with the results obtained by training with synthetic data generated by alternative SOTA over-sampling methods: SMOTE, ADASYN, … - Innovative methods to assess the similarity of the probability distributions of features for the real and synthetic data are proposed. - Analysis of different variants of a VAE architecture for the proposed model, providing an extensive study on the alternatives. - The new method allows to synthesize the new samples just knowing the intrusion label to which the synthetic data should belong, with the advantage of not relying on specific samples associated with the labels. This association is usually noisy and identifying a canonical set of samples associated with each label can be complex. Therefore, the proposed model streamlines the data generation process based exclusively on the intrusion label.
Synthesize missing data	- First proposal, as far as we know, of a feature reconstruction model using a conditional VAE and first application for intrusion detection. - General framework available for other areas that may need a technique for feature reconstruction and imputation of missing values - Generative method that learns the probability distribution of the features conditioned on the label value. Inclusion of the label value to obtain the probability distribution of the features. - The model achieves excellent accuracy for the recovery of features: over 90% accuracy for labels with around 10 values and over 70% for labels with around 70 values. The accuracy is based in the correct prediction of all these values. - Intrusion data is strongly unbalanced, noisy and with numerous continuous and categorical features, which is a challenge to synthesize intrusion data with a probabilistic structure similar to the original one. That is the reason why we generate synthetic samples conditioned to the distribution of labels. That is,

	from a particular set of labels, we generate training samples associated with that set of labels, replicating the probabilistic structure of the original data that comes from those labels. In this way, we obtain the probability distribution of P(X, Y) instead of P(X) (Section 3.1.3).

Table 7. Contributions of the thesis to the research areas

5.2 Lessons learned

The following table presents the lessons learned from the research carried out for this thesis:

Objective/Area	Lessons learned
Intrusion detection	- An unsupervised algorithm (VAE) can be used for intrusion detection providing better results than classic supervised ML methods: random forest, SVM, MLP and logistic regression. - Very simple encoder and decoder networks (3 layers only) are enough to obtain best results. Increasing the number of layers does not improve results. - Using a conditional VAE instead of a VAE provides many advantages in terms of a faster classification algorithm. - VAEs and conditional VAES present robust and easier training than alternatives that do not use variational methods.
Type of traffic classification	- In a prediction/detection problem related to time-series of vectors, in addition to using an RNN which is the natural choice given the time-series nature of the problem, it is a good strategy to include a CNN as an initial step in a deep learning architecture. A CNN, in addition to performing feature engineering, is able to extract time patterns that are useful when they are subsequently processed by the RNN. - When a sequence of packet headers is used to predict the type of traffic of a network flow, it is not necessary to deal with the entire sequence; a small number of packets are enough to make the prediction with high accuracy.
Traffic prediction	- One week of historical data is enough to provide good forecasts - Independently of the method, the on-off connectivity from IoT devices presents a rich periodic structure, allowing good prediction results, even with short training data. - The non-time-series methods require, in general, less training time than the time-series-methods. - From the results obtained, we expect in future works to find additional predictors (covariates) that will probably improve the predicting power of the methods presented. One of these new predictors could be obtained by performing clustering of the signals, trying to use the cluster index as an additional new predictor. The main problem will be the high computational demand for this task. - It is interesting that logistic regression which is a very simple model can provide an accuracy comparable (in practical terms) to more sophisticated models (e.g. ARIMAX, Random Forest, ARIMA,..).
QoE estimation	- It is possible to apply new deep learning models whose origin focused mainly on the areas of video, audio and language processing for the prediction of QoE of transmitted videos. - We extended the study with the inclusion of a GP Classifier that, being a non-parametric model, could make full use of the scarce data available. This inclusion provides a slight improvement in the prediction results. In addition, it requires much more memory and processing time, which makes it less useful than expected. - The performance does not increase when increasing the number of samples per flow beyond three, implying that prediction is conditioned on the amount of previous information, but information too distant in time is not only less useful but indeed a problem. Similarly, reducing the number of samples below three also decreases the prediction performance. - From this experience, we plan to investigate the application of generative models (e.g. variational autoencoders) to create synthetic data and explore

	one-shot learning advances.
Synthesize training data to improve classification **Synthesize missing data**	- A VAE with an architecture adequately tuned provides better data over-sampling/synthesis results than alternative methods (SMOTE, ADASYN...) - Very simple encoder and decoder networks (3 layers only) are enough to obtain best results. Increasing the number of layers does not improve results. - Using a conditional VAE instead of a VAE provides many advantages in terms of the ability to generate features conditioned on the classification labels. - VAEs and conditional VAEs present robust and easier training than alternatives that do not use variational methods.

Table 8. Lessons learned

6. METHODOLOGY

In this section it is explained the approach taken for the different papers. All papers have similar points of interest: **objectives**, **datasets**, **models** and **results**. In the following sections these **common points of interest** are analysed in detail for the three papers.

In Fig. 17 is provided the workflow connecting all points of interest for the papers. It is depicted in a model diagram, with arrows defining the relationship between the points.

We can see in Fig. 17 that the **objectives** established at the start of the research *define* the **datasets**, and likewise the **datasets** finally available can *constrain* the achievable **objectives**. Once the **objectives** and **datasets** are established, both are crucial to *select* the possible applicable **models**. The different **models** that are finally chosen may have specific requirements in the *format* of the **dataset** used. Different combinations of **models** and **datasets** will normally *generate* different **results** that will ultimately *confirm* (or not) the **objectives** set at the beginning.

The influence of the objectives to define the datasets is clear in all papers since all of them make use of a dataset specifically suited for the mission. The constraint imposed by the datasets on the objectives is also clear, for example in [3] where we had to limit the intrusions that could be detected to the ones provided by the NSL-KDD dataset.

In all cases, we have tried models appropriate to the objectives and available datasets. For example, in [1] where the objective is to predict the future activity of IoT devices with a dataset of time-series of past activity, we applied time-series algorithms and re-formatted the dataset to allow the use of other algorithms initially not suitable for time-series data. In [2], we have made possible an innovative application of deep learning algorithms to the detection of the type-of-service by formatting the data appropriately and using the specific models more suited to the detection problem (CNNs for representation learning and LSTMs for time-series information processing). In [3], since intrusion detection is a stochastic problem, we wanted to try models specifically suited to learn the probability distribution of intrusions and their co-occurrence with the available predictors (features), which is the reason to try a generative model based on a conditional VAE to an intrusion detection problem. In [4], where the goal is to perform QoE estimation using information extracted from network flow packets, in order to generate the data that the QoE prediction models will use, it was necessary to establish an experimental setup that would allow identifying the QoE of the video transmissions while recording the associated network flow packets. The resulting data are multivariate time series, in time-steps of 1-second, which contain the network packets transmitted in each time-step plus the presence or absence of seven video transmission errors in that time-step. Finally, in [5] where the objective is to generate synthetic data that could be used as new training data for an intrusion detection classifier, we selected the NSL-KDD dataset which has a large number of continuous features and three categorical features with high cardinality, this imposes a higher difficulty to the problem of data generation what makes this dataset appropriate for the research work. In this case we had to perform a scaling of the continuous features and a one-hot encoding of the discrete features which was needed to apply a VAE to this problem.

In relation with the results reported by the papers, we have presented separate results for all the models investigated in the study. A detailed comparison between the proposed and alternative models is always provided in all papers.

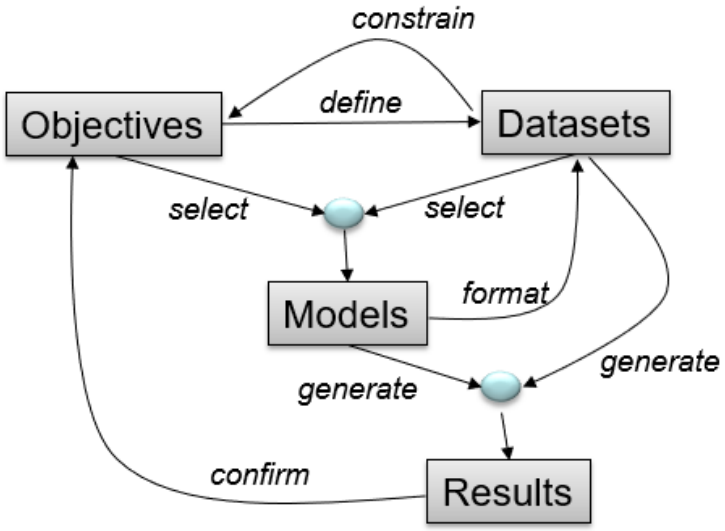


Fig. 17. Model of relationships between the main “points of interests” for the papers.

All the papers that make up this thesis employ the workflow shown in Fig. 17. Similarly, all papers follow the schema of datasets management presented in Fig. 18. It is very important to mention that all reported results, in all papers, are provided in accordance with Fig. 18, therefore, all results are obtained using a separate test dataset that was never used in the training phase of any of the models that have been part of this research.

In Fig. 18 is observed how the original data is split between training and test subsets. The training subset can be further split in a training part and a validation part. This latter division of the training dataset is used when the successive parameters tuning of the model can affect the generalization results if all the validation would be done on the test dataset. This good practice has some limitations, mainly due to the large volume of data required to have the three separated datasets. For this reason, we have not employed a validation dataset. In addition, the validation dataset is mandatory when the parameters tuning is thoroughly done, trying a substantial number of parameter values. In our case, we have done a basic parameter tuning of the algorithms, since we have not had the intention of providing the best possible result, but to make clear the applicability, the possibilities and the good behaviour of the proposed models, and not fitting a final model to be used in an industrial setting.

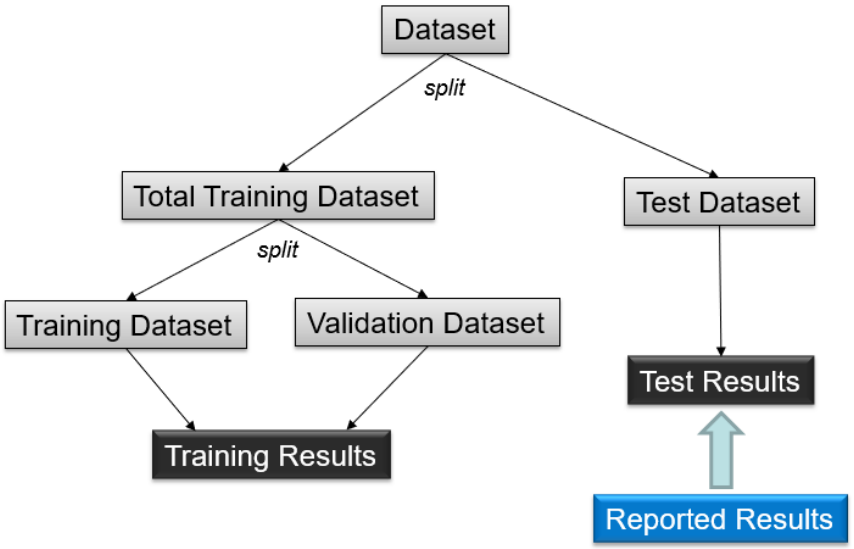


Fig. 18. Schema of datasets and their management to obtain the reported results.

As an anticipation of the more detailed description of the main points of interest for the different papers, which is provided in the following sections, we can see in Fig. 19 and 20 an outline of the different datasets and models that are used in the papers.

In Fig. 19 are presented the datasets used. We can see that two of them correspond to actual operational data obtained from real Operators or Service Providers, and the third is a well-known intrusion detection dataset (NSL-KDD) that has been used for many similar research studies. Due to the difficulties associated with obtaining real data for video QoE estimation, we had to obtain our own dataset of video QoE scores from real individuals.

The main reasons to choose NSL-KDD as the dataset for intrusion detection have been: (a) its availability, (b) the large number of research works carried out with it that facilitate the comparison of the results with the methods proposed in this thesis, and (c) its relatively reduced size that avoids the need for large computational resources. In particular, we would like to mention the UGR16 [136] dataset, which is a modern and very relevant dataset for intrusion detection that was not included in the research due to its large size.

The use of real data to conduct studies is an important advantage. It makes the results more realistic too. In addition, the "No free lunch theorem" [137] states that, on average, all algorithms are similar when faced with all possible problems, therefore, the difference between algorithms is their ability to provide a performance advantage for a particular kind of problem. With this in mind, having a realistic dataset that resembles a real environment is important in order to adapt the best algorithm to the desired real problem.

For intrusion detection, the availability of realistic data is problematic due to the stochastic nature of the intrusions, their low frequency of appearance, the large number of possible types of intrusions and their dependence on the nature of the network attacked. For these reasons we have used the NSL-KDD dataset as a representative dataset for intrusion detection. The NSL-KDD [67] dataset is a derivation of the original KDD 99 dataset. It solves the problem of

redundant samples in KDD 99, being more useful and realistic. NSL-KDD provides a sufficiently large number of samples. The distribution of samples among intrusion classes (labels) is quite unbalanced and provides enough variability between training and test data to challenge any method that tries to reproduce the structure of the data.

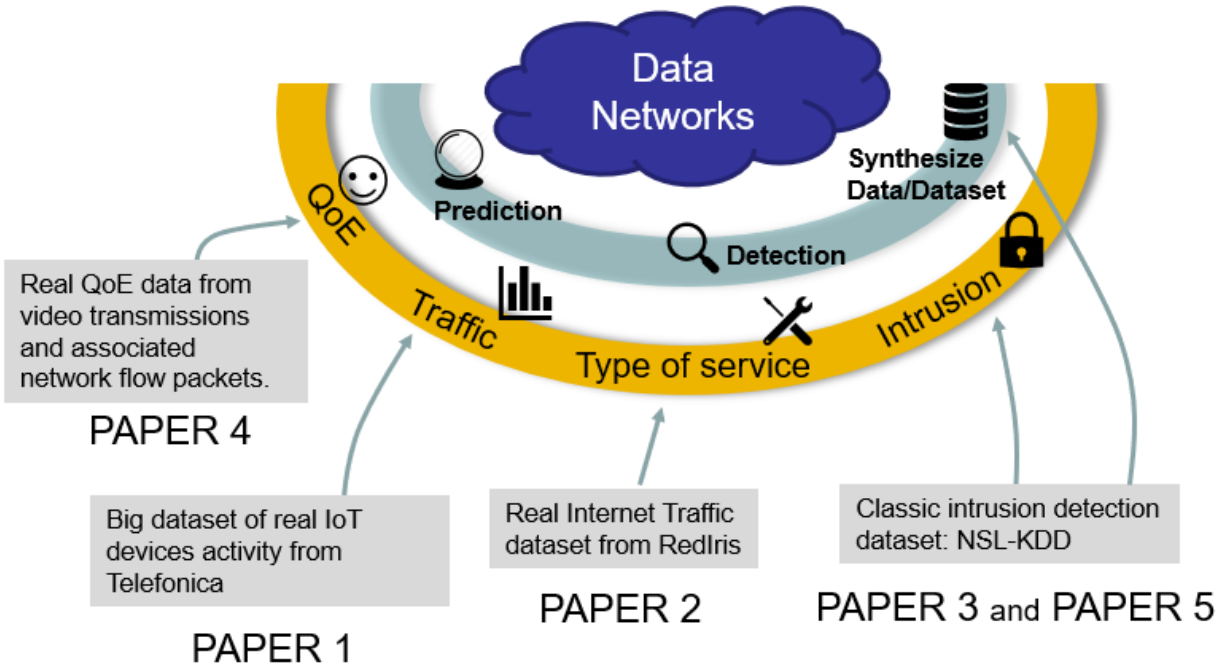


Fig. 19. Overview of datasets used in each paper

In Fig. 20 are shown the different main models proposed in the papers. In addition to these models, other models have been considered not as part of the main research activity, but to provide a comparison of results. For example, in [3], the results obtained from the C-VAE model are compared with the results of several classic machine learning models such as Random Forest, Support Vector Machine (SVM), Logistic Regression and Multilayer Perceptron (MLP). Similarly, in [5] the synthetic data generated by the proposed model is compared with other over-sampling algorithms: SMOTE, SMOTE Borderline, SMOTE ENN, SMOTE Tomek, SMOTE SVM, Easy Ensemble and ADADSYN. In this case to compare the properties of the synthetic data we also needed to use several ML algorithms to check with them that the new data was able to improve the algorithms training; for this task we used 4 well-known ML algorithms: random forest, MLP, logistic regression and SVM.

Most of the algorithms proposed in this thesis and mentioned in Fig. 17 are deep learning algorithms. For example, in [2] we use a combination of CNN and LSTM (a variant of RNN) networks to detect the type-of-service of a network flow; and, in [3], we propose a variant of a VAE which is called a conditional VAE (C-VAE) to perform prediction and generate synthetic features for intrusion detection. In [4] we also propose a final classifier based on a combination of CNN and LSTM networks.

In all these cases, it has been a challenge the application of deep learning algorithms:

For [2], it is the first time, as far as we know, that a CNN+LSTM network is applied to an NTC problem. A CNN network is mainly intended to deal with image data, and the application to NTC has been possible with our initial intuition to assimilate the vector time-series extracted from network packets as an image. The good results obtained confirm that the initial intuition was correct, and therefore CNNs are valid candidates for dealing with vector time-series.

Similar reasoning can be applied to [4] which also presents the application of a CNN+LSTM to a video QoE problem. In this case, it was necessary to prepare the dataset to provide information chunks that could be assimilated to images. In this case, it was also an interesting discovery to observe the importance of the CNN network, which is more critical, for prediction performance, than the RNN network, which is curious given the time-series nature of the data also in this case. For this work, we also used a Gaussian process classifier as the final layer of the entire model. The interesting finding in this case was to observe that by adding this last layer the performance improved, but in a non-significant way.

In the case of [3], it is also the first time, as far as we know, that a C-VAE is applied to an intrusion detection problem. The inclusion in the network of the intrusion label (in the C-VAE case) is particularly important for intrusion detection since it generates a resulting model which is less complex than other classifier implementations based on a pure VAE. The model operates creating a single model in a single training step, using all training data irrespective of their associated labels, while a classifier based on a VAE needs to create as many models as there are distinct label values, each model requiring a specific training step (one vs. rest). Training steps are highly demanding in computational time and resources; therefore, reducing its number from n (number of labels) to 1 is an important improvement. In addition, the model is also able to perform feature reconstruction, for which there is no previous published work. Both capabilities can be used in current Network Intrusion Detection Systems (NIDS), which are part of network monitoring systems, and particularly in IoT networks [34].

Similarly, [5] provides the application of a C-VAE to generate synthetic data in networking for an intrusion detection problem. To arrive to the best architecture for the proposed model it was necessary to check: a) several network configurations, b) options on prior probability distributions for the latent and final layers and c) to consider alternative loss functions.

The work in [1] is an exception in terms of models applied since we have not used any deep learning model. In this case, we have applied a set of well-known classic machine learning models (Random Forest, Logistic Regression...) to a time-series prediction problem that is usually treated with time-series algorithms (ARIMA, Hidden Markov Model-HMM,…). The challenge has been in transforming a time-series dataset to a cross-sectional format suitable for the non-time-series models and selecting the appropriate features in the new formatted dataset.

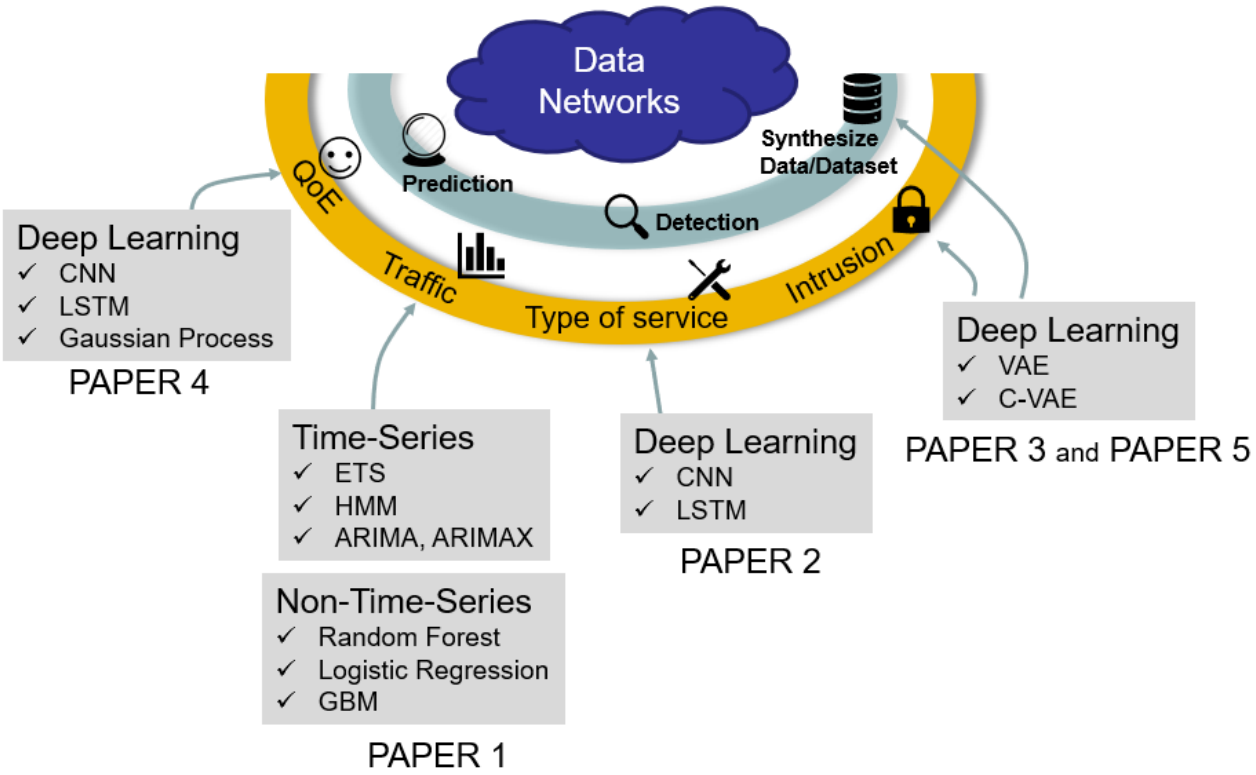


Fig. 20. Overview of algorithms used in each paper

7. PAPERS SUMMARY

The following sections provide a summary of the main points of interests of the papers:

7.1 Paper 1: Review of methods to predict connectivity of IoT wireless devices

Authors	Manuel Lopez Martin, Antonio Sanchez-Esguevillas and Belen Carro.
Title	Review of methods to predict connectivity of IoT wireless devices
Journal	Ad Hoc & Sensor Wireless Networks, Volume 38, Number 1-4 (2017), p. 125-141
Impact Factor	1.034
Quartile	Q4
#Citations	2 (https://scholar.google.es/citations?user=3RSZbOYAAAAJ&hl=es)
Status	Published, July-2017
Link	http://www.oldcitypublishing.com/journals/ahswn-home/ahswn-issue-contents/ahswn-volume-38-number-1-4-2017/ahswn-38-1-4-p-125-141/

7.1.1 Objectives

The main objective of this study has been the identification of machine learning techniques to forecast the on-off activity state of a large number of IoT mobile devices, using time-series and non-time-series methods. All the results are based on data from real IoT mobile devices.

Additional goals of the study have been the analysis of the connectivity behaviour of real IoT devices and the creation of a generic dataset structure to be used by all the algorithms.

7.1.2 Datasets

For this work we have employed a dataset composed of real IoT samples from a multinational operator. The data has been formatted to be used by both time-series and non-time-series algorithms.

The dataset is the result of 6214 devices with 30 days of historical data, with a highly heterogeneous activity among the devices. The devices were active a 58% of the time (in average).

7.1.3 Models

We have considered two groups of models:

- Based in cross-sectional data:
  - Logistic Regression
  - Bayesian Logistic Regression
  - Random Forest
  - GBM
- Based in time-series data:
  - Exponential Smoothing
  - HMM
  - ARIMA
  - ARIMAX

We have additionally explored the combination of results from various classifiers to assess a possible improvement in results.

7.1.4 Results/Conclusions

We have obtained a global accuracy over 90% for most of the methods. ARIMAX has provided the highest prediction accuracy, with a global mean accuracy over 93%. Other algorithms as Random Forest, ARIMA and Logistic Regression have provided very good results as well.

ARIMAX training time is extremely high, which makes it unsuitable for industrial applications despite its good prediction results.

We have observed that connectivity behavior has more structure and less noise than was initially predicted, and that prediction accuracy has a periodic nature over forecasting time. Prediction accuracy gets reduced with forecasting time, but more slowly than expected.

7.2 Paper 2: Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things

Authors	Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas and Jaime Lloret
Title	Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things
Journal	IEEE Access, vol. 5, pp. 18042-18050, 2017.
Impact Factor	3.244
Quartile	Q1
#Citations	32 (https://scholar.google.es/citations?user=3RSZbOYAAAAJ&hl=es)
Status	Published, September-2017
Link	https://doi.org/10.1109/ACCESS.2017.2747560

7.2.1 Objectives

One of the main objectives of this paper has been to apply deep learning methods to detect the type-of-service of a network flow (NTC). To do this, we have used only the headers of the network packets without using the IP or the payload data transported by the packets, since these data may have confidentiality or encryption problems.

Another objective was to apply a CNN network to the vector time-series associated with the headers of the packets, and to prove that the representation of these as images is appropriate.

Finally, we wanted to demonstrate that the detection results using a deep learning model can be equivalent or better than using other classic models: C4.5, Naive Bayes, SVM, KNN, MLP, Random Forest, ...

7.2.2 Datasets

The main source of data for this work has been real network packets traffic (pcap) from a national internet service provider.

The dataset was formed by 266160 network flows with 20 packets each. The distribution of frequencies by type-of-service was very unbalanced, with a large number of different values (detection of 108 different types of service)

7.2.3 Models

We have studied the following deep learning models: RNN only, CNN only and different combinations of CNN and RNN.

We have produced several performance metrics: Accuracy, Precision, Recall and F1. The F1 score has been considered the most important due to the unbalanced nature of the dataset.

The impact of the different features and the number of packets per flow has been extensively analyzed.

7.2.4 Results/Conclusions

It has been proved that a combination of CNN and RNN models is applicable to NTC with excellent results, as well as that a temporal series of vectors can be assimilated to an image.

A model formed by a combination of CNN and RNN gives the best prediction results, although an exclusive RNN model also gives excellent results.

Although we have used 20 packets per flow, good prediction results can be obtained using very few packages per flow (5-15). These results are achieved despite having a large number of label values (108) with a very unbalanced distribution.

Considering aggregated results (weighted average over all labels), the best model attains an accuracy of 0.9632, an F1 score of 0.9574, a precision of 0.9543 and a recall of 0.9632. Similarly, considering One-vs.-Rest results (for each label separately) for all labels with a frequency higher than 1% we achieve accuracy always higher than 98%, and many cases higher than 99%, and an F1 score higher than 0.96. For labels with a frequency lower than 1% the results are worse with more variability.

7.3 Paper 3: Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT

Authors	Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas and Jaime Lloret.
Journal	Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT
Journal	Sensors 2017, 17(9), 1967
Impact Factor	2.677
Quartile	Q1
#Citations	12 (https://scholar.google.es/citations?user=3RSZbOYAAAAJ&hl=es)
Status	Published, August-2017
Link	https://doi.org/10.3390/s17091967

7.3.1 Objectives

This paper has two main objectives: (1) to apply a C-VAE to the intrusion detection problem in data networks, and (2) to be able to synthesize predictors (features) with the same probability distribution as the originals, making it possible to apply this capability to recover damaged datasets or with missing features.

In the first objective the intention is to obtain better prediction metrics than with classic algorithms (Random Forest, SVM, Logistic Regression...).

In the second objective the purpose is to achieve an accuracy of the synthetic features as high as possible.

7.3.2 Datasets

For this work we have used the NSL-KDD [67] dataset. This is a classic Intrusion Detection dataset. The dataset has 32 continuous and 3 categorical features, with an intrusion label of 5 values (Normal, DoS, Probe, R2L and U2R). This is a quite unbalanced dataset which is important to be representative of the datasets found with intrusion detection problems.

The dataset needed to be transformed before applying the detection algorithm. All categorical variables were one-hot encoded and the continuous were scaled in the range [0,1].

The dataset was split between training and test subsets, with 125973 and 22544 samples respectively.

7.3.3 Models

In the paper we have analyzed two main models: VAE and C-VAE.

In this case, we have also produced several performance metrics: Accuracy, Precision, Recall and F1. Again, the F1 score has been considered the most important due to unbalanced nature of the dataset.

All results provided in the paper are obtained using exclusively the NSL-KDD test subset.

7.3.4 Results/Conclusions

The results from the paper allow us to conclude that a C-VAE model is applicable to the prediction of intrusions in data networks with better results than other classic models (SVM, Random Forest, ...)

Similarly, a C-VAE model can be used successfully to reconstruct features in accordance with a probability distribution similar to the original features and conditioned to the detected intrusion.

The results can be divided in two groups: (1) classification prediction results and (2) accuracy of synthetic features results.

Considering classification results, our proposed model obtains an F1 score of 0.79 and an accuracy and recall of 0.80 which are the highest among all the algorithms studied. These metrics may not be seen as very high, but it is important to realize that these are aggregated results for a very unbalanced predicted label. When each label is considered separately, for one-vs.-rest results, we obtain a F1 score greater than 0.82 for the most frequent labels and accuracy over 0.91 for four of the five label values. The behavior of lower frequency labels is noisy due to the nature of the training and test datasets (NSL-KDD). For one-vs.-rest the accuracy obtained is always greater than 0.83 regardless of the label.

Taking into account the results of synthetic (reconstructed) features, we have mainly considered the recovery of the categorical features, where the achievable accuracy is related to the number of values of the feature. For the features *protocol* and *flag* with 3 and 11 values, we have obtained an accuracy of 99% and 92% respectively, while for the feature *service* with 70 values the accuracy is 71% (quite good considering the large number of values to recover). The remaining metrics (F1, precision, and recall) have very similar values to the accuracy score.

7.4 Paper 4: Deep learning model for multimedia Quality of Experience prediction based on network flow packets

Authors	Manuel Lopez-Martin, Belen Carro, Jaime Lloret, Santiago Egea, Antonio Sanchez-Esguevillas
Title	Deep learning model for multimedia Quality of Experience prediction based on network flow packets
Journal	IEEE Communications Magazine, vol. 56, no. 9, pp. 110-117, Sept. 2018 Feature Topic: Enabling Technologies for Smart Internet of Things
Impact Factor	10.435
Quartile	Q1
#Citations	1 (https://scholar.google.es/citations?user=3RSZbOYAAAAJ&hl=es)
Status	Published, September 2018
Link	https://doi.org/10.1109/MCOM.2018.1701156

7.4.1 Objectives

The objective of this work was to build a video QoE detector/predictor using information extracted from network packets. The intention was also to explore the use of deep learning models to achieve the goal.

Other important objective was to have a QoE classifier which could be integrated into a network management system to monitor network quality (as observed by the end-user), allowing at the same time an efficient network reconfiguration and control (in our case an SDN network). Therefore, the QoE classifier needed to identify the QoE score of the video transmitted at the current time-interval, but also be able to anticipate (predict) the quality score for the next time-interval. Since, this prediction can be crucial to anticipate actions on network resources.

The resulting QoE classifier must implement a binary classification (good or bad quality) for seven usual classes of video anomalies (blur, ghost, columns, chrominance, blockness, color bleeding and black pixel) that can happen when watching the videos.

7.4.2 Datasets

For this work we have created our own dataset of video QoE scores provided by individuals watching video streams under different network conditions. To obtain this dataset we created a specific experimental setup.

The topology of the experimental setup included three components: (1) A video transmission server, which allowed us to vary the characteristics of the video. (2) The clients, where the

video streams are visualized by the end user to label them with QoE errors. And, (3) a packet analyser (Wireshark) that extracts the network parameters on the end user's side. This configuration allows us to vary several network and video features (jitter, delay, bandwidth, packet loss, bitrate ...) and test their impact on network packets and their associated visual effects. We used several network protocols (HTTP, RTP and UDP) to increase the variety of video transmissions.

The resulting dataset was subsequently processed to obtain 40 features of aggregated information extracted from the network packets. These features (elementary flows) form a time-series of vectors that were additionally packaged in groups of three to finally achieve a dataset of 2078 elementary flows. This final dataset is the one used to train the classifier.

7.4.3 Models

We have studied the following deep learning models: RNN only, CNN only and different combinations of CNN and RNN.

We have also studied the addition of a final layer based on a Gaussian process classifier. This addition increases the detection/prediction performance but not significantly.

We have produced several performance metrics: Accuracy, Precision, Recall, AUC and F1. The F1 and AUC scores have been considered the most important due to the unbalanced nature of the dataset.

The impact of the time-series length (number of elementary flows) has been also analyzed. Interestingly, the best performance metrics are obtained for three samples per elementary flow. The performance does not increase when increasing the number of samples per flow beyond three, implying that prediction is conditioned on the amount of previous information, but information too distant in time is not only less useful but indeed a problem. Similarly, reducing the number of samples below three also decreases the prediction performance.

7.4.4 Results/Conclusions

The classifier achieves excellent prediction results for the non-extremely unbalanced labels. These results are more significant considering the small number of training samples available and the noisy nature of the video anomalies detection (subjective component).

Considering each label separately we obtained for the less unbalanced labels an F1 between 0.6429 to 0.8416, an accuracy between 0.8397 to 0.8974 and an AUC between 0.8689 to 0.9520. For the extremely unbalanced labels the results are quite poor.

For the aggregated results, considering all labels, the proposed model formed by a CNN+LSTM+Gaussian-Process achieves the best F1 score (0.6987).

It is interesting to note that the performance metrics get worse at the next time-step vs. the current time-step, as expected, but the reduction in performance is quite small, which is good news as it confirms the implicit initial hypothesis that the prediction of QoE was possible.

7.5 Paper 5: Variational data generative model for intrusion detection

Authors	Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas
Title	Variational data generative model for intrusion detection
Journal	Knowledge and Information Systems
Impact Factor	2.247
Quartile	Q2
#Citations	0 (https://scholar.google.es/citations?user=3RSZbOYAAAAJ&hl=es)
Status	Published: 13-December-2018
Link	https://doi.org/10.1007/s10115-018-1306-7

7.5.1 Objectives

In order to train an intrusion detection classifier is very important to have access to representative and balanced training data. This is usually a difficult task since intrusion detection samples of network traffic are strongly biased to normal traffic, being difficult to access traffic associated with intrusion events. Considering these difficulties, it is important to have a way to produce traffic samples associated to intrusion events which are rare compared with the main/normal traffic.

There are several classic techniques to over-sample the minority classes in order to have a more balanced dataset (e.g. SMOTE, ADASYN…). These techniques create new data points based in the proximity to existing points of the same class. They are based on topological proximity and do not consider the probability distribution of features for the different classes.

In this work is presented a new method to create synthetic data based on their probability distribution conditioned on the class to which they belong. This new method consists of a generative model based on a customized Variational Autoencoder (VAE). The VAE architecture has been modified to create a novel model based on a conditional VAE which integrates the class label as a new input to the VAE's decoder architecture.

The advantage of using a conditional generative model to generate new data is based on the capacity of this model to create data using noise as input, therefore we do not create synthetic samples based on proximity to existing samples (a noisy and error prone task) but on following a probability distribution for the minority class which is learned as part of the training of the resulting model.

Besides the above-mentioned advantage as an easier and more principled way to generate synthetic data, we show that the novel architecture, based on the conditional VAE, produces synthetic data that provides better results when this data is used as additional data to train a

classifier for intrusion detection. The work carries out an extensive comparison of the synthetic data produced by the new method with data produced by classis over-sampling techniques showing the better performance (when used as synthetic training data) of the new proposed method.

7.5.2 Datasets

For this work we have used the NSL-KDD [67] dataset. This is a classic Intrusion Detection dataset. The dataset has 32 continuous and 3 categorical features, with an intrusion label of 5 values (Normal, DoS, Probe, R2L and U2R). This is a quite unbalanced dataset.

We have performed an additional data transformation: scaling all NSL-KDD continuous features to the range [0,1] and one-hot encoding all categorical features. This provides a final dataset with 116 features: 32 continuous and 84 with values in {0,1} associated to the three one-hot encoded categorical features.

The three categorical features: protocol, flag and service have respectively 3, 11 and 70 distinct values. The accuracy obtained when synthesizing these discrete features (having as reference the original ones) depends heavily on the cardinality of the feature.

We provide all results using the full original training dataset of 125973 samples and the full original test dataset of 22544 samples.

7.5.3 Models

The novel proposed architecture consists of a VAE which tries to recover an output identical to the inputs (the inputs being the network traffic features used to detect the intrusion class) but introducing a variation to the normal VAE consisting of the inclusion of an additional input to the decoder. This additional input is the one-hot encoded class label. The addition of this input is critical to improve the model in two directions: making easier the data generation process (which is now conditioned on the class label) and producing better synthetic data which is more closely related to the original one in terms of probability distribution conditioned on the class label.

To arrive to the proposed model, we have analyzed different VAE architecture variants, providing an extensive study on the alternatives.

Besides the proposal of a new architecture based on a conditional VAE we have used several machine learning techniques to demonstrate that the generated synthetic data can be used to improve the intrusion detection results of several classifiers (Random Forest, Logistic Regression, SVM and MLP).

The work also shows that the synthetic data has a similar probability distribution for the features depending on their intrusion classes. We have developed several approaches to verify the similarity: (1) extended histograms of the original and synthesized features; and (2)

classification results obtained from the application of original and synthesized data to several classification algorithms.

To conclude the comparative of results we have checked the results obtained with the new model compared to some well-known over-sampling methods: SMORE, SMOTE-Borderline, SMOTE-ENN, SMOTE-Tomek, ADASYN.

7.5.4 Results/Conclusions

This work is the response to several challenges:

- Generate synthetic data for an intrusion detection dataset, with many and heterogeneous features both continuous and discrete and with a highly imbalanced distribution of intrusion labels.

  This has been achieved by using a new generative model based on a conditional VAE.

- To show that the synthetic generated data have similar probabilistic structure to the original data. Verifying this similarity is a hard problem since it involves comparing the probability distributions of multivariate vectors (116 features) with non-Gaussian marginals (discrete and continuous features) and complex joint probability distributions. The challenge is twofold: obtain the joint probability distributions and compare them.

  To handle these problems, we have proposed several methods based on extended histograms and the comparison of classification results under different scenarios of training with original and synthetic data

- To show that the synthetic data generated with the new architecture produces better results than synthetic data generated by state-of-the-art (SOTA) over-sampling methods.

  This has been shown when comparing accuracy and F1 classification results when using training data generated by several over-sampling algorithms including the proposed one. We demonstrate that both accuracy and F1 are improved when the new architecture is used.

8. TOOLS

To carry out the different experiments of this thesis, the hardware/software resources used have been a PC (i7-4720-HQ, 16 GB RAM) and several open-source software packages: (a) the machine learning package scikit-learn (python), (b) the deep learning software platform Tensorflow and Keras (python), and (c) the language R to perform some statistical analysis and to implement all the time-series models (ARIMA, ARIMAX, Hidden Markov Model (HMM), and Exponential Smoothing) with functions provided mainly by the forecast R package.

9. GENERAL CONCLUSIONS AND SUMMARY OF CONTRIBUTIONS

The application of machine learning techniques to data networking and telecommunications is providing fruitful results; however, there are still many application opportunities, which will arise in the future as soon as the new methods developed, mainly in the very active space of deep learning research, move beyond their initial research scope: image processing, natural language processing, automatic translation, voice recognition… to the field of networking and telecommunication. This thesis tries to provide a contribution in that direction to shorten the gap between the advances of research in machine learning and its application to the Telco area.

In the following paragraphs, a summary of the contributions of the different papers is provided:

Paper1: "Review of methods to predict connectivity of IoT wireless devices"

- ***Contribution_1***: Study of the activity behaviour of IoT devices in a real environment.
- ***Contribution_2***: Thorough comparison of application of "time-series" and "cross-sectional" models to IoT future activity prediction.
- ***Contribution_3***: The lessons learned, and results obtained for the best algorithms can be applicable to a real environment with a big number of IoT devices. The results present a very high forecasting accuracy.
- ***Contribution_4***: As far as we know, it is the first reported application of a Random Forest algorithm to a time-series prediction scenario.
- ***Contribution_5***: It is shown that one week of historical data is enough to provide good forecasts and the method with best absolute accuracy performance is ARIMAX, but Logistic Regression or Random Forest could be better operational models due to the excessive training time of ARIMAX.

Paper 2: "Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things"

- ***Contribution_1***: First application, as far as we know, of a CNN+RNN model to the traffic classification problem (NTC).
- ***Contribution_2***: The proposed method provides better detection results than alternative algorithms without requiring any feature engineering, which is usual when applying other models**.**
- ***Contribution_3***: The resulting model is applicable to a very unbalanced dataset and using only a few packages per flow as predictors.

Paper 3: "Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT"

- ***Contribution_1***: First application, as far as we know, of a C-VAE model to an intrusion detection problem.
- ***Contribution_2***: It proposes a new approach for attacks classification and synthetic features generation based in a generative model.
- ***Contribution_3***: It obtains better prediction results than classic machine learning models.
- ***Contribution_4***: It provides the capacity to generate synthetic features associated to particular intrusion events.

*Paper 4: "*Deep learning model for multimedia Quality of Experience prediction based on network flow packets*"*

- ***Contribution_1:*** First application, as far as we know, of a CNN+RNN model to video QoE prediction.
- ***Contribution_2:*** Prediction based on network packet information.
- ***Contribution_3:*** Network flows treated as pseudo-images that allow applying a CNN.
- ***Contribution_4:*** Excellent prediction performance for not extremely unbalanced labels with a small dataset.

Paper 5: "Variational data generative model for intrusion detection"

- ***Contribution_1*:** First application, as far as we know, of a VAE as a generative model for intrusion detection
- ***Contribution_2:*** To provide means to demonstrate that the data generated is similar to the original data, and, at the same time, have enough variability to be effective in improving the detection performance of several classifiers when used together with the original data.
- ***Contribution_3:*** To provide the ability to synthesize the new samples from the intrusion labels to which the synthetic data should belong, with the advantage of not relying on specific samples associated with the labels.
- ***Contribution_4:*** To propose a new over-sampling algorithm whose synthetic data improves the classification results of classic SOTA over-sampling techniques.

The contributions from the different papers can be integrated into a more general list of global contributions of the thesis:

- The new deep learning models are based on the assembly of known layers in a lego-like form, which offer endless possibilities to explore new architectures. This creates opportunities for the application of these models to networking, as evidenced by the deep learning models proposed in this thesis [2][3][4][5].

- Many well-known ML models that were not originally thought to be applied in networking can be successfully modified to be used in this important area, by performing data transformations or adaptations of the original model. This thesis demonstrates several of these adaptations/transformations [1][2][3][4][5].

10. FUTURE LINES OF RESEARCH

Considering the lessons learned from this work and the possibilities anticipated by the new models that are actively being developed in the scientific community, we consider the following lines of research could be feasible and provide promising results:

- Explore new models of deep learning: Generative Adversarial Networks (GAN) [135], ladder VAE [138], structured VAE [139] and Siamese networks [140]. There is a continuous flow of new models in the very active area of ML research, and there will be great opportunities in applying the new models to different functional areas in networking, as evidenced by a recent survey on deep learning for mobile and wireless networking [29] which shows the large number of works on this area and the future opportunities. In this line, other studies [141][142] clearly indicate the growing importance of the application of machine learning in general and, specifically, deep learning models in other fields of prediction in data networks.

- Explore the end-to-end training of a deep learning network with a Gaussian process for regression problems [143]. Gaussian processes are especially suitable for regression problems and the possibility of having a model that combines a neural network architecture with a Gaussian process as the final layer of the network is especially interesting, mainly when the entire model can be trained end-to-end with an appropriate loss function related to the optimization of the Gaussian process [143].

- Explore models that require very little data for training: one-shot learning, zero-shot learning [144][145]. These new models will surely have an important role in networking problems where data is not always available, at least of the type and nature required e.g. in intrusion detection there is a large amount of normal data, but very few samples that correspond to known intrusions.

- Explore the application of reinforcement learning models to problems of traffic control and cybersecurity in data networks [146][147][148]. A reinforcement learning algorithm can learn by receiving sparse indications (rewards) of the good or bad actions taken so far. These algorithms are the focus of an important research interest and will surely be important in future ML applications for networking [29], particularly in network and resources management, routing and control problems, and intrusion detection and cybersecurity applications. An interesting extension of the current research would be to apply deep reinforcement learning algorithms to intrusion detection.

- Explore the use of aggregated data from different information sources of heterogeneous nature which relates to the study and analysis of security logs. Security logs are critical for all security management aspects. Security logs are the main entry point of information for security threats, having their own ecosystem of functions [58]: acquisition, filtering, normalization, collection management, storage, analysis and long-term storage. Sometimes, the logs ecosystem is highly optimized and automated, but in

other cases, or for some functions they require an important manual/supervision labour. This opens the opportunity for the application of ML algorithms for statistical analysis and logs data mining [58].

- Explore the necessary measures that must be implemented to ensure that an intrusion detection algorithm is not attacked by people/groups who wish to change its intended behaviour. Many machine learning algorithms (e.g. deep learning) are “black boxes”, whose decisions are difficult or impossible to interpret. This opens the door to attacks that are not based on modifying an algorithm that already works, but on changing very slightly the features used as predictors to produce a huge impact on the decision made by the algorithm. These attacks may go unnoticed by the user. This is an important area of research in other fields, as image processing [149] and also attracts interest in the area of safety-critical environments [150]. An area of related interest are the measures necessary to implement fair systems in terms of racial discrimination or other possible areas of discrimination [151]. In this case the emphasis is placed on the selection of the dataset used for training and/or on constraints in the optimization of the algorithm.

11. RESEARCH DISSEMINATION PLAN

In order to disseminate the results of the different research works carried out in this thesis, a poster session was held in the last Machine Learning Summer School - MLSS 2018-Madrid (http://mlss.ii.uam.es/mlss2018/index.html). The title of the poster was: Application of Machine Learning to prediction problems in data networking (http://mlss.ii.uam.es/mlss2018/posters.html). The poster is included in Section III after the manuscripts of all papers.

To address the important requirement of reproducibility of results, the code for papers without restrictions due to the property rights of the datasets, is available at: https://github.com/mlopezm/thesis-experiments2

12. LIST OF REFERENCES

[1] Lopez-Martin M, Sanchez-Esguevillas A. and Carro B., “Review of methods to predict connectivity of IoT wireless devices”, Ad Hoc & Sensor Wireless Networks, 38.1-4, p. 125-141. http://www.oldcitypublishing.com/journals/ahswn-home/ahswn-issue-contents/ahswn-volume-38-number-1-4-2017/ahswn-38-1-4-p-125-141/

[2] 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

[3] Lopez-Martin M et al., “Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT”. Sensors 17 (9), 2017. doi: https://doi.org/10.3390/s17091967

[4] Lopez-Martin M et al., “Deep learning model for multimedia Quality of Experience prediction based on network flow packets”, IEEE Communications Magazine, September 2018. doi: https://doi.org/10.1109/MCOM.2018.1701156

[5] Lopez-Martin M, Carro B. and Sanchez-Esguevillas A., “Variational data generative model for intrusion detection”, Knowledge and Information Systems, December-2018. doi: https://doi.org/10.1007/s10115-018-1306-7

[6] 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.

[7] Mahdavinejad M.S. et al., “Machine learning for Internet of Things data analysis: A survey”, Digital Communications and Networks, 2017.

[8] 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

[9] 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.

[10] Jiang C. et al., "Machine Learning Paradigms for Next-Generation Wireless Networks," in IEEE Wireless Communications, vol. 24, no. 2, pp. 98-105, April 2017

[11] Breiman L., “Random Forests. Machine Learning”, Volume 45, Issue 1, October 1 2001, Pages 5-32

[12] Friedman J.H., “Greedy Function Approximation: A Gradient Boosting Machine”. The Annals of Statistics, Vol. 29, No. 5 (Oct., 2001), pp. 1189-1232

[13] Cortes, C., Vapnik, V., "Support-vector networks". Machine Learning. 20 (3): 273–297. 1995.

[14] Hastie T., Tibshirani R. and Friedman J., “The Elements of Statistical Learning: Data mining, inference, and prediction”. Second Edition (2009), Springer Series in Statistics

[15] Haykin, S.,“Neural Networks: A Comprehensive Foundation”. 1998, Prentice Hall.

[16] Bishop C.M., “Pattern Recognition and Machine Learning” (Information Science and Statistics). 2006, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[17] Chapelle, O., Schölkopf, B. and Zien, A., “Semi-supervised learning”. Cambridge, 2006, Mass.MIT Press

[18] Sutton R.S and Barto A.G., “Reinforcement Learning”, Cambridge, 1998, Mass.MIT Press

[19] Goodfellow I., Bengio Y. and Courville A., “Deep Learning”, Cambridge, 2016, Mass.MIT Press

[20] Behnke S., “Hierarchical Neural Networks for Image Interpretation” (Lecture Notes in Computer Science), vol. 2766. Berlin, Germany: Springer-Verlag, 2003.

[21] Lipton Z. C., Berkowitz J. and Elkan C., “A critical review of recurrent neural networks for sequence learning.’’. 2015, arXiv:1506.00019 [cs.LG]
[22] Greff K. et al., ‘‘LSTM: A search space odyssey.’’. 2015, arXiv:1503.04069 [cs.NE]
[23] Kingma, D.P. and Welling, M. “Auto-Encoding Variational Bayes”., 2014, arXiv:1312.6114v10.
[24] Kingma, D.P. et al., “Semi-supervised learning with deep generative models”. In Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS’14), Montreal, QC, Canada, 8–13 December 2014, MIT Press: Cambridge, MA, USA, 2014; pp. 3581–3589.
[25] Sohn K., Yan X. and Lee H., “Learning structured output representation using deep conditional generative models”. In Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS’15), Montreal, QC, Canada, 7–12 December 2015, MIT Press: Cambridge, MA, USA, 2015; pp. 3483–3491.
[26] Rabiner L. R., “A tutorial on Hidden Markov Models and selected applications in speech recognition”. Proceedings of the IEEE, Vol. 77, No. 2. (06 February 1989), Pages. 257-286
[27] Holt C.C., “Forecasting Trends and Seasonal by Exponentially Weighted Averages”. International Journal of Forecasting, Volume 20, Issue 1 (January–March 2004), Pages 5–10.
[28] 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
[29] Zhang C., Patras P. and Haddadi H., "Deep Learning in Mobile and Wireless Networking: A Survey," in IEEE Communications Surveys & Tutorials. 2019. doi: 10.1109/COMST.2019.2904897
[30] Kim K. and Aminanto M. E., "Deep learning in intrusion detection perspective: Overview and further challenges," 2017 International Workshop on Big Data and Information Security (IWBIS), Jakarta, 2017, pp. 5-10. doi: 10.1109/IWBIS.2017.8275095
[31] Papernot N. and McDaniel P., “Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning”. arXiv:1803.04765v1 [cs.LG] 13 Mar 2018.
[32] Anderson J. W. et al., "Synthetic data generation for the internet of things," 2014 IEEE International Conference on Big Data (Big Data), Washington, DC, 2014, pp. 171-176.
[33] OpenAI Blog, (June 2016), web: “https://blog.openai.com/generative-models/ “. html, as of Nov 29, 2018
[34] Zarpelo, B.B. et al., “A survey of intrusion detection in Internet of Things”. J. Netw. Comput. Appl. 2017, 84, 25–37.
[35] Papadopouli M., “Evaluation of short term traffic forecasting algorithms in wireless networks”, 2006 2nd Conference on Next Generation Internet Design and Engineering, NGI, Valencia (April 2006), Pages 102-109
[36] Yantai S. et al, “Wireless traffic modeling and prediction using seasonal ARIMA models”, IEEE International Conference on Communications, 2003 (ICC 2003), Anchorage (May 2003), Vol. 3, Pages 1675-1679
[37] Stolojescu-Crisan C., “Data mining based wireless network traffic forecasting”, 2012 10th International Symposium on Electronics and Telecommunications (ISETC), Timisoara (Nov 2012), Pages 115-118.
[38] Sivanathan A et al., ‘‘Characterizing and classifying IoT traffic in smart cities and campuses,’’ in Proc. IEEE INFOCOM Workshop SmartCity, Smart Cities Urban Comput., Atlanta, GA, USA, May 2017, pp. 1–6.
[39] Wang J., Pan J., and Esposito F., “Elastic urban video surveillance system using edge computing,”, Proceedings of the Workshop on Smart Internet of Things (SmartIoT '17). ACM, New York, NY, USA, 2017, Article 7

[40] Bilal K. and Erbad A., "Edge computing for interactive media and video streaming," 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, Spain, May 8-11, 2017, pp. 68-73
[41] Ananthanarayanan G. et al., "Real-Time Video Analytics: The Killer App for Edge Computing," Computer, vol. 50, no. 10, 2017, pp. 58-67.
[42] Chen Y., Wu K. and Zhang Q., "From QoS to QoE: A Tutorial on Video Quality Assessment," IEEE Communications Surveys & Tutorials, vol. 17, no. 2, 2015, pp. 1126-1165.
[43] Huang F. et al., "Reliability Evaluation of Wireless Sensor Networks Using Logistic Regression", 2010 International Conference on Communications and Mobile Computing, Shenzhen (April 2010), Vol. 3, Pages 334-338
[44] Feng H et al., "SVM-Based Models for Predicting WLAN Traffic", 2006 IEEE International Conference on Communications (ICC 2006), Istanbul (June 2006), Vol. 2, Pages 597-602
[45] Meidan Y. et al., ''ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis,'' in Proc. ACM Symp. Appl. Comput. (SAC), New York, NY, USA, 2017, pp. 506–509.
[46] Althunibat S. et al., ''Countering intelligent-dependent malicious nodes in target detection wireless sensor networks,'' IEEE Sensors J., vol. 16, no. 23, pp. 8627–8639, Dec. 2016.
[47] Grajzer M. et al., ''A multi- classification approach for the detection and identification of eHealth applications,'' in Proc. 21st Int. Conf. Comput. Commun. Netw. (ICCCN), Munich, Germany, Jul./Aug. 2012, pp. 1–6.
[48] Bhuyan, M.H., Bhattacharyya, D.K. and Kalita, J.K. "Network Anomaly Detection: Methods, Systems and Tools". In IEEE Communications Surveys & Tutorials; IEEE: Piscataway, NJ, USA, 2014; Volume 16, pp. 303–336.
[49] Vacca J. R., "Computer and Information Security Handbook (Second Edition)", Morgan Kaufmann, 2013, Pages 81-95. https://doi.org/10.1016/B978-0-12-394397-2.00005-2
[50] Kruegel C. et al., "Intrusion detection and correlation - Challenges and Solutions", Part of the Advances in Information Security book series (ADIS, volume 14). 2005. DOI: 10.1007/B101493
[51] Kumar D.A. and Venugopalan S.R., "Intrusion detection systems: A review", International Journal of Advanced Research in Computer Science, vol 8, no. 8, 2017. DOI: http://dx.doi.org/10.26483/ijarcs.v8i8.4703
[52] Marty R., The Security Data Lake, O'Reilly Media, Inc., 2015, https://learning.oreilly.com/library/view/the-securitydata/9781491927748/
[53] Marty R., "AI & ML IN CYBERSECURITY, Why Algorithms Are Dangerous", BlackHat, USA, August 2018. https://i.blackhat.com/us-18/Thu-August-9/us-18-Marty-AI-and-ML-in-Cybersecurity.pdf
[54] Gao D., Reiter M.K. and Song D., "Behavioral distance for intrusion detection". In Proceedings of the 8th international conference on Recent Advances in Intrusion Detection (RAID'05), Alfonso Valdes and Diego Zamboni (Eds.). Springer-Verlag, Berlin, Heidelberg, 63-81. 2005. DOI=http://dx.doi.org/10.1007/11663812_4
[55] Song Y., Keromytis A. D. and Stolfo S., "Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic", Network and Distributed System Security Symposium 2009: February 8-11, San Diego, California. 2009. https://doi.org/10.7916/D8891G6G
[56] Marty R., "Applied Security Visualization (1 ed.)". Addison-Wesley Professional. 2008.
[57] Sharafaldin I. et al., "An Evaluation Framework For Network Security Visualizations". Computers & Security, 2019. https://doi.org/10.1016/j.cose.2019.03.005 .
[58] Chuvakin A. A. et al., "Logging and Log Management". Syngress, Elsevier. Book. 2013

[59] Aggarwal, C.C. Outlier Analysis; Springer: New York, NY, USA, 2013; pp. 10–18, ISBN 978-1-4614-639-5.
[60] Hodo E., Bellekens X. and Hamilton A., "Threat analysis of IoT networks using artificial neural network intrusion detection system". In Proceedings of the 2016 International Symposium on Networks, Computers and Communications (ISNCC), Yasmine Hammamet, Tunisia, 11–13 May 2016; pp. 1–6.
[61] Kang M.J. and Kang J.W., "Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security". PLoS ONE 2016, 11, e0155781.
[62] Thing, V.L.L., "IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach". In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 19–22 March 2017; pp. 1–6.
[63] Ma T. et al., "A Hybrid Spectral Clustering and Deep Neural Network Ensemble Algorithm for Intrusion Detection in Sensor Networks". Sensors 2016, 16, 1701.
[64] An J. and Cho S., "Variational Autoencoder based Anomaly Detection using Reconstruction Probability", Seoul National University, SNU Data Mining Center, 2015-2 Special Lecture on IE, Seoul, Korea, 2015.
[65] Suh, S., Chae, D.H., Kang, H.G and Choi, S. "Echo-state conditional Variational Autoencoder for anomaly detection". In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016, pp. 1015–1022, doi:10.1109/IJCNN.2016.7727309.
[66] Sölch, M. "Detecting Anomalies in Robot Time Series Data Using Stochastic Recurrent Networks". Master's Thesis, Department of Mathematics, Technische Universitat Munchen, Munich, Germany, 2015.
[67] Tavallaee, M. et al. "A detailed analysis of the KDD CUP 99 data set". In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
[68] Ingre, B. and Yadav, A. "Performance analysis of NSL-KDD dataset using ANN". In Proceedings of the 2015 International Conference on Signal Processing and Communication Engineering Systems, Guntur, India, 2–3 January 2015; pp. 92–96, doi:10.1109/SPACES.2015.7058223.
[69] Ibrahim, L.M, Basheer, D.T. and Mahmod, M.S. "A comparison study for intrusion database (KDD99, NSL-KDD) based on self-organization map (SOM) artificial neural network". In Journal of Engineering Science and Technology; School of Engineering, Taylor's University: Selangor, Malaysia, 2013; Volume 8, pp. 107–119.
[70] Wahb, Y, ElSalamouny E. and ElTaweel, G. "Improving the Performance of Multi-class Intrusion Detection Systems using Feature Reduction". arXiv 2015, arXiv:1507.06692.
[71] Chen, C.-M et al.,"Anomaly Network Intrusion Detection Using Hidden Markov Model". Int. J. Innov. Comput. Inform. Control 2016, 12, 569–580.
[72] Alom, M.Z., Bontupalli, V. and Taha, T.M. "Intrusion detection using deep belief networks". In Proceedings of the 2015 National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 June 2015, pp. 339–344.
[73] Xu, J. and Shelton, C.R. "Intrusion Detection using Continuous Time Bayesian Networks". J. Artif. Intell. Res. 2010, 39, 745–77.
[74] Hodo, E. et al., "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey". arXiv 2017, arXiv:1701.02145.
[75] Abolhasanzadeh B., "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features," in 2015 7th Conference on Information and Knowledge Technology (IKT), 2015, pp. 1–5.
[76] Fiore U. et al., "Network anomaly detection with the restricted Boltzmann machine," Neurocomputing, vol. 122, pp. 13–23, Dec. 2013.

[77] Gao N. et al., “An Intrusion Detection Model Based on Deep Belief Networks,” in 2014 Second International Conference on Advanced Cloud and Big Data, 2014, pp. 247–252.
[78] Panda M., et al., “Discriminative Multinomial Naïve Bayes for Network Intrusion Detection”, Proceedings of 6th Intl. conf. on information assurance and security (IAS-2010), Aug. 2010, USA, 5-10, IEEE Press, 2010
[79] Dhanabal , L and Shantharajah, S.P. “A Study on NSL- KDD Dataset for Intrusion Detection System Based on Classification Algorithms”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 6 , June 2015, 2015
[80] Kamel S.O.M. et al, “AdaBoost Ensemble Learning Technique for Optimal Feature Subset Selection”, International Journal of Computer Networks and Communications Security VOL. 4, NO. 1, January 2016, 1–11, 2016
[81] Patil D.R and. Pattewar T.M, “A Comparative Performance Evaluation of Machine Learning-Based NIDS on Benchmark Datasets”, International Journal of Research in Advent Technology, Vol.2, No.2, April 2014 E-ISSN: 2321-9637, 2014
[82] Gao H., Wang X. and Yang H., "LS-SVM Based Intrusion Detection using Kernel Space Approximation and Kernel-Target Alignment," 2006 6th World Congress on Intelligent Control and Automation, Dalian, 2006, pp. 4214-4218.
[83] Movahedi P. et al. “Fast regularized least squares and k-means clustering method for intrusion detection systems”. 2015. Proceedings of the International Conference on Pattern Recognition Applications and Methods.
[84] Joshi M.R. and Hadi T.H., “A Review of Network Traffic Analysis and Prediction Techniques”, 2015, arXiv:1507.05722 [cs.NI]
[85] Priyamvada and Wadhvani R. "Review on various models for time series forecasting," 2017 International Conference on Inventive Computing and Informatics (ICICI), Coimbatore, 2017, pp. 405-410.
[86] Mahalakshmi G., Sridevi S. and Rajaram S., "A survey on forecasting of time series data," 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16), Kovilpatti, 2016, pp. 1-8.
[87] Stolojescu-Crisan C., “Data mining based wireless network traffic forecasting”, 2012 10th International Symposium on Electronics and Telecommunications (ISETC), Timisoara (Nov 2012), Pages 115-118.
[88] Papadopouli M., “Evaluation of short term traffic forecasting algorithms in wireless networks”, 2006 2nd Conference on Next Generation Internet Design and Engineering, NGI, Valencia (April 2006), Pages 102-109
[89] Shu Y. et al, “Wireless traffic modeling and prediction using seasonal ARIMA models”, IEEE International Conference on Communications, 2003 (ICC 2003), Anchorage (May 2003), Vol. 3, Pages 1675-1679
[90] Huang C.W., Chiang C.T. and Li Q., “A Study of Deep Learning Networks on Mobile Traffic Forecasting,”, IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2017.
[91] Wang J. et al., "Spatiotemporal modelling and prediction in cellular networks: A big data enabled deep learning approach," IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, Atlanta, GA, 2017, pp. 1-9.
[92] Vinayakumar R., Soman K. P. and Poornachandran P., "Applying deep learning approaches for network traffic prediction," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, 2017, pp. 2353-2358.
[93] Feng H. et al, “SVM-Based Models for Predicting WLAN Traffic”, 2006 IEEE International Conference on Communications (ICC 2006), Istanbul (June 2006), Vol. 2, Pages 597-602
[94] Z. Chen, J. Wen, and Y. Geng, “Predicting Future Traffic Using Hidden Markov Models,” Proc. IEEE 24th Int’l. Conf. Network Protocols (ICNP) 2016, pp. 1–6.

[95] Nguyen T. and Armitage G., “A survey of techniques for internet traffic classification using machine learning,” IEEE Commun. Surv. Tutorials, vol. 10, no. 4, pp. 56–76, 2008.
[96] Zhang J. et al., “Robust Network Traffic Classification,” IEEE/ACM Trans. Netw., vol. 23, no. 4, pp. 1257–1270, 2015.
[97] Auld T., Moore A. W. and Gull S. F., ‘‘Bayesian neural networks for Internet traffic classification,’’ IEEE Trans. Neural Netw., vol. 18, no. 1, pp. 223–239, Jan. 2007.
[98] Xie X. et al., ‘‘Network traffic classification based on error-correcting output codes and NN ensemble,’’ in Proc. 6th Int. Conf. Fuzzy Syst. Knowl. Discovery, Tianjin, China, Aug. 2009, pp. 475–479.
[99] Chen Z. et al., ‘‘Improving neural network classification using further division of recognition space,’’ Int. J. Innov., Comput., Inf. Control, vol. 5, no. 2, pp. 301–310, 2009.
[100] Zhou W. et al., ‘‘Internet traffic classi- fication using feed-forward neural network,’’ in Proc. Int. Conf. Comput. Problem-Solving (ICCP), Chengdu, China, Oct. 2011, pp. 641–646.
[101] Mathewos B., Carvalho M. and Ham F., ‘‘Network traffic classification using a parallel neural network classifier architecture,’’ in Proc. 7th Annu. Workshop Cyber Secur. Inf. Intell. Res. (CSIIRW), New York, NY, USA, 2011, p. 33.
[102] Moore A. W. and Papagiannaki D., “Toward the accurate identification of network applications,” in Proc. 6th Passive Active Meas. Workshop (PAM), Mar. 2005, vol. 3431, pp. 41–54.
[103] Zhou W. et al., "Internet traffic classification using feed-forward neural network," 2011 International Conference on Computational Problem-Solving (ICCP), Chengdu, 2011, pp. 641-646.
[104] Moore A., Zuev D. and Crogan L., "Discriminators for use in flow-based classification", Technical Report RR-05-13. Department of Computer Science Queen’s Mary University, 2005.
[105] Kim, H. et al., “Internet traffic classification demystified: myths, caveats, and the best practices”, In Proceedings of the 2008 ACM CoNEXT Conference (CoNEXT '08). ACM, New York, NY, USA, pp. 11:1–11:12, 2008.
[106] Shafiq M. et al., "Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms," 2016 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 2016, pp. 2451-2455.
[107] Wang C., Xu T. and Qin X., "Network Traffic Classification with Improved Random Forest," 2015 11th International Conference on Computational Intelligence and Security (CIS), Shenzhen, 2015, pp. 78-81. doi: 10.1109/CIS.2015.27
[108] Zhang J. et al. “An Effective Network Traffic Classification Method with Unknown Flow Detection”, IEEE Transaction on Network and ServiceManagement, Vol 12, Dec 2013
[109] Hao S. et al., "Network traffic classification based on improved DAG-SVM," 2015 International Conference on Communications, Management and Telecommunications (ComManTel), DaNang, 2015, pp.256-26
[110] Yamansavascilar B et al., "Application identification via network traffic classification," 2017 International Conference on Computing, Networking and Communications (ICNC), Santa Clara, CA, 2017, pp. 843-848.
[111] Canadian Institute for Cybersecurity, University of New Brunswick, Datasets, web: “http://www.unb.ca/research/iscx/dataset/ISCX-network-traffic-VPNdataset “, html as of Nov 29, 2018
[112] Yuan Z. and Wang C., "An improved network traffic classification algorithm based on Hadoop decision tree," 2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS), Chongqing, 2016, pp. 53-56. doi: 10.1109/ICOACS.2016.7563047

[113] Erman J. et al., “Offline/realtime traffic classification using semi-supervised learning,” Performance Evaluation, vol. 64, no. 9-12, pp. 1194–1213, Oct. 2007.
[114] Lloret J. et al., “A QoE management system to improve the IPTV network,” International Journal of Communication Systems, 24, 2011, pp. 118–138.
[115] Garcia M. et al., "A QoE Management System for Ubiquitous IPTV Devices," 2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, Sliema, 2009, pp. 147-152.
[116] Menkovski V., Exarchakos G. and Liotta A., "Machine Learning Approach for Quality of Experience Aware Networks," 2010 International Conference on Intelligent Networking and Collaborative Systems, Thessaloniki, 2010, pp. 461-466.
[117] Aroussi S., Mellouk A., "Survey on machine learning-based QoE-QoS correlation models," 2014 International Conference on Computing, Management and Telecommunications (ComManTel), Da Nang, 2014, pp. 200-204.
[118] Balachandran A. et al., “Developing a predictive model of quality of experience for internet video,” Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (SIGCOMM '13). ACM, New York, NY, USA, 2013, pp. 339-350.
[119] Bampis C. G., Bovik A. C., “Learning to Predict Streaming Video QoE: Distortions, Rebuffering and Memory,” arXiv:1703.00633 [cs.MM], 2017.
[120] Skopik F. et al., "Semi-synthetic data set generation for security software evaluation," 2014 Twelfth Annual International Conference on Privacy, Security and Trust, Toronto, ON, 2014, pp. 156-163. doi: 10.1109/PST.2014.6890935
[121] Patki N., Wedge R. and Veeramachaneni K., "The Synthetic Data Vault," 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, 2016, pp. 399-410. doi: 10.1109/DSAA.2016.49
[122] Boggs N., Zhao H., Du S., Stolfo S.J. “Synthetic Data Generation and Defense in Depth Measurement of Web Applications”. In: Stavrou A., Bos H., Portokalidis G. (eds) Research in Attacks, Intrusions and Defenses. RAID 2014. Lecture Notes in Computer Science, vol 8688. Springer. 2014. https://doi-org.ponton.uva.es/10.1007/978-3-319-11379-1_12
[123] Belenko V. et al., “Synthetic datasets generation for intrusion detection in VANET”. In Proceedings of the 11th International Conference on Security of Information and Networks (SIN '18). ACM, New York, NY, USA, Article 9, 6 pages. 2018. DOI: https://doi.org/10.1145/3264437.3264479
[124] Pendleton M. and Xu S., "A dataset generator for next generation system call host intrusion detection systems," MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, 2017, pp. 231-236. doi: 10.1109/MILCOM.2017.8170835
[125] Glasser J. and Lindauer B., "Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data," 2013 IEEE Security and Privacy Workshops, San Francisco, CA, 2013, pp. 98-104. doi: 10.1109/SPW.2013.37
[126] Fekade, B. et al., “Probabilistic Recovery of Incomplete Sensed Data in IoT”. IEEE Int. Things J. 2017, 1, doi:10.1109/JIOT.2017.2730360.
[127] Shang C. et al., “VIGAN: Missing View Imputation with Generative Adversarial Networks”, 2017, arXiv:1708.06724 [cs.CV].
[128] Ariga K., “Generative models for missing value completion”, 2016, University of Washington, Computer Science Department, CSE446 ML Final Project.
[129] Kingma D.P. and Welling M, “Auto-Encoding Variational Bayes”. arXiv:1312.6114v10 [stat.ML]. 2014.
[130] Goodfellow I.J. et al, “Generative Adversarial Networks”. arXiv:1406.2661v1 [stat.ML]. 2014.
[131] Miao Y, Yu L and Blunsom P, “Neural Variational Inference for Text Processing”. arXiv:1511.06038 [cs.CL]. 2015.

[132] Yang Z et al., “Improved Variational Autoencoders for Text Modeling using Dilated Convolutions”. arXiv:1702.08139 [cs.NE]. 2017.
[133] Rasmussen C.E. and Williams C.K.I., “Gaussian Processes for Machine Learning”, Chapter 3, MIT Press 2006, Massachusetts Institute of Technology, USA.
[134] Williams C.K.I. and Barber D., "Bayesian classification with Gaussian processes," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 12, 1998, pp. 1342-1351.
[135] Goodfellow, I et al., "Generative Adversarial Networks". 2014, arXiv:1406.2661
[136] Fernandez G. M. et al., “Ugr'16: a new dataset for the evaluation of cyclostationarity-based network IDSs”. In Computers & Security, 2017. https://nesg.ugr.es/nesg-ugr16/
[137] Wolpert D. H. and Macready W. G., "No free lunch theorems for optimization," in IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67-82, Apr 1997.
[138] Sønderby, C.K. et al., “Ladder Variational Autoencoders”. 2016, arXiv:1602.02282v3.
[139] Johnson, M.J. et al., “Structured VAEs: Composing Probabilistic Graphical Models and Variational Autoencoders”. 2016, arXiv:1603.06277v1.
[140] Chopra S., Hadsell R. and LeCun Y., "Learning a similarity metric discriminatively, with application to face verification," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, pp. 539-546 vol. 1.
[141] AI in Telecom survey (November 2017) web: “https://www.h2o.ai/telecom/ ”, html, as of Nov 29, 2018
[142] Forni A. A., “Gartner Identifies the Top 10 Strategic Technology Trends for 2017”, Gartner,” http://www.gartner.com/newsroom/id/3482617 “ , html, as of Nov 29, 2018
[143] Graeme de Garis A., “Scalable Gaussian process inference using variational methods”. Phd thesis. Department of Engineering, University of Cambridge, September 2016.
[144] Santoro A. et al., “One-shot Learning with Memory-Augmented Neural Networks”, May 2016,arXiv:1605.06065 [cs.LG]
[145] Vinyals O. et al., “Matching Networks for One Shot Learning”, Jun 2016, arXiv:1606.04080 [cs.LG]
[146] Cannady, J. (2000, October). “Next Generation Intrusion Detection: Autonomous Reinforcement Learning of Network Attacks”. Proceedings of the 23rd National Information Systems Security Conference.
[147] Zhu M., Hu Z. and Liu P., “Reinforcement Learning Algorithms for Adaptive Cyber Defense against Heartbleed”. ACM Workshop on Moving Target Defense (MTD '14). New York, NY.
[148] Mao H. et al., “Resource Management with Deep Reinforcement Learning,” Proc. HotNets 2016, pp. 50–56.
[149] Kurakin A., Goodfellow I.J. and Bengio S., “ADVERSARIAL MACHINE LEARNING AT SCALE”. arXiv:1611.01236v2 [cs.CV] 11 Feb 2017
[150] Yuan X. eta al., “Adversarial Examples: Attacks and Defenses for Deep Learning”. arXiv:1712.07107v3 [cs.LG] 7 Jul 2018
[151] Corbett-Davies S., “Algorithmic decision making and the cost of fairness”. arXiv:1701.08230v4 [cs.CY] 10 Jun 2017

III. PAPERS

PAPER 1

Review of methods to predict connectivity of IoT wireless devices

Manuel Lopez Martin, Antonio Sanchez-Esguevillas and Belen Carro

Dpto. TSyCeIT, ETSIT, Universidad de Valladolid, Paseo de Belén 15, Valladolid 47011, Spain ;

mlopezm@acm.org ; antoniojavier.sanchez@uva.es; belcar@tel.uva.es

Abstract: Services related to Internet of Things (IoT) demand agile anticipation and response to eventual lack of service continuity. Machine learning methods may obtain predictions of IoT wireless sensors and devices connectivity patterns, enabling telcos to adjust maintenance periods, plan network upgrades, estimate outages risk and best allocate customer value to connection resources.

This article analyses how different algorithms forecast near/medium term connectivity of IoT wireless devices, based on their historical activity. The study considers time-series algorithms (Hidden Markov Model, exponential smoothing, AutoRegressive Integrated Moving-Average (ARIMA)), non-time-series algorithms (logistic regression, bayesian logistic regression, random forest, Gradient Boosting Method), mixed approaches (ARIMA with eXogenous covariates (ARIMAX)) and combinations of classifiers. Real obfuscated data obtained from a telecommunications operator is employed. Results present very advantageous prediction performance of IoT connectivity wireless devices, with an accuracy of over 90 percent most of the time, and even higher for the best performing algorithms.

Keywords: machine learning algorithms; Internet of Things; time-series prediction; wireless sensors; wireless devices.

---

1. Introduction

With the advent of IoT there is an exponential growth of wireless sensors and wireless devices in general that need a wireless connection to send and/or receive information. These wireless devices may act alone, e.g. a wearable like a smartwatch or smartband connecting to a smartphone or belong to a complex system like a smart home, e.g. a temperature sensor sending outdoor temperature to the residential gateway that controls the whole smart home system.

Therefore, there is a huge number of wireless devices connecting on a frequent basis (periodic or aperiodic) to some central collecting server, many of these devices being wireless

sensors that send environmental information to a server or to other devices. The information received by the server can be part of a business service provided to a final customer or can be part of operating services needed to fulfil other business life-cycle activities (fault management, activity mediation, billing…). These unattended devices, connecting in an automatic way to other devices or to a central server, are part of the IoT, where thousands if not millions of these devices provide a complex and inter-twined network. Needless to say, that IoT will be one of the mainstreams of automation and service delivery in the following years and their growth and importance is increasing rapidly.

From the point of view of an IoT Service Operator (namely a telecommunications operator) it is extremely useful to know in advance the probability distribution of wireless devices connectivity. Forecasting how likely is for a wireless device to send or not to send information during a period of time in the future is important, in order to: 1) anticipate business impact, 2) accommodate maintenance activity periods to lower the impact in connectivity, 3) enhance infrastructure to reduce risk for highly important services and associated devices.

This paper shows the results of applying different machine learning algorithms to forecast the activity/no-activity (on/off) of a wireless device connection, based on its past activity. For us, the activity of a device in a certain period is a binary value; either it is "on" when the device has sent data during that period, or, "off" when, otherwise, the device has not sent any data during that period.

A large group of wireless devices with heterogeneous connectivity patterns has been used. The results of applying the various methods to real wireless connections are compared (no simulated data was used in this work). The prediction accuracy has been used as the performance indicator for the analysis, obtaining its mean value throughout several hours of prediction.

The paper is organized as follows: Section 2 describes the followed methodology. Section 3 describes the results obtained from the different methods. Section 4 presents related works and finally, sections 5 and 6 provide the discussion and conclusions.

2. Materials and Methods

The objective has been to identify the best algorithm able to predict the on/off connection activity for periods of one hour in a foreseen window of two days (48 hours). We have considered the device to have connection activity as far as it sent any data during the one-hour period.

2.1. Methodology and tools

The steps followed are the typical ones of data science, namely:

- Perform a preliminary exploration/analysis of the available historical data
- Propose the prediction methods to use
- Pre-process the historical data
- Establish possible validation methods
- Define the forecasting accuracy comparison method (so called cost/utility function, in order to minimize forecasting error) which allows to benchmark the results of the different methods
- Obtain results from different methods
- Analyze results, propose adjustments or new methods to explore

This work has been executed in a desktop PC with Intel i7 processor and 16 GB of RAM, using R and RStudio software. All the R packages used are open source and freely available.

2.2. Algorithms

Regarding the prediction methods, taking into account the time-series nature of the data it was natural to consider time series forecasting methods; additionally, it seems to make sense to explore the adequacy of non-time–series methods by using other variables for prediction: e.g. time of day (hour of day), day of the week, customer identity, access point, etc.

In this study we have evaluated the following time-series methods: Hidden Markov Model (HMM), Exponential Smoothing, ARIMA and ARIMAX. And the following non-time-series methods: Logistic Regression, Random Forest, Gradient Boosting Method (GBM) and Bayesian Logistic Regression. It is out of scope of the paper to explain the details of the algorithms and good references cover them and are indicated through the paper, e.g. random forest [1] and GBM [2] are both based on decision trees.

2.3. Evaluation

To evaluate the results, in order to calculate the prediction performance of the different methods, the last 8 days were used (making forecasts for 4 consecutive periods of 48 hours, based on the available data of the previous days).

In order to optimize available data, we have used a variant of cross-validation (CV) applied to time series data. We have divided the training and testing data in 4 blocks, and for each consecutive block we have added to the training data the test data of the previous block.

We have trained individually all the methods for each Subscriber Identification Module (SIM) and for each cross-validation block (x4). Finally, we have performed an averaging process (along SIMs and cross-validation blocks) to have a final performance result for the 48 hours forecasting period.

In Figure 1 we present graphically the evaluation process; to obtain the performance results for the different methods we have used a kind of cross-validation applied to the time series data coming from the different devices. We have averaged results coming from the cross-validated test results and for all devices.

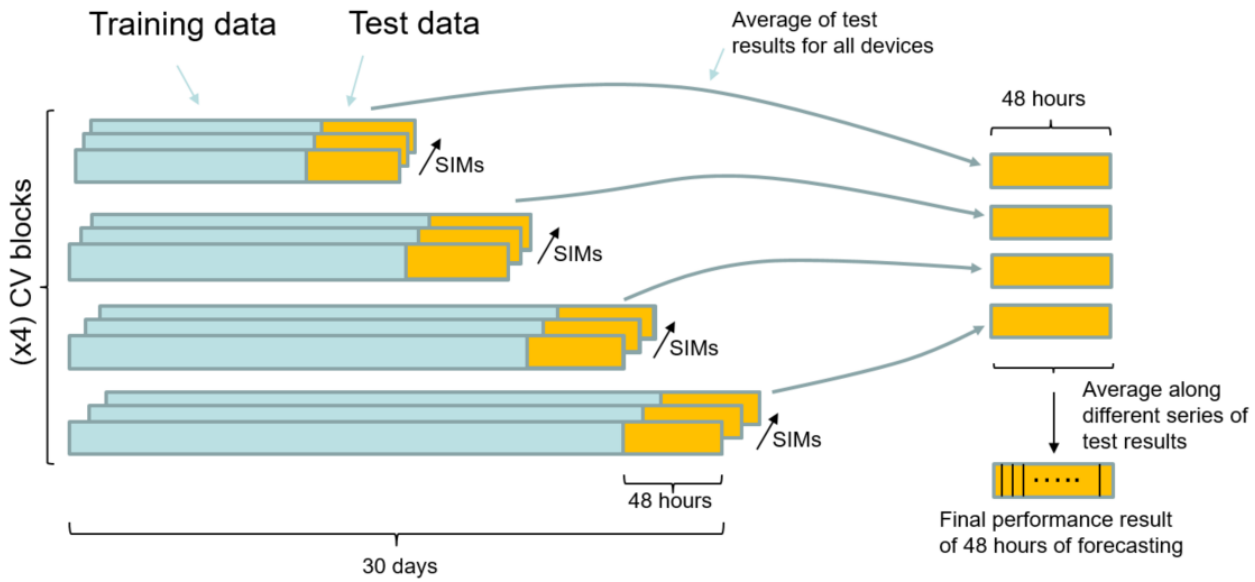


Figure 1. Evaluation process

As for the forecasting function to maximize, the simple and well-known accuracy function has been used:

$$accuracy = \frac{total\ number\ of\ correct\ predictions}{total\ number\ of\ predictions} \quad (1)$$

2.4. Data

First it is important to note that, in any forecasting problem there is a tradeoff between the so called bias (also known as underfitting, in which the algorithm fails to predict the relation between the features of past data and future data) and the variance (also known as overfitting, in which the algorithm only works well with a subset of past data but does not generalize well) [3]. For the forecasting problem presented here the main issue is at the bias side, which means that it is more important to discover additional important features, than increasing the amount of data.

This work is based on real (from a telecommunications operator) activity data (bytes sent/received) from 6,214 mobile devices (identified by their SIMs) for a period of almost 30 consecutive days (4/4/2015 to 1/5/2015, which includes public holidays making the forecast more challenging). For every SIM available data includes the starting connection time, duration of the connection and total transmitted data. The SIMs had different origins in terms of access point (so called Access Point Name (APN)), geographic area, customer and subscribed rate plan.

The exploratory data analysis showed that the connectivity data was quite irregular, not showing any representative pattern between devices. Neither was any clear pattern considering the other features like access point, region or even customer as aggregating variables.

Taking into account the proposed non-time-series prediction methods, it was necessary to perform a previous preparation of data, with two main purposes: 1) Transform connection activity of SIMs to a one-hour period aggregated activity (from the initial time connection data), and 2) Present the data in a form that could be used by both the time-series and non-

time-series methods. In Figure , the final format used is shown, with columns having: SIM identification (ID), date/hour, several numeric indices which provide information about day of the week, hour of the day account number, access point and rate plan.

The data is presented in sequential time-ordered along rows grouped by SIM. The data in Figure corresponds to a block of data for a particular SIM ordered by time, following this data block we have a consecutive data block for other SIM ordered in the same way and so on. This data arrangement allows to apply all the methods under study.

TRANSFORMED DATA

Time

SIM id	date/time	day/week	hour/day	hour index	day/month	data traffic	activity	account	rate_plan	apn
4666	2015-04-04 16:00:00	7	16	1	4	0.0	FALSE	29	43	55
4666	2015-04-04 17:00:00	7	17	2	4	0.0	FALSE	29	43	55
4666	2015-04-04 18:00:00	7	18	3	4	0.0	FALSE	29	43	55
4666	2015-04-04 19:00:00	7	19	4	4	0.0	FALSE	29	43	55
4666	2015-04-04 20:00:00	7	20	5	4	0.0	FALSE	29	43	55
4666	2015-04-04 21:00:00	7	21	6	4	0.0	FALSE	29	43	55
4666	2015-04-04 22:00:00	7	22	7	4	0.0	FALSE	29	43	55
4666	2015-04-04 23:00:00	7	23	8	4	0.0	FALSE	29	43	55
4666	2015-04-05 00:00:00	1	0	9	5	0.0	FALSE	29	43	55
4666	2015-04-05 01:00:00	1	1	10	5	0.0	FALSE	29	43	55
4666	2015-04-05 02:00:00	1	2	11	5	0.0	FALSE	29	43	55
4666	2015-04-05 03:00:00	1	3	12	5	0.0	FALSE	29	43	55
4666	2015-04-05 04:00:00	1	4	13	5	0.0	FALSE	29	43	55
4666	2015-04-05 05:00:00	1	5	14	5	0.0	FALSE	29	43	55
4666	2015-04-05 06:00:00	1	6	15	5	0.0	FALSE	29	43	55
4666	2015-04-05 07:00:00	1	7	16	5	961.0	TRUE	29	43	55
4666	2015-04-05 08:00:00	1	8	17	5	1168.0	TRUE	29	43	55
4666	2015-04-05 09:00:00	1	9	18	5	1001.0	TRUE	29	43	55
4666	2015-04-05 10:00:00	1	10	19	5	1001.0	TRUE	29	43	55
4666	2015-04-05 11:00:00	1	11	20	5	1243.0	TRUE	29	43	55
4666	2015-04-05 12:00:00	1	12	21	5	961.0	TRUE	29	43	55
4666	2015-04-05 13:00:00	1	13	22	5	1243.0	TRUE	29	43	55
4666	2015-04-05 14:00:00	1	14	23	5	1001.0	TRUE	29	43	55

One block of data per SIM

Figure 2. Transformed data format used for all methods.

The processed dataset contains circa 4 million entries, each one indicating whether data has been transmitted by a given SIM in a given hour or not.

3. Results

3.1 Results from non-time-series methods

In Figure 3, we present the results for the non-time-series methods and in following paragraphs the details about each method. The best results are obtained for random forest (see Figure 8). The upper chart in Figure 3 gives the mean accuracy of prediction (for all SIMs) in periods of one-hour for a prediction interval of 48 hours. The bottom chart gives the standard deviation (amount of variation) of the mean prediction accuracy (please note the opposite nature of both charts, as a higher accuracy is associated with a lower standard deviation).

It is interesting the periodic nature of the forecasting performance. A sharp reduction in performance, as the prediction time increases, would have been more expected, but the results

show a maximum in performance for a 24 hours period and a minimum around a 12 hours period. This behavior is similar for all methods. An explanation for this behavior is given in section 5, connecting the mean global activity of the SIMs with the mean accuracy of the forecasts.

The training speed (i.e. the time it takes for the algorithms to tune its parameters) is very high for all methods except GBM which is much slower (see section 3.3).

Figure 3 shows the performance results for non-time-series methods; the upper diagram presents the mean accuracy over a prediction period of 48 hours. The mean accuracy is calculated using the process presented in section 2.3 (Figure 1). Lower diagram presents the standard deviation of the accuracy values used to build the upper diagram.

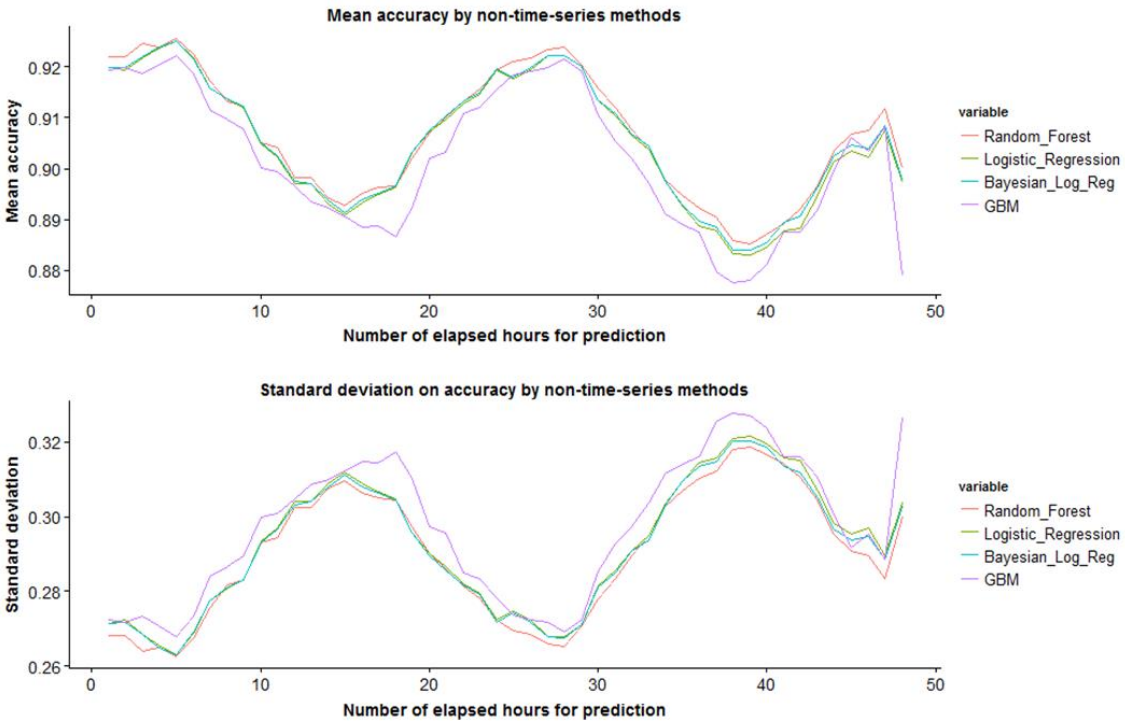


Figure 3. Performance results for non-time-series methods.

As already mentioned, the non-time-series methods explored have been: Logistic regression, Bayesian logistic regression, Random Forest and GBM. All of them are well known methods with good performance in several areas of application.

For all these methods, a training of the algorithm for each particular SIM was performed, using the day of the week (7 possible values) and hour of day (24 values) as predictor variables, and the on/off activity in one-hour periods as the predicted variable. We considered these features as the best due to the time-series nature of the data. We tried to add additional predictors related with other time elapsed periods in hours (e.g. 2 or 4-hour periods) not improving the results significantly. Other available features were not used since the computational needs would have increased substantially. Intuitively another interesting feature to explore could be the customer, since devices from the same customer may have similar traffic patterns (e.g. a smart meter from a utility, or a connected vehicle from a car manufacturer); this feature could be explored in future work.

The results for logistic regression were quite satisfactory; nevertheless, we incurred in a complete separation (also named perfect separation) problem during training. This problem happens when one or several independent variables can fully predict the result, this usually implies over-fitting (results are good for the training set but not as good for the real set). To

avoid this problem, two possible solutions are: 1) Firth logistic regression [4] or 2) Bayesian logistic regression [5]. We used the second approach. However, the results from both methods are very close.

The next method to try was random forest, for which default parameters for all training (as provided by the R package: randomForest) were used. The results obtained from random forest are the most accurate for non-time series methods.

We wanted finally to examine GBM. Considering that, as we will detail later on, our problem seems to have more troubles from the bias than from the variance (over-fitting) side, we expected good results from this method. We used default parameters when training (as provided by the R package: xgboost). We decided not to adjust the parameters for each SIM, because the number of SIMs (and training rounds) was too large, so we used the same default parameters for all trainings. This had an impact in GBM performance, due to the sensitivity of GBM to fine parameters tuning. We think that is the reason of the poor results from this method which were worse than expected.

Other standard methods like Support Vector Machines (SVM) and Artificial Neural Networks (ANN) were not considered in the tests, since these methods are highly demanding in processing time for model parameters tuning and they usually require an exhaustive parameters adjustment which could not be performed given the number of models to tune (as big as the number of devices).

Having the prediction accuracy as our main performance indicator, we did not consider other possible indicators as false and true positive rate, sensitivity and specificity (proportion of positives or negatives respectively, that are correctly identified as such), etc. Nevertheless, we did perform a quick assessment on representative values of false and true positive rates obtained, at least, from some of the methods results.

The best way to analyze the false and true positive rate behavior is using a so-called Receiver Operating Characteristic (ROC) curve. In Figure 4 we present the ROC curve for the results obtained with Random Forest; being, in this case, the value for the Area Under the Curve (AUC) of 0.956, which is quite good, as an AUC value of 1 is considered a perfect result. Both ROC and AUC show very good results.

Figure 4 shows the different false and true positive rates obtained as we change the probability cut-off threshold. This threshold defines which values will be considered as positive or negative. The numbers inside Figure 4 provide explicitly some of these cut-off values (with all values, from 0 to 1, as a color gradient).

For all methods that require a probability cut-off threshold, we have used a value of 0.5. From Figure 4, we can see that, in the case of random forest, a cut-off value of 0.5 provides a false positive rate of around 0.1, and a true positive rate of 0.9, which are both quite good.

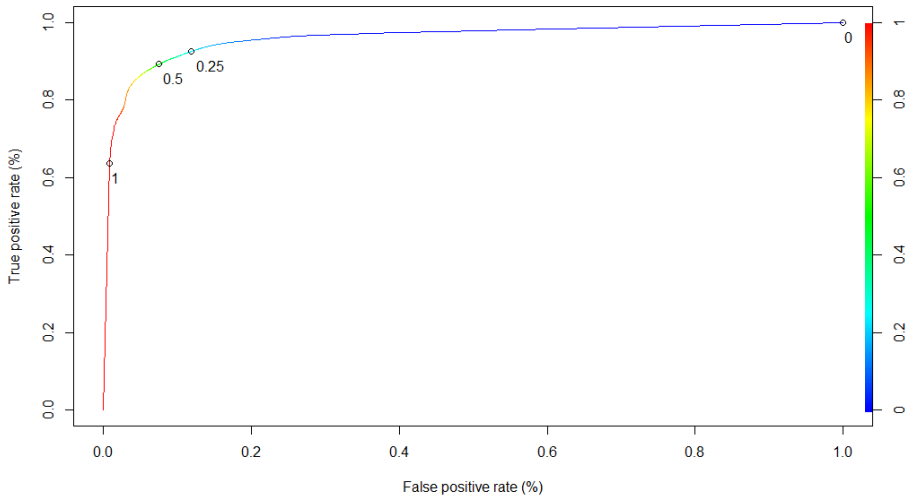


Figure 4. ROC curve for Random Forest prediction.

Another interesting insight has been obtained when analyzing the dependence of the prediction performance of the different methods with the amount of data available for training. We have detected that the methods actually do not need too much data to accomplish its highest performance, as it can be seen in Figure . The mean accuracy achieved by random forest (similar for logistic regression) does not significantly increase when incrementing the number of days given for training. To obtain the values in Figure 5 we have performed the validation method explained in section 2.3, and, finally, we have averaged the 48 hours predictions in single value; carrying out this process for different numbers of days of training.

This result is again due to the fact that we are not incurring in overfitting (where adding more data usually reduces the overfitting problem and increases predictive performance).

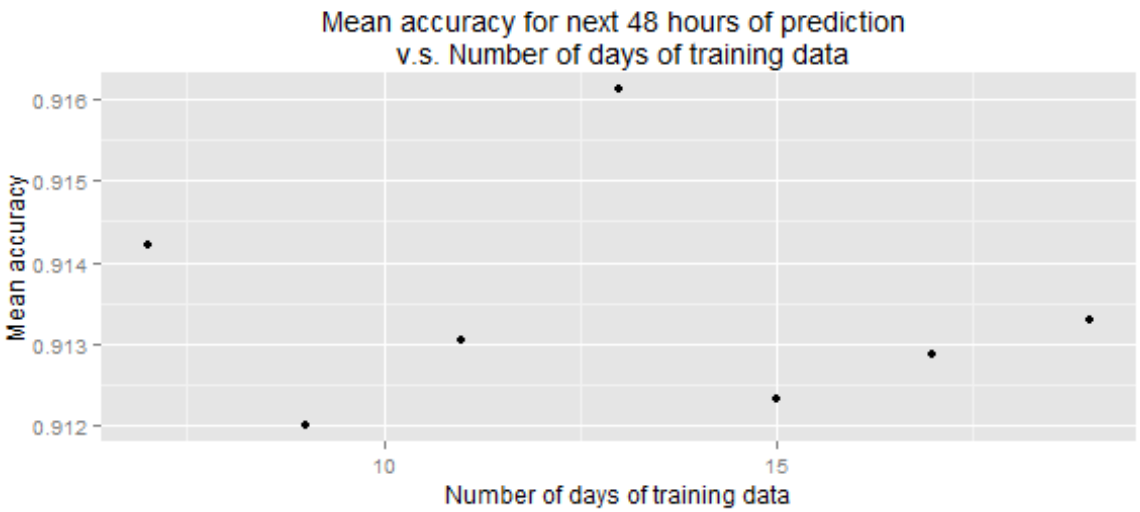


Figure 5. Mean prediction accuracy of random forest for a period of 48 hours vs. the number of days used for training.

3.2 Results from time-series methods

The time-series methods considered have been: HMM, Exponential Smoothing (ETS), ARIMA and ARIMAX.

In Figure 6, we present the results for the time-series methods and, in following sections, the details about each method. The upper and bottom charts in the figure give similar information to Figure 3, being the discussion on results also similar.

In this case, the best results are obtained for ARIMAX (see Figure 8). The training speed is similar for ARIMA and ETS, and much slower for HMM and ARIMAX, being ARIMAX particularly slow. Training speed for all time-series methods has been worse than for non-time series methods (see section 3.3). In fact, this difference in computing requirements and training speed should be taken into consideration when choosing the final method in a production environment.

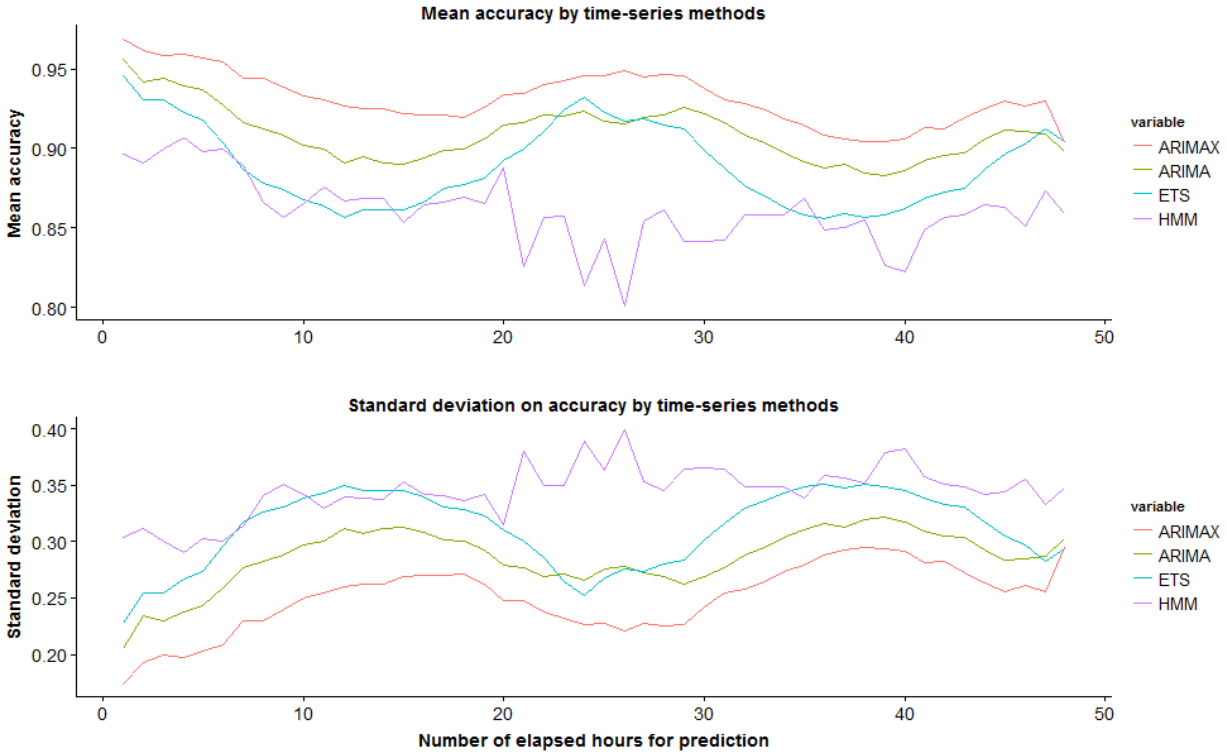


Figure 6. Performance results for time-series methods.

The first time-series method we tried was a HMM [6]. We performed a training of the algorithm for each particular SIM, using as predicted variable the on/off activity during the predicted period. Training an HMM implies defining its four parameters (number of states, observation values and transition and emission probabilities) providing an observation sequence as training data.

In our case the observations have two values (on/off SIM activity). We used two hidden states; we observed that increasing the number of hidden states did not improve the results. To obtain the transition and emission probabilities we applied the Baum–Welch algorithm [6] which is a special case of the expectation maximization algorithm.

Though HMM usually has good prediction performance for this kind of problems, its results

were not as expected but unusually poor.

The next time-series method tried was Exponential Smoothing [7], performing also a training of the algorithm for each particular SIM, using as predicted variable the on/off activity during the predicted period. The results for exponential smoothing are neither good (see Figure 8). The periodic nature of the signals and the rapid loss of memory of this method, due to its exponential decreasing factor for passed times, may be the reason for its poor performance.

The following method tried was ARIMA [8], in a similar way to the other methods; we performed a training of the algorithm for each particular SIM, using as predicted variable the on/off activity during the predicted period. The results for ARIMA are very good. The reasons behind these advantageous results are, mainly, the fine automatic parameters adjustment done by the auto.arima function from the forecast R package. The auto.arima function automatically adjusts the parameters ("(p,d,q)(P,D,Q)m"), considering also seasonality.

The use of an algorithm that automatically adjusts the above-mentioned parameters is critical, since the manual adjustment using ACF (Auto Correlation Function) and PACF (Partial Autocorrelation Function) [8] was not possible, because adjusting the parameters for each SIM would be extremely computationally demanding.

Finally, ARIMAX was applied [8], performing again a training of the algorithm for each particular SIM, using the day of the week (7 possible values) and hour of day (24 values) as predictor variables (also named covariates) and the on/off activity in one-hour period as the predicted variable.

For this particular problem, preparing the additional data set of covariates required by ARIMAX was an easy task, considering that the day of the week and the hour of a day are known information, both for the training and testing data, in other occasions, in order to prepare the testing covariates an additional prediction task is necessary.

For ARIMAX we used also the auto.arima function from the forecast R package, providing an additional data set with values of the covariates for each value of the training data (the on/off activity variable in the training data). This additional data set is necessary, since ARIMAX uses these external covariates as predictors on top of the usual past values and errors from the time-series.

The results from ARIMAX have been the best of all the applied methods. The use of the ARIMAX [9] method was not initially planned but was considered after examining the good behavior of ARIMA and the right results coming from the non-time-series methods (logistic regression and random forest), even when just using two predictors (time of day and day of the week).

The SIM's activity distribution seems to have a clear structure defined both from its past activity and, equally important, from external predictors as the day of the week and hour of day. The ARIMAX method is able to combine both aspects, as the "ARIMA part" takes into account the past of the signal and the "exogenous covariates part" considers other external variables to incorporate to the prediction. In this case we have precisely considered the day of the week and the hour of day as these "exogenous" external variables.

Similarly, to the non-time-series methods, we have come to the conclusion that, having a minimum number of days for training, the performance does not seem to improve by increasing the amount of days. Actually, we have seen that for ARIMA, the method does not improve its performance when increasing the number of days of training beyond 5-7 days (see Figure 5 for similar behavior for non-time-series methods).

ARIMA performs slightly better than random forest (the best non-time series) but from a practical point of view their performances are identical (see Figure 8).

From a computing perspective, the faster of all methods is logistic regression (similar to bayesian logistic regression). The ranking is followed by ETS, random forest, GBM, ARIMA,

HMM, and ARIMAX (see Figure 7).

3.3 Other results

In average, the devices are active 58% of the time (this can be seen as well at the bottom chart in Figure). That implies that our baseline accuracy is 0.58 as this value can be obtained simply by giving a constant value of 'on' activity for all devices and times.

In order to improve the final prediction, we tried to ensemble [10] the results from the different methods using a voting schema to choose the final one, and, in all cases, the result obtained was worse than the result from the best algorithm used in the input mix. This bad result from ensembling, suggests again that there is not a variance problem and that prediction improvement will come from an improvement at the bias side.

It is interesting to compare the computing requirements for the training of the different methods. In Figure 7 we provide the results; the values correspond to training times for 6214 SIMs for a forecasting interval of 48 hours. The complete process was done 4 times for consecutives intervals of 48 hours, in order to have significant data to compute the average of the forecasting values. In the figure, training times are given in a log scale due to the big difference between values.

We can see that the non-time-series methods requires in general less computing time, and that ARIMAX and HMM require much more computing time that the rest.

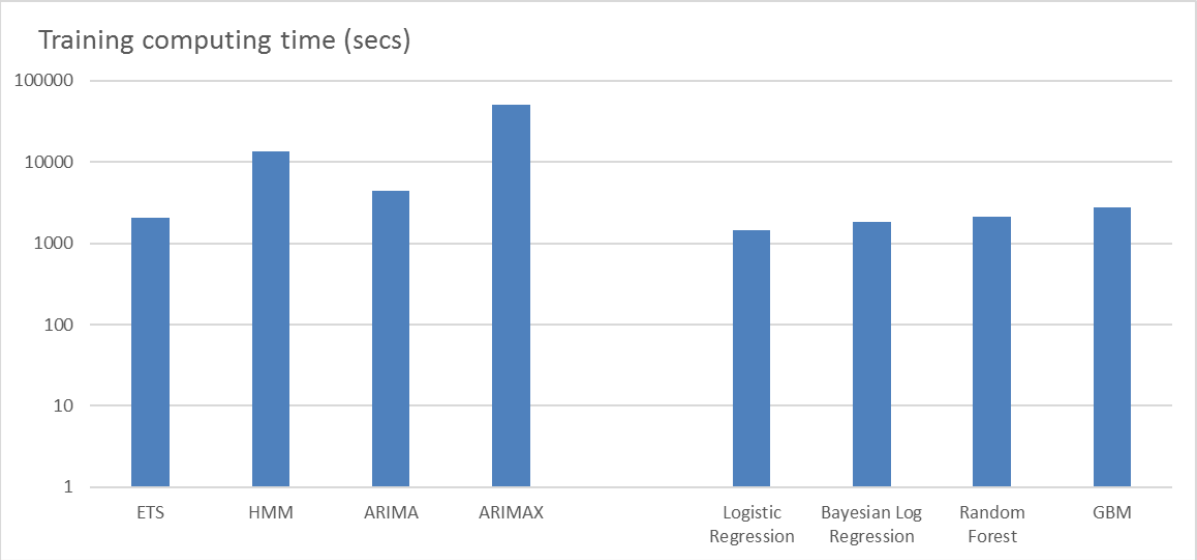


Figure 7. Computing time used to train the different methods.

3.4 Considerations about the forecasting interval

The forecasting interval used for the study has been a 1-hour interval, performing all forecasting calculations over the next 48 1-hours. A 1-hour interval value has been also used to aggregate the on/off activity for each SIM for the training data.

The 1-hour period used both as the training feature (activity) to perform predictions, and, as

the forecasting period to obtain results, has been chosen due to business adequacy, and for being a value small enough to have significant results, and big enough to accommodate and filter the usually noisy activity of the SIMs.

4. Related work

Previous studies have been performed on connectivity predictions for wireless networks and wireless sensor networks. Most of this previous work is focused in ARIMA and Artificial Neural Networks methods [11,9,12], and some of them in Support Vector Machines and logistic regression [13,14].

The novelty of the presented work is based on two main aspects. First, our study provides new insights comparing the results for time-series and non-time-series methods, applying the methods to a large number of devices with very different connectivity behaviors. In particular, the results obtained from the application of random forest are apparently new.

And second, while most of previous works on this subject have focused on the prediction of data volume transmitted (which is a continuous variable), this article focuses on predicting on/off connectivity (a discrete variable), which constitutes a new starting point and a different issue itself, providing novel results. Besides, the work is carried out employing real activity data (bytes sent/received) offered by a telecommunications operator.

5. Discussion

The main objective of this study has been the identification of a series of machine learning methods suitable to predict the busy/idle connectivity states of IoT wireless devices, being able to select the ones with most accurate performance.

The main original contributions are shown:

- One week of historical data is enough to provide good forecasts
- The results present a very high forecasting accuracy, which can be successfully used for the prediction of connectivity state of IoT wireless devices, in order to avoid lack of service problems and improve the quality of service
- Method with best absolute accuracy performance is ARIMAX
- However, not so much lower accuracy can be obtained with other methods (not ARIMAX) and at a lower computational cost. Therefore, for a production environment, it is recommended to employ the methods that require less computation and, nevertheless, give a very good accuracy, as it is the case for logistic regression, ARIMA or random forest.

In Figure 8 we present the global accuracy for all methods. This accuracy has been calculated averaging the mean accuracy over the 48 hours of the forecasting interval.

We have obtained a global accuracy over 90% for most of the methods (with the exception of HMM and ETS).

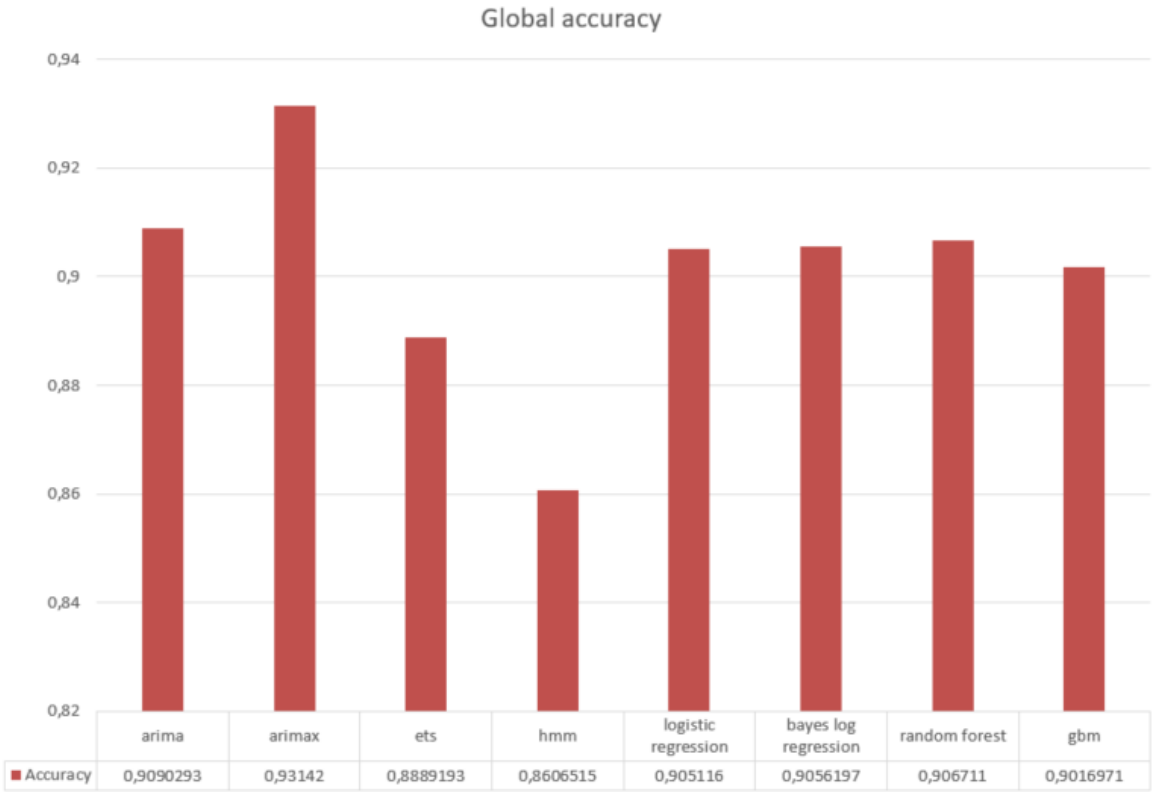


Figure 8. Global accuracy for all methods.

ARIMAX has excelled over the rest and has provided the highest prediction accuracy, with a global mean accuracy over 93%. Other algorithms as random forest, ARIMA and logistic regression have provided very good results as well, but some of the more promising algorithms as GBM have given poorer results, due to its higher sensitivity to hyper-parameters tuning and the inability to tune them for each particular device.

It is interesting the good behavior of the non-time-series methods (mainly random forest) considering that we have only used two predictors. It was also surprising the bad results from HMM and ensemble classifiers.

In Figure 9 we show the mean prediction accuracy and its standard deviation for all methods, and for each of the 48 hours of prediction interval; the upper chart presents the mean prediction accuracy for a 48 hours period of prediction; the intermediate chart provides the standard deviation of the mean prediction accuracy, and, the lower chart presents the mean activity of SIMs (percentage of active SIMs) for same prediction interval

One interesting outcome from the study has been to observe the periodic nature of the prediction accuracy. In Figure 9, we can observe that the periodic behavior of the accuracy over time is comparable for all the methods. We have added an additional chart to the figure (bottom chart) giving the percentage of active SIMs per hour (in the 48 hours prediction period), by looking at this chart we can see that the mean accuracy has an opposite dynamic to the percentage of active SIMs. This could explain the periodic behavior of the accuracy since it seems that the low activity periods are easier to predict.

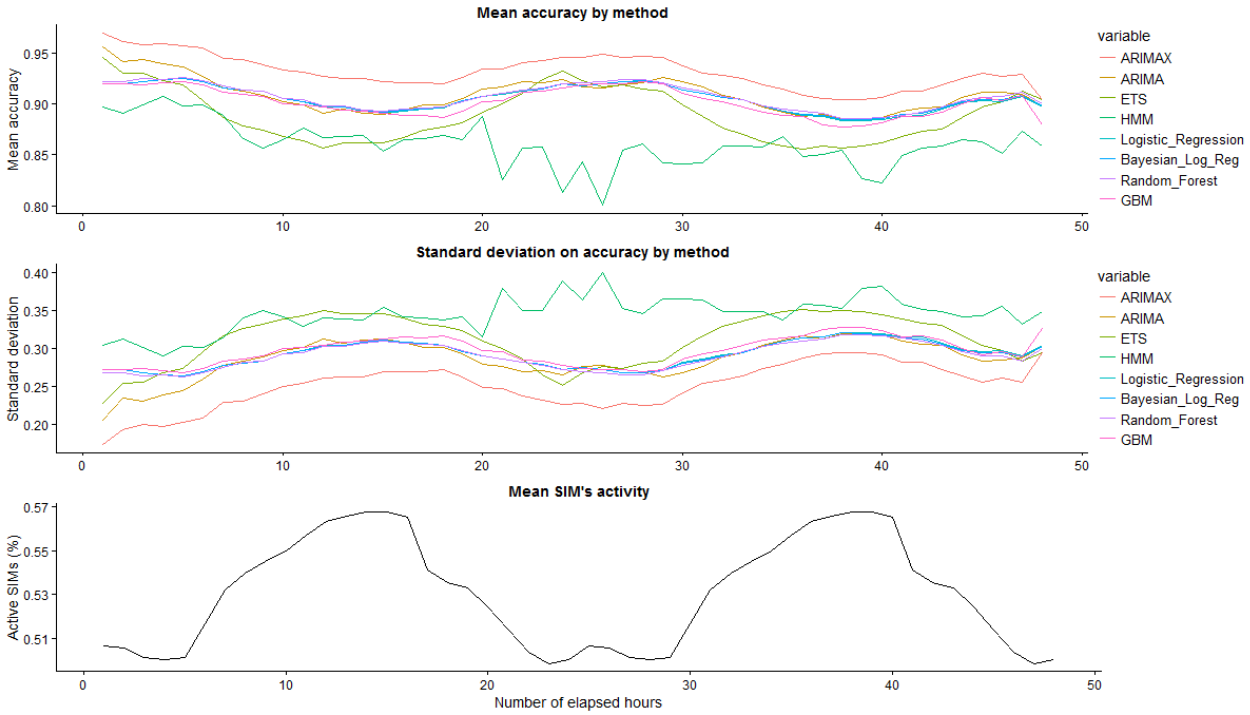


Figure 9. Comparison of all prediction methods.

We have observed that connectivity times have more structure and less noise than was initially predicted. This can justify the consistent behavior of the different methods and the coherence of the results obtained under the different strategies of validation and test.

6. Conclusion

This paper presents results of applying machine learning techniques to forecast the on-off activity state of a big number of IoT mobile devices, using time-series and no-time-series methods. All the results are based on data from real IoT mobile devices.

ARIMAX time-series method had the best accuracy (93%), but requires a huge training time. ARIMA and some non-time-series methods (logistic regression and random forest) also shown very good performances.

The non-time-series methods require, in general, less training time than the time-series-methods.

Considering the higher computational requirements for ARIMAX compared to logistic regression, random forest and ARIMA; the latter methods would be a better choice for a production environment (as they provide similar practical accuracy with less computing time).

Independently of the method, the on-off connectivity from IoT devices presents a rich periodic structure, allowing good prediction results, even with short training data.

Test results were achieved with a specifically developed cross-validation process, applied to both, time-series and non-time-series methods.

From the results obtained, we expect in future works to find additional predictors (covariates) that will probably improve the predicting power of the methods presented. One of these new predictors could be obtained by performing clustering of the signals, trying to use the cluster index as an additional new predictor. The main handicap will be the high

computational demand for this task.

Acknowledgments: We would like to express our gratitude for the support and data provided by Telefonica M2M, who is actively working in the application of machine learning techniques to optimize their business. The data used were originally obfuscated with no access to raw data at any time. Besides, this work has been partially funded by the Ministerio de Economía y Competitividad del Gobierno de España and the Fondo de Desarrollo Regional (FEDER) within the project "Inteligencia distribuida para el control y adaptación de redes dinámicas definidas por software, Ref: TIN2014-57991-C3-2-P", in the Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento.

References

[1] Leo Breiman, Random Forests. Machine Learning, Volume 45, Issue 1, October 1 2001, Pages 5-32

[2] Jerome H. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, The Annals of Statistics, Vol. 29, No. 5 (Oct., 2001), pp. 1189-1232

[3] Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning: Data mining, inference, and prediction. Second Edition (2009), Springer Series in Statistics, Pages 219-230

[4] Georg Heinze and Michael Schemper, A solution to the problem of separation in logistic regression, Statistics in Medicine, Volume 21, Issue 16 (30 August 2002), Pages 2409–2419.

[5] Andrew Gelman et al, A weakly informative default prior distribution for logistic and other regression models, The Annals of Applied Statistics 2008, Vol. 2, No. 4, Pages 1360–1383

[6] Lawrence R. Rabiner, A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE, Vol. 77, No. 2. (06 February 1989), Pages. 257-286

[7] Charles C Holt, Forecasting Trends and Seasonal by Exponentially Weighted Averages. International Journal of Forecasting, Volume 20, Issue 1 (January–March 2004), Pages 5–10.

[8] M. Xie 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] Cristina Stolojescu-Crisan, Data mining based wireless network traffic forecasting, 2012 10th International Symposium on Electronics and Telecommunications (ISETC), Timisoara (Nov 2012), Pages 115-118.

[10] Josef Kittler et al, On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 3 (March 1998), Pages 226-239.

[11] Papadopouli M, Evaluation of short-term traffic forecasting algorithms in wireless networks, 2006 2nd Conference on Next Generation Internet Design and Engineering, NGI, Valencia (April 2006), Pages 102-109

[12] Yantai Shu et al, Wireless traffic modeling and prediction using seasonal ARIMA models, IEEE International Conference on Communications, 2003 (ICC 2003), Anchorage

(May 2003), Vol. 3, Pages 1675-1679

[13] Fei Huang et al, Reliability Evaluation of Wireless Sensor Networks Using Logistic Regression, 2010 International Conference on Communications and Mobile Computing, Shenzhen (April 2010), Vol. 3, Pages 334-338

[14] Huifang Feng et al, SVM-Based Models for Predicting WLAN Traffic, 2006 IEEE International Conference on Communications (ICC 2006), Istanbul (June 2006), Vol. 2, Pages 597-602

PAPER 2

Network traffic classifier with convolutional and recurrent neural networks for Internet of Things

Manuel Lopez-Martin (Senior Member, IEEE), Belen Carro, Antonio Sanchez-Esguevillas

(Senior Member, IEEE) and Jaime Lloret (Senior Member, IEEE)

***Abstract*— A Network Traffic Classifier (NTC) is an important part of current network monitoring systems, being its task to infer the network service that is currently used by a communication flow (e.g. HTTP, SIP…). The detection is based on a number of features associated with the communication flow, for example, source and destination ports and bytes transmitted per packet. NTC is important because much information about a current network flow can be learned and anticipated just by knowing its network service (required latency, traffic volume, possible duration…). This is of particular interest for the management and monitoring of Internet of Things (IoT) networks, where NTC will help to segregate traffic and behavior of heterogeneous devices and services. In this paper, we present a new technique for NTC based on a combination of deep learning models that can be used for IoT traffic. We show that a Recurrent Neural Network (RNN) combined with a Convolutional Neural Network (CNN) provides best detection results. The natural domain for a CNN, which is image processing, has been extended to NTC in an easy and natural way. We show that the proposed method provides better detection results than alternative algorithms without requiring any feature engineering, which is usual when applying other models. A complete study is presented on several architectures that integrate a CNN and an RNN, including the impact of the features chosen and the length of the network flows used for training.**

***Index Terms*—Convolutional Neural Network; Deep Learning; Network traffic classification; Recurrent Neural Network**

I. INTRODUCTION

A Network Traffic Classifier (NTC) is an important part of current network management and administration systems. An NTC infers the service/application (e.g. HTTP, SIP...) being used by a network flow. This information is important for network management and Quality of Service (QoS), as the service used has a direct relationship with QoS requirements and user contracts/expectations.

It is clear that Internet of Things (IoT) traffic will pose a challenge to current network management and monitoring systems, due to the large number and heterogeneity of the connected devices. NTC is a critical component in this new scenario [1, 2], allowing to detect the service used by dissimilar devices with very different user-profiles. Network traffic identification is crucial for implementing effective management of network policy and

resources in IoT networks, as the network needs to react differently depending on traffic profile information.

There are several approaches to NTC: port-based, payload-based, and flow statistics-based [3, 4]. Port-based methods make use of port information for service identification. These methods are not reliable as many services do not use well-known ports or even use the ports used by other applications.

Payload-based approaches the problem by Deep Packet Inspection (DPI) of the payload carried out by the communication flow. These methods look for well-known patterns inside the packets. They currently provide the best possible detection rates but with some associated costs and difficulties: the cost of relying on an up-to-date database of patterns (which has to be maintained) and the difficulty to be able to access the raw payload. Currently, an increasing proportion of transmitted data is being encrypted or needs to assure user privacy policies, which is a real problem to payload-based methods.

Finally, flow statistics-based methods rely on information that can be obtained from packets header (e.g. bytes transmitted, packets interarrival times, TCP window size…). They rely on packet header high-level information which makes them a better option to deal with non-available payloads or dynamic ports. These methods usually rely on machine learning techniques to perform service prediction [3]. Two machine learning alternatives are available in this case: supervised and unsupervised methods. Supervised methods learn an association between a set of features and the desired labeled output by training an algorithm with samples containing ground-truth labeled outputs. In unsupervised methods, we do not have data with their associated ground-truth labeled outputs; therefore, they can only try to separate the samples in groups (clusters) according to some intrinsic similarities.

In this paper, we propose a new flow statistics-based supervised method to detect the service being used by an IP network flow. The proposed method employs several features extracted from the headers of packets exchanged during the flow lifetime. For each flow, we build a time-series of feature vectors. Each element of the time-series will contain the features of a packet in the flow. Likewise, each flow will have an associated service/application (a labeled value) which is required to train the algorithm.

To ensure data confidentiality our method only makes use of features from the packet's header, not including the IP addresses.

In order to train the method, we have used more than 250,000 network flows which contained more than 100 distinct services. As an additional challenge, the frequency distribution of these services was highly unbalanced.

The proposed method is a classifier based on a deep learning model formed by the combination of a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN).

One of the main drivers of this work was to assess the applicability of deep learning advances to the NTC problem. Therefore, we have studied the adequacy of different deep learning architectures and the influence of several design decisions, such as the features selected, or the number of packets per flow included in the analysis.

In the paper, we present a comparison of performance results for different architectures; in particular, we have considered RNNs alone, CNNs alone and different combinations of CNN and RNN.

In order to apply a CNN to a time-series of feature vectors, we propose an approach that renders the data as an associated pseudo-image, to which CNN can be applied.

When assessing the suitability of a new method it is important to apply it to real data. We have made use of data from RedIRIS, which is the Spanish academic and research network.

The paper is organized as follows: Section II presents the related works. Section III describes

the work performed. Section IV describes the results obtained and finally, Section V provides discussion and conclusions.

II. RELATED WORKS

Comparison of work results is difficult in NTC because the datasets being studied and the performance metrics applied are very different. NTC is intrinsically a multi-class classification problem. There is no single universally agreed metric to present results in a multi-class problem, as will be shown in Section IV. Considering these facts, we now present several related works.

There are many works that apply neural networks to NTC, but the network models employed are very different in nature to the ones presented here.

In [5] they propose a multi-layer perceptron (MLP) with zero or one hidden layer, but it is actually adopted as the internal architecture to apply a fully Bayesian analysis. The best one vs. rest accuracy, using 246 features, for 10 grouped labels is 99.8%, and a macro averaged accuracy of 99.3% (10 labels).

An ensemble of MLP classifiers with error-correcting output codes is applied in [6], achieving an average overall accuracy (for 5 labels) of 93.8%. Meanwhile, in [7] an MLP with a particle swarm optimization algorithm is employed to classify 6 labels with a best one vs. rest accuracy of 96.95%. Somehow related, the purpose of [8] is to investigate neural projection techniques for network traffic visualization. Towards that end, they propose several dimensionality reduction methods using neural networks. No classification is performed. Another work [9] explores the applicability of rough neural networks to deal with uncertainty but does not give any performance results for NTC.

Zhou et al. [10] apply an MLP with 3 hidden layers and different numbers of hidden neurons to the Moore dataset [11]. They give an overall accuracy greater than 96%, for a grouping of labels in 10 classes, resulting in a final class distribution very unbalanced (a frequency of almost 90% for highest frequency class), no F1 score is provided. A Parallel Neural Network Classifier Architecture is used in [12]. It is made up of parallel blocks of radial basis function neural networks. To train the network is employed a negative reinforcement learning algorithm. They classify 6 labels reporting a realistic overall accuracy of 95%, no F1 score is provided.

Another set of papers applies general machine learning techniques, not related with neural networks, to the NTC problem.

Kim et al. [13] propose an entropy-based minimum description length discretization of features as a preprocessing step to several algorithms: C4.5, Naïve Bayes, SVM and kNN. Claiming an enhanced performance of the algorithms, achieving a one vs. rest accuracy of 93.2%- 98% for 11 grouped labels. In [14] authors apply different machine learning techniques to NTC (C4.5, Support Vector Machine, Naïve Bayes) reporting an average accuracy of less than 80% using 23 features and detecting only five services (www, dns, ftp, p2p, and telnet)

Wang et al. [15] employ an enhanced random forest with 29 selected features. They group the services in 12 classes, providing only one vs. rest metrics (not aggregated). Having F1 scores in the interval 0.3-0.95, with only 3 classes higher than 0.96. They use their own dataset. Authors of [16] include flows correlation in a semi-supervised model providing overall accuracy of less than 85% and a one vs. rest F1 score, for 10 labels, of less than 0.9 (except two labels with 0.95 and 1). They use the WIDE backbone dataset [16]. They report having better results than other works using C4.5, kNN, Naïve Bayes, Bayesian Networks and

Erman´s semi-supervised methods [17, 18, 19, 20, 21].

A Directed Acyclic Graph-Support Vector Machine is proposed in [22], attaining an average accuracy of 95.5%. The method is applied to a one-to-one combination of classes with a dataset provided by the University of Cambridge (Moore dataset) [11]. Yamansavascilar et al. [23] study the application of several algorithms: J48, Random Forest, Bayes Net, and kNN to UNB ISCX Network Traffic dataset, with 14 classes and 12 features, reporting the best accuracy of 93.94%.

Yuan et al. [24] present a variant of decision tree algorithm C4.5 working on the Hadoop platform. They classify 12 labels giving a one vs. rest accuracy in the interval 60-90% with only two labels higher than 90%. The dataset is the Moore set from Cambridge University [11].

In this paper, we present the first application of the RNN and CNN models to an NTC problem. The combination of both models provides automatic feature representation of network flows without requiring costly feature engineering.

III. WORK DESCRIPTION

Following sections present the dataset used for this work and a description of the different deep learning models that were applied.

A. Selected dataset

For this work, we have made use of real data from RedIRIS. RedIRIS is the Spanish academic and research backbone network that provides advanced communication services to the scientific community and national universities. RedIRIS has over 500 affiliated institutions, mainly universities and public research centers.

We have extracted 266,160 network flows from RedIRIS. These flows contained 108 distinct labeled services, with a highly unbalanced frequency distribution. Fig. 1 shows the names and frequency distribution for the 15 most frequent services. The frequency distribution is based on the proportion of flows with a specific service.

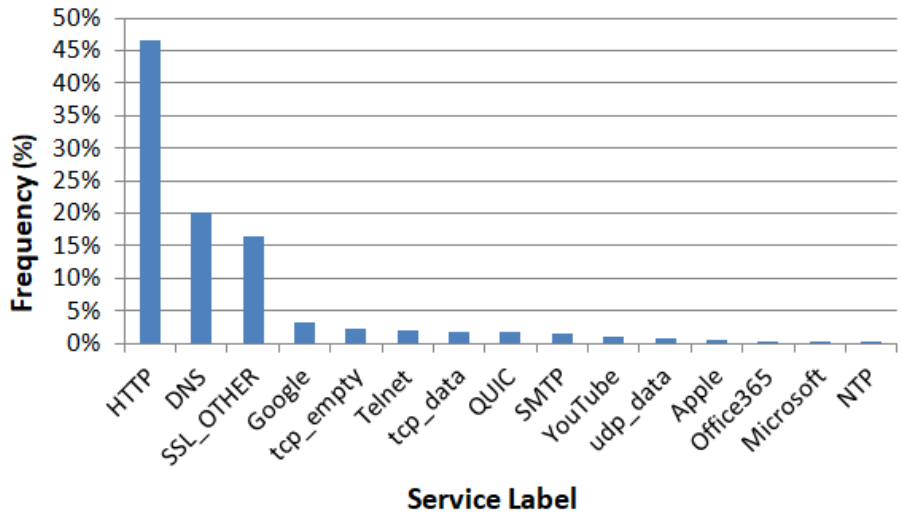


Fig. 1. Frequency distribution of the 15 most frequent services.

A network flow consists of all packets sharing a unique bi-directional combination of source and destination IP addresses and port numbers, and transport protocol: TCP or UDP. We include encrypted packets, as algorithms considered do not rely on payload content.

Each flow is associated with a particular service. In order to train and evaluate the models, we need to assign a ground-truth service to each flow. This assignment is not initially available and has been made possible by applying the nDPI tool [25] to the packets exchanged during the flow lifetime. nDPI applies a DPI technique to perform service detection. DPI provides the best available classification results by inspecting both the header and payload of the packet. With this in mind, we assume the output of a DPI tool as our best approximation to the ground-truth service. nDPI handles encrypted traffic and it is the most accurate open source DPI application [26]. The flows which nDPI was not able to label were discarded. For this work, we have considered UDP and TCP flows.

Each flow is formed by a sequence of up to 20 packets. For each packet, we have extracted the following six features: source port, destination port, the number of bytes in packet payload, TCP window size, interarrival time and direction of the packet. The TCP window size (TCP flow control) is set to zero for UDP packets. The packet address may have a value of 0-1 indicating whether the packet goes from source to destination or in the opposite direction.

We have considered only the first 20 packets exchanged in a flow lifetime. In the case of flows with more than 20 packets, we have discarded any packet after packet number 20. As we will see, 20 packets are more than enough to obtain a good detection rate, and even a much smaller number still provides excellent performance.

Finally, from these flows, we have built our dataset. Therefore, the dataset consists of 266,160 flows, each flow containing a sequence of 20 vectors, and each vector is made up of 6 features (the six features extracted from the packets' header). The final result is a time-series of feature vectors associated with each flow.

To evaluate the models, we set apart a 15% of flows as a validation set. All the performance metrics given in this paper correspond to this validation set. In order to build the validation set, we sampled the original flows, keeping the same labels frequency between the validation set and the remaining flows (training set).

Fig. 2 presents the final arrangement of a network flow inside the dataset.

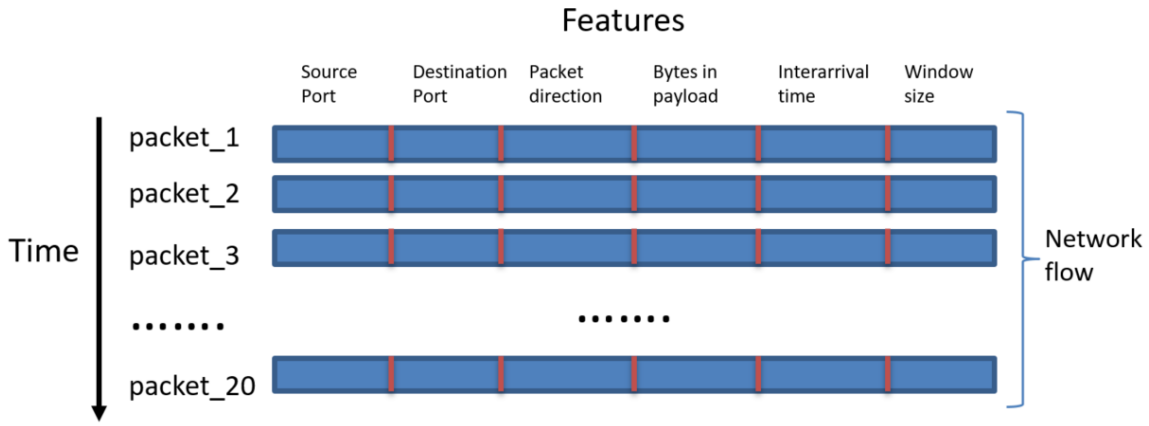


Fig. 2. The composition of a network flow.

B. Models description

Different deep learning models have been studied. The model with best detection

performance has been a combination of CNN [27] and RNN [28]. In this section, we will show all the models considered for this work.

The first model analyzed (Fig. 3) was a simple RNN. In particular, we used a variant of an RNN called LSTM [29], which is easier to train (it solves the vanishing gradient problem). An LSTM is trained with a matrix of values with two dimensions: the temporal dimension and a vector of features. LSTM iterates a neural network (cell) with the time sequential feature vectors and two additional vectors associated with its internal hidden and cell states. The final hidden state of the cell corresponds to the output value. Therefore, the output dimension of an LSTM layer is the same as the size of its internal hidden state (LSTM units). In the model of Fig. 3 we add at the end several fully connected layers. Two layers are fully connected when each node of the previous layer is fully forward connected to every node of the consecutive layer. The fully connected layers have been added to all models.

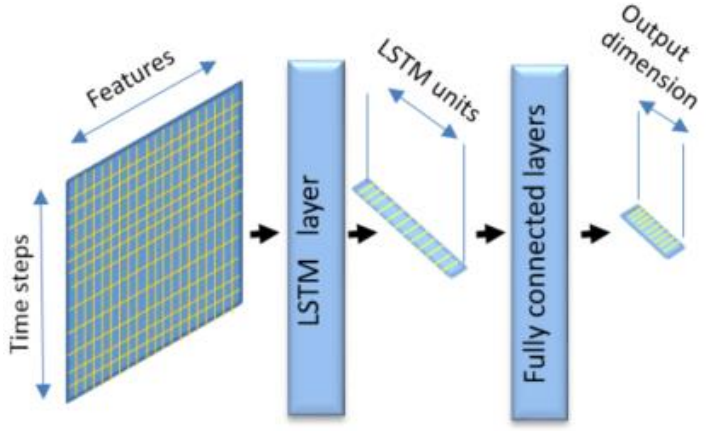


Fig. 3. Deep learning RNN model

In Fig. 4 a pure CNN network is shown. CNNs were initially applied to image processing, as a biologically inspired model to perform image classification, where feature engineering was done automatically by the network thanks to the action of a kernel (filter) which extracts location invariant patterns from the image. Chaining several CNNs allows extracting complex features automatically.

In our case, we have used this image-processing metaphor to apply the technique to a very different dataset. In order to do that, we consider the matrix formed by the time-series of feature vectors as an image. Image pixels are locally correlated; similarly, feature vectors associated with consecutive time slots present a correlated local behavior, which allows us to adopt this analogy.

Each CNN layer generates a multidimensional array (tensor) where the dimensions of the image get reduced but, at the same time, a new dimension is generated, having this new dimension a size equal to the number of filters applied to the image. Consecutive CNN layers will further decrease the image dimensions and increase the new generated dimension size. To top off the model it is necessary to transform the tensor to a vector that can be the input to the final fully connected layers. To accomplish this transformation a simple tensor flattening can be done (Fig. 4).

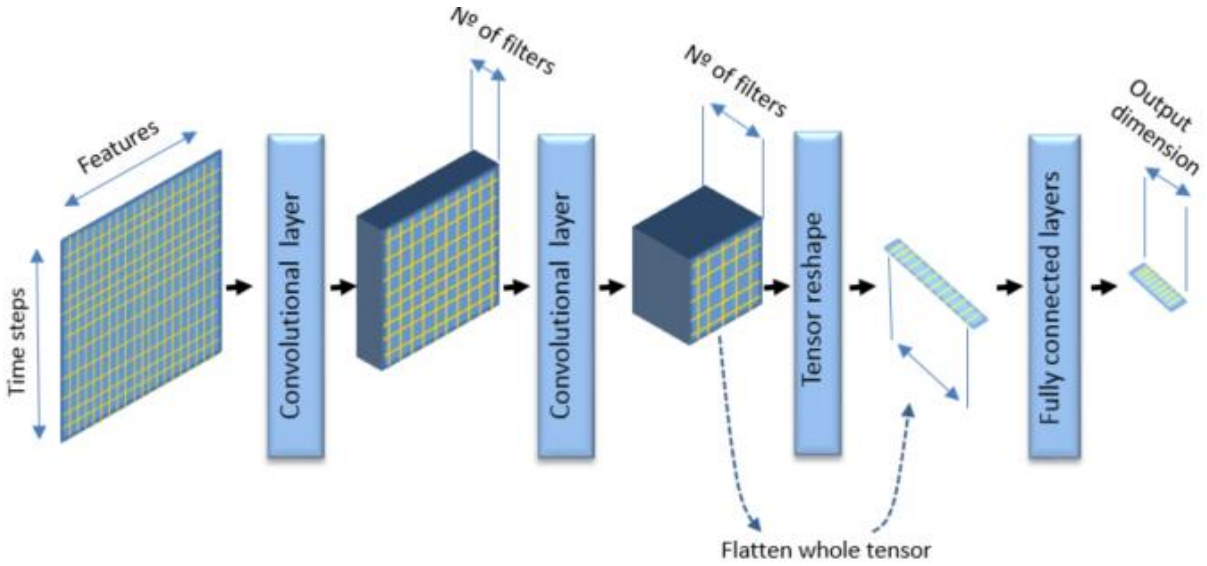


Fig. 4. Deep learning CNN model

The previous models can be combined in a single model as presented in Fig. 5. In this combined model, the final tensor of several chained CNNs is reshaped into a matrix that can act as the input to an RNN (LSTM network). To reshape the tensor as a matrix we keep the dimension associated with the filter's action unchanged, performing a flattening on the other two dimensions, to finally reach a matrix shape. The values produced by the filters of the last CNN will be the equivalent of feature vectors, and the flattened vector produced by the reshaping operation will act as the time dimension needed by the LSTM layer.

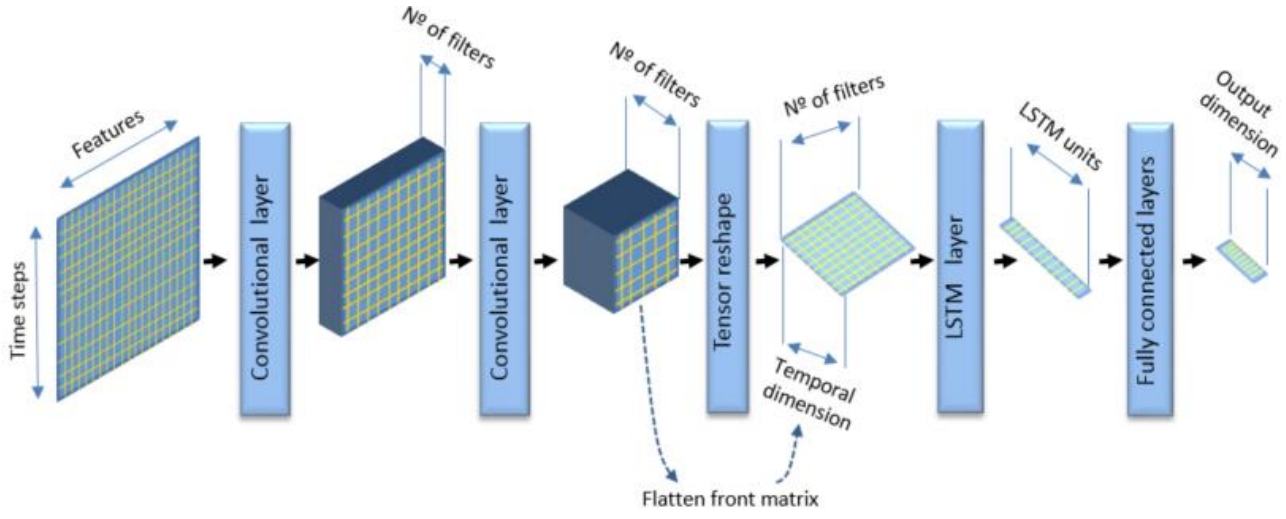


Fig. 5. Combination of CNN and single-layer RNN

Finally, the model introduced in Fig. 6 is similar to the previous model with the inclusion of an additional LSTM layer. When several LSTM layers are concatenated, the LSTM behavior is

different to the one explained previously (Fig. 3). In this case, all LSTM layers (except the last one) adopt a 'return-sequences' mode that produces a sequence of vectors corresponding to the successive iteration of the recurrent network. This sequence of vectors can be grouped in a time sequence, forming the entry point to the next LSTM layer. It is important to note that, for successive LSTM layers, the temporal-dimension of data input does not change (Fig. 6), but the vector-dimension of the successive inputs does.

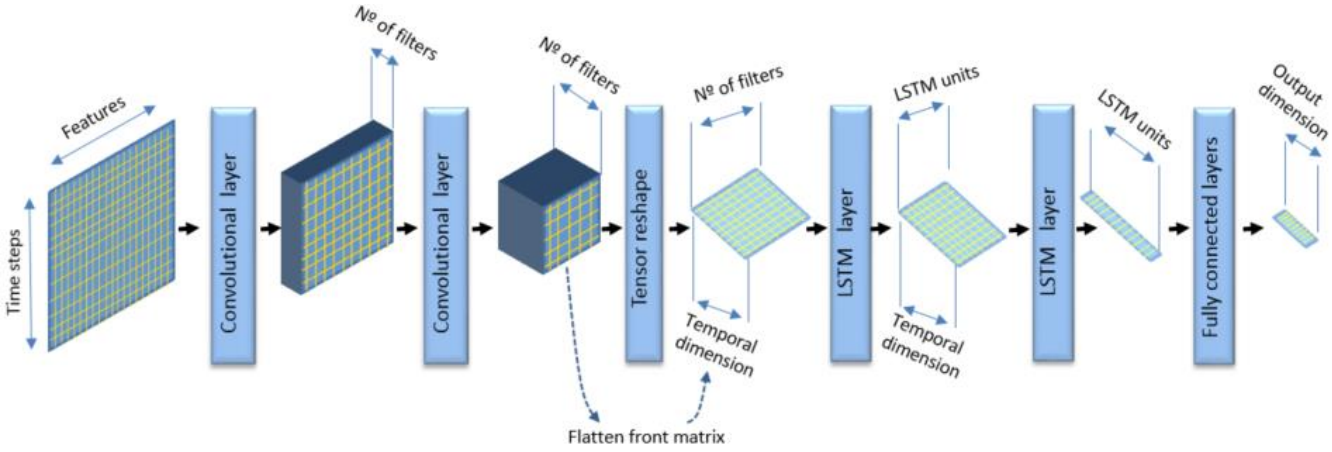


Fig. 6. Combination of CNN and two-layer RNN

Additionally, to the different types of layers presented previously, we have made use of some additional layers: batch normalization, max pooling, and dropout layers.

A dropout layer [30] provides regularization (a generalization of results for unseen data) by dropping out (setting to zero) a percentage of outputs from the previous layer. This apparently nonsensical action forces the network to not over-rely on any particular input, fighting over-fitting and improving generalization.

Max pooling [31] is a kind of convolutional layer. The difference is the filter used. In max pooling, it is used a max-filter, that selects the maximum value of the image region to which the filter is applied. It reduces the spatial size of the output, decreasing the number of features and the computational complexity of the network. The result is a down-sampled output. Similar to a dropout layer, a max pooling layer provides regularization.

Batch normalization [32] makes training convergence faster and can improve performance results. It is done by normalizing, at training time, every feature at batch level (scaling inputs to zero mean and unit variance) and re-scaling again later considering the whole training dataset. The newly learned mean and variance replace the ones obtained at batch-level.

IV. RESULTS

This section presents the results obtained when applying several deep learning models to NTC. The influence of several important hyper-parameters and design decisions is analyzed, in particular: the model architecture, the features selected and the number of packets extracted from the network flows.

In order to appreciate the detection quality of the different options, and considering the

highly unbalanced distribution of service labels, we provide the following performance metrics for each option: accuracy, precision, recall, and F1. Considering all metrics, F1 can be considered the most important metric in this scenario. F1 is the harmonic mean of precision and recall and provides a better indication of detection performance for unbalanced datasets. F1 gets its best value at 1 and worst at 0.

We base our definition of accuracy, F1, precision, and recall in the following four previous definitions: (1) false positive (FP) that happens when there is actually no detection but we conclude there is one; (2) false negative (FN) when we indicate no detection but there is one; (3) true positive (TP) when we indicate a detection and it is real and (4) true negative (TN) when we indicate there is no detection and we are correct. Considering these previous definitions:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (1)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

$$\text{F1} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

We have used Tensorflow to implement all the models, and the python package scikit-learn to calculate performance metrics. All computations have been performed in a commercial PC (i7-4720-HQ, 16GB RAM).

A. Impact of network architecture

We have tried different deep learning architecture models to see their suitability for the NTC problem. In order to build the different architectures, we have considered different combinations of RNN and CNN: RNN only, CNN only, and various arrangements of a CNN followed by an RNN. In all cases, we have added at the end two additional fully connected layers.

In Table I, we provide a description of the different architectures and in Fig. 7 we present their performance metrics. From Fig. 7, we can see that the model CNN+RNN-2a gives the best results for both accuracy and F1.

The architecture description provided in Table I is as follows: Conv(z,x,y,n,m) stands for a convolutional layer with z filters where x and y are the width and height of the 2D filter window, with a stride of n and SAME padding if m is equal to S or VALID padding if m is equal to V (VALID implies no padding and SAME implies padding that preserves output dimensions). MaxPool(x,y,n,m) stands for a Max Pooling layer where x and y are the pool sizes, with a stride of n and SAME padding if m is equal to S or VALID padding if m is equal to V (VALID implies no padding and SAME implies padding that preserves output dimensions). BN stands for a batch normalization layer. FC(x) stands for a fully connected layer with x nodes. LSTM(x) stands for an LSTM layer where x is the dimensionality of the output space; in the case of several LSTM in sequence, each LSTM, except the last one, will return the successive recurrent values which will be the entry values to the following LSTM. DR(x) stands for a dropout layer with a dropout coefficient equal to x.

In all cases, the training was done with a number of epochs between 60-90 epochs, with early stopping if the last 10 epochs did not improve the loss function. We consider an epoch as

a single pass of the complete training dataset through the training process.

All the activation functions were Rectified Linear Units (ReLU) with the exception of the last layer with Softmax activation. The loss function was Softmax Cross Entropy and the optimization was done with batch Stochastic Gradient Descent (SGD) with Adam.

In Table I, an added suffix 'a' to a model name, implies that the model has only changed the dropout percentage at the dropout layers.

Model	Architecture details
RNN-1	LSTM(100)-FC(100)-FC(108)
CNN-1	Conv(32,4,2,1,V)-MaxPool(3,2,1,V)-BN-Conv(64,4,2,1,V)-MaxPool(3,2,1,V)-BN-FC(200)-FC(108)
CNN+RNN-1	Conv(32,4,2,1,V)-MaxPool(3,2,1,V)-BN-Conv(64,4,2,1,V)-MaxPool(3,2,1,V)-BN-LSTM(100)-FC(100)-FC(108)
CNN+RNN-2	Conv(32,4,2,1,V)-BN-Conv(64,4,2,1,V)-BN-LSTM(100)-FC(100)-FC(108)
CNN+RNN-2a	Conv(32,4,2,1,V)-BN-Conv(64,4,2,1,V)-BN-LSTM(100)-DR(0.2)-FC(100)-DR(0.4)-FC(108)
CNN+RNN-3	Conv(16,4,2,1,V)-BN-Conv(32,4,2,1,V)-BN-Conv(64,4,2,1,V)-BN-Conv(128,4,2,1,V)-BN-LSTM(100)-DR(0.1)-LSTM(200)-DR(0.3)-FC(200)-DR(0.5)-FC(108)
CNN+RNN-3a	Conv(16,4,2,1,V)-BN-Conv(32,4,2,1,V)-BN-Conv(64,4,2,1,V)-BN-Conv(128,4,2,1,V)-BN-LSTM(100)-DR(0.2)-LSTM(200)-DR(0.4)-FC(200)-DR(0.5)-FC(108)
CNN+RNN-4	Conv(32,3,2,1,S)-BN-Conv(32,3,2,1,S)-MaxPool(2,2,2,S)-BN-Conv(64,3,2,1,S)-BN-Conv(64,3,2,1,S)-MaxPool(2,2,2,S)-BN-Conv(128,3,2,1,S)-Conv(64,1,1,1,S)-LSTM(100)-DR(0.1)-LSTM(200)-DR(0.3)-FC(200)-DR(0.5)-FC(108)
CNN+RNN-5	Conv(32,3,2,1,S)-BN-Conv(32,3,2,1,S)-MaxPool(2,2,2,S)-BN-Conv(64,3,2,1,S)-BN-Conv(64,3,2,1,S)-MaxPool(2,2,2,S)-BN-Conv(128,3,2,1,S)-Conv(64,1,1,1,S)-LSTM(100)-DR(0.1)-LSTM(150)-DR(0.3)-FC(150)-DR(0.5)-FC(108)

Table I. Details of deep learning network models applied to NTC problem

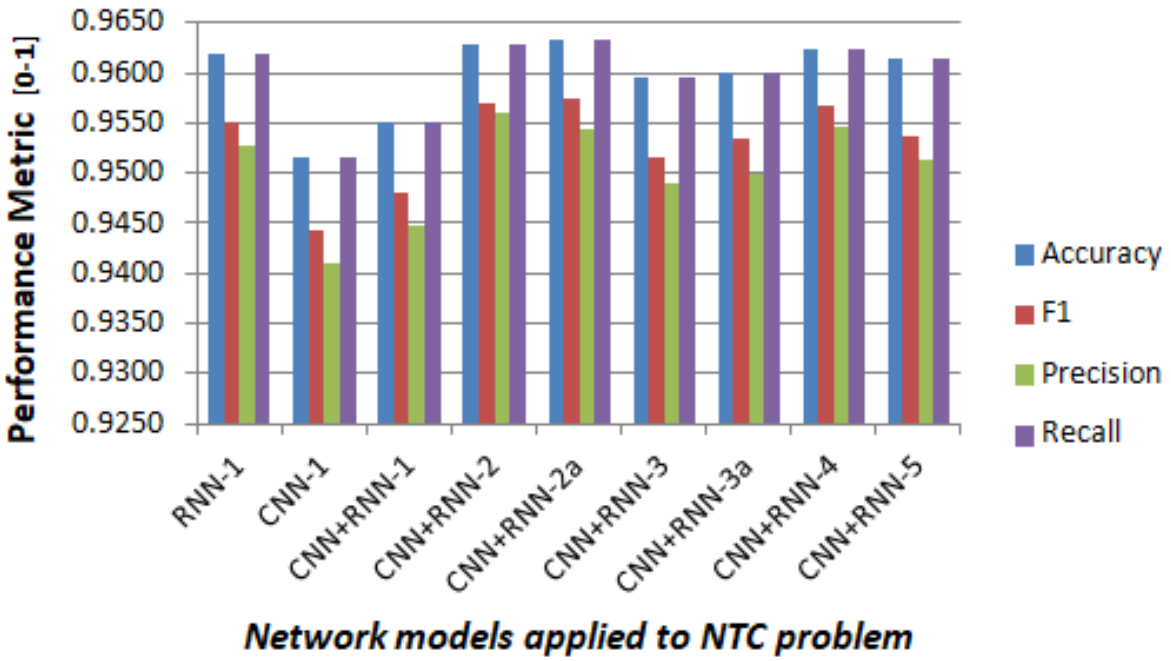


Fig. 7. Classification performance metrics (aggregated) vs. network models

The best model attains an accuracy of 0.9632, an F1 score of 0.9574, a precision of 0.9543 and a recall of 0.9632 (model CNN+RNN-2a).

Analyzing the results, we can see that a simple model, of two CNN layers followed by one

LSTM layer with two fully connected layers at the end, provides best detection results, both in terms of accuracy and F1. The inclusion of MaxPooling or additional CNN or LSTM layers does not improve results. Batch normalization between CNN layers and the inclusion of some dropout layers at the end of the network do improve results.

For this problem, we have 108 distinct service labels to be detected. This is a multi-class classification problem. There are two possible ways to give results in this case: aggregated and One-vs.-Rest results.

For One-vs.-Rest, we focus in a particular class (label) and consider the other classes as a single alternative class, simplifying the problem to a binary classification task for each particular class (one by one). In the case of aggregated results, we try to give a summary result for all classes. There are different alternatives to perform the aggregation (micro, macro, samples, weighted), varying in the way the averaging process is done.

The performance metrics in Fig. 7 are aggregated metrics using a weighted average. We have used the weighted average provided by scikit-learn [33], to calculate the aggregated F1, precision, and recall scores.

In Fig. 8 we provide the One-vs.-Rest metrics for the classification of the first 15 more frequent labels (results obtained with model CNN+RNN-2a). An important observation in Fig. 8 is that for all labels with a frequency higher than 1% (Fig. 1) we achieve accuracy always higher than 98%, and many cases higher than 99%, and an F1 score higher than 0.96. The macro averaged accuracy for these 15 labels is 99.59% (best value in literature).

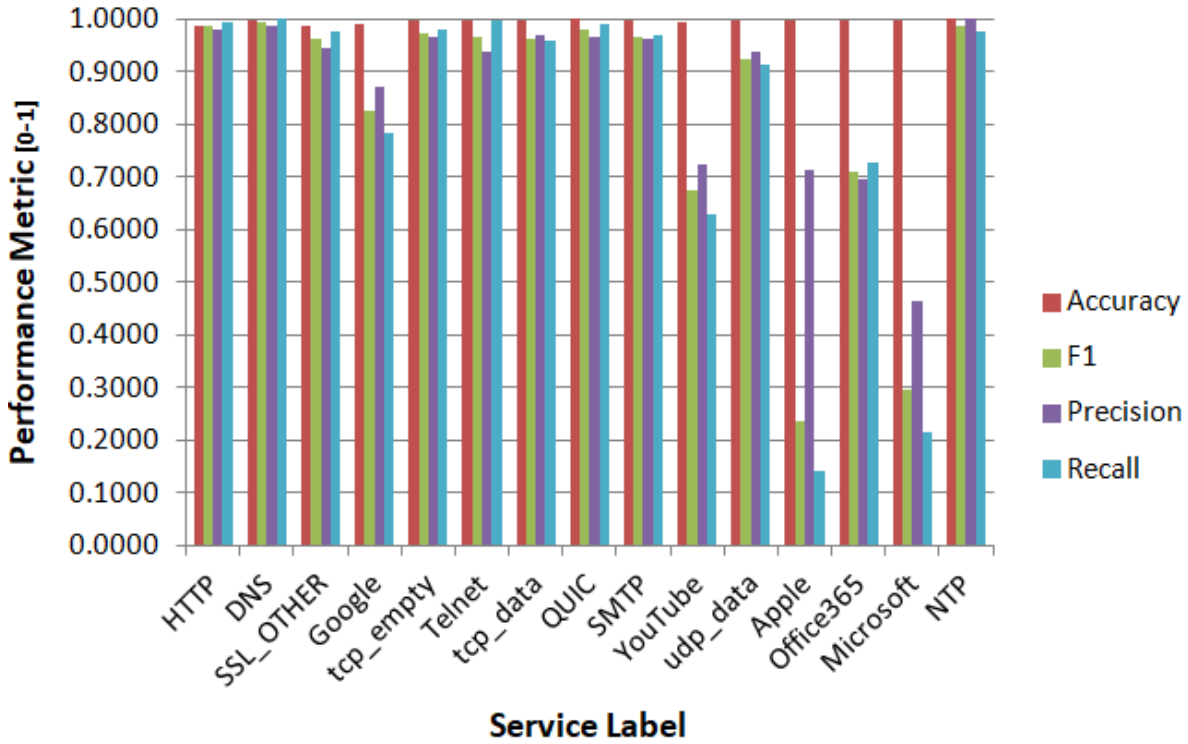


Fig. 8. Performance metrics (one vs. rest) for the classification of several service labels (15 more frequent)

B. Impact of features

In Table II we can see the influence of the features employed in the learning process. Fig. 2

gives the full set of possible features, but it is important to appreciate which features have a higher importance in the detection process.

Table II shows the importance of features by analyzing the detection metrics as we remove different features. The first column in Table II gives the features that are used to train the model (grouped in feature sets) and the right columns the usual aggregate performance metrics for the detection process. The chart in Fig. 9 presents the same results in a different format, to make it easier to compare different feature sets.

As expected, in general, the more features render better results. But, interestingly the packets inter-arrival time (TIMESTAMP) gives slightly worse results when added to the full features set. It seems it provides some not well-aligned information with the source and destination ports, because, as soon as we take away the source and destination port, it is clear it becomes again an important feature. It is also interesting to appreciate the importance of the feature TCP window size (WIN SIZE), being more important than TIMESTAMP when operating with a reduced set of features.

Table II provides metric values with a color code to make it easier to rank the results. In this color code, darker green colors mean better results whereas darker red colors mean worse results.

Features	Accuracy	F1	Precision	Recall
SRC_PORT, DST_PORT, DIR, PAYLOAD, WIN_SIZE, TIMESTAMP	0.9612	0.9553	0.9527	0.9612
SRC_PORT, DST_PORT, DIR, PAYLOAD, WIN_SIZE (Features Set 1)	0.9632	0.9574	0.9543	0.9632
DIR, PAYLOAD , TIMESTAMP , WIN_,SIZE (Features Set 2)	0.8388	0.8170	0.8279	0.8388
DIR, PAYLOAD ,WIN_SIZE (Features Set 3)	0.8202	0.7943	0.7909	0.8202
DIR, PAYLOAD , TIMESTAMP (Features Set 4)	0.7855	0.7500	0.7433	0.7855

Table II. Classification performance metrics (aggregated) vs. features employed (model CNN+RNN-2a)(Table)

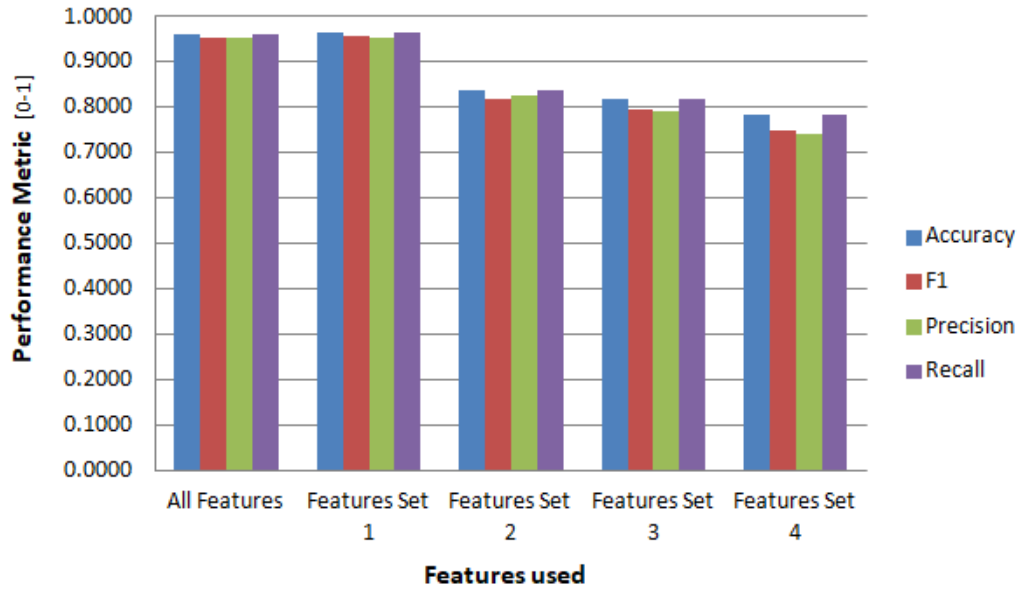


Fig. 9. Classification performance metrics (aggregated) vs. features employed (model CNN+RNN-2a)(Chart)

Model CNN+RNN-2a was used to obtain the results presented in Table II, but the same relationship between results and feature sets was maintained when repeating this same study

with the other models.

C. *Impact of time-series length*

An important parameter to be studied is the influence of the number of packets to be considered when we analyze the network flows. There are flows with hundreds of packets whereas others have only a single packet.

An important doubt at the beginning of the study was the possible influence of this parameter, since increasing the number of included packets could improve detection but at the cost of much higher computing time and resources. As a balanced decision, we opted for a maximum of 20 packets. We have considered only the first 20 packets exchanged in a flow lifetime. In the case of flows with more than 20 packets, we have disregarded any packet after packet number 20. Flows with less than 20 packets were padded with zeros.

Then it was important to know the impact of the number of packets in the overall detection problem and to confirm whether 20 packets was a sensible number. To this aim, we analyzed the performance considering a different number of packets and different architectures. We present here the results for two representative architectures (RNN-1 and CNN+RNN-2a)

We can see the results for the RNN-1 architecture in Fig. 10, and it is important to note that overall detection quality is not significantly changed by using fewer packets until the number of packets is less than five per flow. Therefore, it is enough to consider the very first packets of a flow to have most of the information that allows us to infer their service. In Fig. 10 we can easily appreciate that the detection quality starts to degrade when the number of packets per flow is less than five.

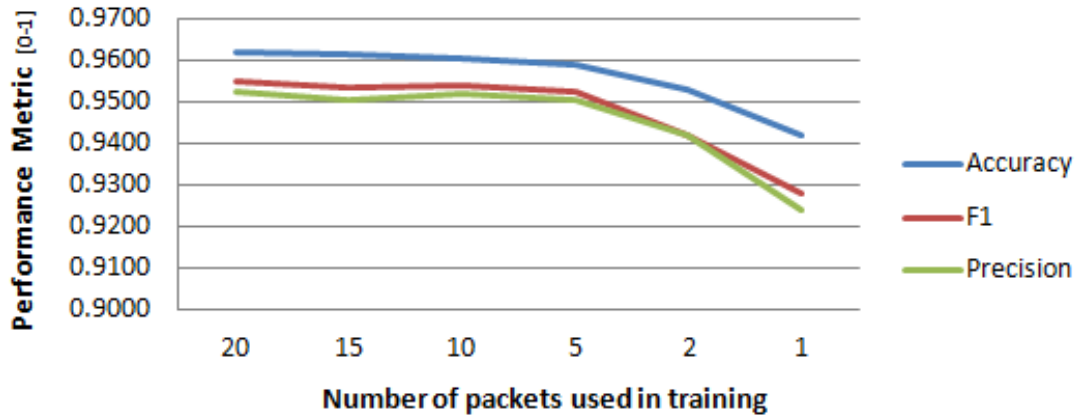


Fig. 10. Classification performance metrics (aggregated) vs. time-series length for architecture RNN-1

Fig. 11 shows the results on the impact of the number of packets for the architecture CNN-RNN-2a. This model supports a smaller reduction in the number of packets (the model needs a minimum length of 7 in the temporal-dimension), but we can still appreciate that regardless of an initial performance decrease, this reduction is not monotonic with the decrease in the number of packets, in fact, it keeps approximately constant for packets lengths in the interval from 7 to 15 (disregarding some intermediate noisy values).

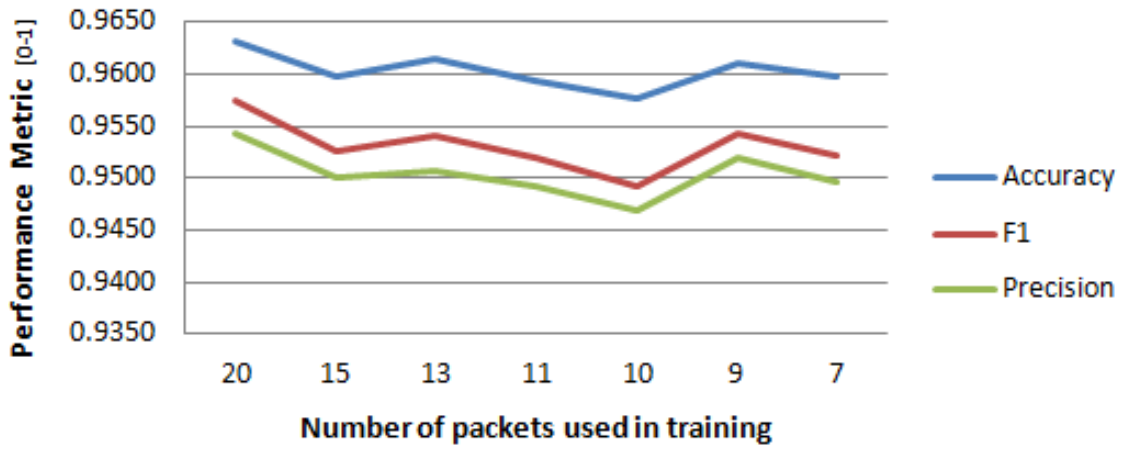


Fig. 11. Classification performance metrics (aggregated) vs. time-series length for architecture CNN+RNN-2a

Therefore, it seems clear that although a minimum number of packets is important, a large number of packets (that is architecture-dependent) is not necessary. In general, between 5 and 15 packets are enough to achieve excellent detection results.

V. CONCLUSION

This work is a contribution to improve the available alternatives and capabilities of NTC in current network monitoring systems; being specially targeted to IoT networks, where traffic classification is highly required [1, 2].

As far as we know, there is no previous application of the RNN and CNN deep learning models to an NTC problem. Therefore, the work presented in this paper is original in essence.

This work provides a thorough analysis of the possibilities provided by deep learning models to NTC. It shows the performance of RNN and CNN models and a combination of them. It demonstrates that a CNN can be successfully applied to NTC classification, giving an easy way to extend the image-processing paradigm of CNN to a vector time-series data (in a similar way to previous extensions to text and audio processing [34, 35]).

A model based on a particular combination of CNN plus RNN gives the best detection results, being these results better than other published works with alternative techniques.

The impact of selected features is demonstrated, and also that it is not necessary to process a large number of packets per flow to have excellent results: any number of packets higher than 5-15 (a number which is architecture-dependent) gives similar results.

The proposed method is robust and gives excellent F1 detection scores under a highly unbalanced dataset with over 100 different classification labels. It works with a very small number of features and does not require feature engineering.

To train the models we have made use of high-level header-based data extracted from the packets. It is not required to rely on IP addresses or payload data, which are probably confidential or encrypted.

A simple RNN model provides already very good results, but it is interesting to appreciate that these results improve when the RNN model is combined with a previous CNN model.

Being it possible to improve results with the inclusion of a CNN shows how the initial intuition that allowed us to assimilate the vector time-series extracted from network packets'

features as an image is correct, and therefore CNNs are valid candidates for dealing with vector time-series of similar nature

Being the deep learning architectures such a fruitful source of new models, we consider, as future work, to experiment with new applications and variants of the CNN and LSTM models.

This work can be especially applicable for new IoT networks in which NTC can be used to differentiate or segregate different classes of traffic, e.g. device identification [36], target detection in Wireless Sensor Networks (WSN) [37] or user priority based [38].

REFERENCES

1. B. Ng, M. Hayes and W. K. G. Seah, "Developing a traffic classification platform for enterprise networks with SDN: Experiences & lessons learned," 2015 IFIP Networking Conference (IFIP Networking), Toulouse, 2015, pp. 1-9.
2. A. Sivanathan, D. Sherrat, H. Habibi Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman , "Characterizing and Classifying IoT Traffic in Smart Cities and Campuses", IEEE INFOCOM Workshop on SmartCity: Smart Cities and Urban Computing, USA, May 2017.
3. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," IEEE Commun. Surv. Tutorials, vol. 10, no. 4, pp. 56–76, 2008.
4. J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust Network Traffic Classification," IEEE/ACM Trans. Netw., vol. 23, no. 4, pp. 1257–1270, 2015.
5. T. Auld, A. W. Moore, and S. F. Gull, "Bayesian Neural Networks for Internet Traffic Classification," IEEE Trans. Neural Networks, vol. 18, no. 1, pp. 223–239, Jan. 2007.
6. X. Xie, B. Yang, Y. Chen, L. Wang and Z. Chen, "Network Traffic Classification Based on Error-Correcting Output Codes and NN Ensemble," 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery, Tianjin, 2009, pp. 475-479.
7. Chen, Z., Wang, H., Abraham, A., Grosan, C., Yang, B., Chen, Y., Wang, L., "Improving Neural Network Classification Using Further Division of Recognition Space". International Journal of Innovative, Computing, Information and Control 5(2) (2009)
8. Herrero, A., Corchado, E., Gastaldo, P., Zunino, R., "Neural projection techniques for the visual inspection of network traffic". Neurocomputing, Volume 72, Issue 16, Pages 3649-3658, 2009.
9. Kothari, A., Keskar, A., "Rough Set Approaches to Unsupervised Neural Network Based Pattern Classifier", Advances in Machine Learning and Data Analysis, Springer Netherlands, Dordrecht, pp. 151-163, 2010.
10. W. Zhou, L. Dong, L. Bic, M. Zhou and L. Chen, "Internet traffic classification using feed-forward neural network," 2011 International Conference on Computational Problem-Solving (ICCP), Chengdu, 2011, pp. 641-646.
11. A Moore D Zuev L. Crogan "Discriminators for use in flow-based classification", Technical Report RR-05-13. Department of Computer Science Queen's Mary University, 2005.
12. Bereket Mathewos, Marco Carvalho, and Fredric Ham. 2011. "Network traffic classification using a parallel neural network classifier architecture". In Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research (CSIIRW '11), Frederick T. Sheldon, Robert Abercrombie, and Axel Krings (Eds.). ACM, New York, NY, USA, , Article 33 , 1 pages.
13. Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., Lee, K.: "Internet traffic classification demystified: myths, caveats, and the best practices", In *Proceedings of the 2008 ACM CoNEXT Conference* (CoNEXT '08). ACM, New York, NY, USA, pp. 11:1–11:12, 2008.
14. M. Shafiq, Xiangzhan Yu, A. A. Laghari, Lu Yao, N. K. Karn and F. Abdessamia, "Network Traffic Classification techniques and comparative analysis using Machine

Learning algorithms," 2016 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 2016, pp. 2451-2455.
15. C. Wang, T. Xu and X. Qin, "Network Traffic Classification with Improved Random Forest," 2015 11th International Conference on Computational Intelligence and Security (CIS), Shenzhen, 2015, pp. 78-81. doi: 10.1109/CIS.2015.27
16. Jun Zhang, Chao Chen, Yang Xiang, Wanlei Zhou, and Athanasios V. Vasilakos, "An Effective Network Traffic Classification Method with Unknown Flow Detection", IEEE Transaction on Network and ServiceManagement, Vol 12, Dec 2013
17. Y.-s. Lim, H.-c. Kim, J. Jeong, C.-k. Kim, T. T. Kwon, and Y. Choi, "Internet traffic classification demystified: on the sources of the discriminative power," In Proceedings of the 6th International COnference (Co-NEXT '10). ACM, New York, NY, USA, pp. 9:1–9:12. 2010.
18. J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/realtime traffic classification using semi-supervised learning," Performance Evaluation, vol. 64, no. 9-12, pp. 1194–1213, Oct. 2007.
19. N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," SIGCOMM Comput. Commun. Rev., vol. 36, pp. 5–16, Oct. 2006.
20. M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification," in Proc. 2004 ACM SIGCOMM Conference on Internet Measurement, pp. 135–148.
21. A. W. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," SIGMETRICS Perform. Eval. Rev., vol. 33, pp. 50–60, June 2005.
22. S. Hao, J. Hu, S. Liu, T. Song, J. Guo and S. Liu, "Network traffic classification based on improved DAG-SVM," 2015 International Conference on Communications, Management and Telecommunications (ComManTel), DaNang, 2015, pp.256-26
23. B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz and M. E. Karsligil, "Application identification via network traffic classification," 2017 International Conference on Computing, Networking and Communications (ICNC), Santa Clara, CA, 2017, pp. 843-848.
24. Z. Yuan and C. Wang, "An improved network traffic classification algorithm based on Hadoop decision tree," *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*, Chongqing, 2016, pp. 53-56.
25. doi: 10.1109/ICOACS.2016.7563047
26. L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "nDPI: Open-source high-speed deep packet inspection," in 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), 2014, pp. 617–622.
27. T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," Comput. Networks, vol. 76, pp. 75–89, 2015.
28. Behnke, Sven, "Hierarchical Neural Networks for Image Interpretation", Volume 2766 of Lecture Notes in Computer Science, Springer-Verlag, 2003
29. Zachary C. Lipton, John Berkowitz, Charles Elkan (2015), "A Critical Review of Recurrent Neural Networks for Sequence Learning", arXiv:1506.00019 [cs.LG]
30. Klaus Greff; Rupesh Kumar Srivastava; Jan Koutník; Bas R. Steunebrink; Jürgen Schmidhuber (2015). "LSTM: A Search Space Odyssey". arXiv:1503.04069
31. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. "Dropout: a simple way to prevent neural networks from overfitting". J. Mach. Learn. Res. 15, 1 (January 2014), 1929-1958.
32. Chen-Yu Lee, Patrick W. Gallagher, Zhuowen Tu (2015), "Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree", arXiv:1509.08985 [stat.ML]

33. Sergey Ioffe, Christian Szegedy (2015), “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, arXiv:1502.03167 [cs.LG]
34. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, “Scikit-learn: Machine Learning in Python”, Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
35. T. N. Sainath, O. Vinyals, A. Senior and H. Sak, "Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks," 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, 2015, pp. 4580-4584.
36. Y. Xiao, K. Cho, “Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers”, arXiv:1602.00367v1 [cs.CL] 1 Feb 2016
37. Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici. 2017. “ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis”. In Proceedings of the Symposium on Applied Computing (SAC '17). ACM, New York, NY, USA, 506-509
38. S. Althunibat, A. Antonopoulos, E. Kartsakli, F. Granelli and C. Verikoukis, "Countering Intelligent-Dependent Malicious Nodes in Target Detection Wireless Sensor Networks," in IEEE Sensors Journal, vol. 16, no. 23, pp. 8627-8639, Dec.1, 2016.
39. M. Grajzer, M. Koziuk, P. Szczechowiak and A. Pescape, "A Multi-Classification Approach for the Detection and Identification of eHealth Applications," 2012 21st International Conference on Computer Communications and Networks (ICCCN), Munich, 2012, pp. 1-6.

PAPER 3

Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT

Manuel Lopez-Martin [1], Belen Carro [1,*], Antonio Sanchez-Esguevillas [1] and Jaime Lloret [2]

[1] Dpto. TSyCeIT, ETSIT, Universidad de Valladolid, Paseo de Belén 15, 47011 Valladolid, Spain; manuel.lopezm@uva.es; belcar@tel.uva.es; antoniojavier.sanchez@uva.es
[2] Instituto de Investigación para la Gestión Integrada de Zonas Costeras, Universitat Politècnica de València, Camino Vera s/n, 46022 Valencia, Spain; jlloret@dcom.upv.es
***** Correspondence: belcar@tel.uva.es; Tel.: +34-983-423-980, Fax: +34-983-423-667

Abstract: The purpose of a Network Intrusion Detection System is to detect intrusive, malicious activities or policy violations in a host or host's network. In current networks, such systems are becoming more important as the number and variety of attacks increase along with the volume and sensitiveness of the information exchanged. This is of particular interest to Internet of Things networks, where an intrusion detection system will be critical as its economic importance continues to grow, making it the focus of future intrusion attacks. In this work, we propose a new network intrusion detection method that is appropriate for an Internet of Things network. The proposed method is based on a conditional variational autoencoder with a specific architecture that integrates the intrusion labels inside the decoder layers. The proposed method is less complex than other unsupervised methods based on a variational autoencoder and it provides better classification results than other familiar classifiers. More important, the method can perform feature reconstruction, that is, it is able to recover missing features from incomplete training datasets. We demonstrate that the reconstruction accuracy is very high, even for categorical features with a high number of distinct values. This work is unique in the network intrusion detection field, presenting the first application of a conditional variational autoencoder and providing the first algorithm to perform feature recovery.

Keywords: intrusion detection; variational methods; conditional variational autoencoder; feature recovery; neural networks

---

1. Introduction

A Network Intrusion Detection System (NIDS) is a system which detects intrusive, malicious activities or policy violations in a host or host's network. The importance of NIDS is growing as the heterogeneity, volume and value of network data continue to increase. This is especially important for current Internet of Things (IoT) networks [1], which carry mission-critical data for business services.

Intrusion detection systems can be host-based or network-based. The first monitor and analyze the internals of a computer system while the second deal with attacks on the communication interfaces [2]. For this work, we will focus on network-based systems.

Intruders in a system can be internal or external. Internal intruders have access to the system but their privileges do not correspond to the access made, while the external intruders do not have access to the system. These intruders can perform a great variety of attacks: denial of service, probe, user to root attacks, etc. [2]

NIDS has been a field of active research for many years, being its final goal to have fast and accurate systems able to analyze network traffic and to predict potential threats. It is possible to classify NIDS by detection approach as signature-based detection approaches and anomaly-based detection methods. Signature-based detection methods use a database of previously identified bad patterns to identify and report an attack, while anomaly-based detection uses a model to classify (label) traffic as good or bad, based mainly on supervised or unsupervised machine learning methods. One characteristic of anomaly-based methods is the need to deal with unbalanced data. This happens because intrusions in a system are usually an exception, difficult to separate from the usually more abundant normal traffic. Working with unbalanced data is often a challenge for both the prediction algorithms and performance metrics used to evaluate systems.

There are different ways to set up an intrusion detection model [3], adopting different approaches: probabilistic methods, clustering methods or deviation methods. In probabilistic methods, we characterize the probability distribution of normal data and define as an anomaly any data with a given probability lower than a threshold. In clustering methods, we cluster the data and categorize as an anomaly any data too far away from the desired normal data cluster. In deviation methods, we define a generative model able to reconstruct the normal data, in this setting we consider as an anomaly any data that is reconstructed with an error higher than a threshold.

For this work, we present a new anomaly-based supervised machine learning method. We will use a deviation-based approach, but, instead of designating a threshold to define an intrusion, we will use a discriminative framework that will allow us to classify a particular traffic sample with the intrusion label that achieves less reconstruction error. We call the proposed method Intrusion Detection CVAE (ID-CVAE). The proposed method is based on a conditional variational autoencoder (CVAE) [4,5] where the intrusion labels are included inside the CVAE decoder layers. We use a generative model based on variational autoencoder (VAE) concepts, but relying on two inputs: the intrusion features and the intrusion class labels, instead of using the intrusion features as a single input, as it is done with a VAE. This change provides many advantages to our ID-CVAE when comparing it with a VAE, both in terms of flexibility and performance.

When using a VAE to build a classifier, it is necessary to create as many models as there are distinct label values, each model requiring a specific training step (one vs. rest). Each training step employs, as training data, only the specific samples associated with the label learned, one at a time. Instead, ID-CVAE needs to create a single model with a single training step, employing all training data irrespective of their associated labels. This is why a classifier based on ID-CVAE is a better option in terms of computation time and solution complexity. Furthermore, it provides better classification results than other familiar classifiers (random forest, support vector machines, logistic regression, multilayer perceptron), as we will show in Section 4.1.

ID-CVAE is essentially an unsupervised technique trained in a supervised manner, due to the use of class labels during training. More important than its classification results, the proposed model (ID-CVAE) is able to perform feature reconstruction (data recovery). ID-CVAE will learn the distribution of features values by relying on a mapping to its internal latent variables, from which a later feature recovery can be performed in the case of input samples with incomplete features. In particular, we will show that ID-CVAE is able to recover categorical features with accuracy over 99%. This ability to perform feature recovery can be an important asset in an IoT network. IoT networks may suffer from connection and sensing errors that may render some of the received data invalid [6]. This may be particularly important for categorical features that carry device`s state values. The work presented in this paper allows recovering those missing critical data, as long as we have available some related features, which may be less critical and easier to access (Section 4.2).

This work is unique in the NIDS field, presenting the first application of a conditional VAE and providing the first algorithm to perform feature recovery. The paper is organized as follows: Section 2 presents related works. Section 3 describes the work performed. Section 4 describes the results obtained and, finally, Section 5 provides conclusion and future work.

2. Related Works

As far as we know, there is no previous reported application of a CVAE to perform classification with intrusion detection data, although there are works related with VAE and CVAE in other areas.

An and Cho [7] presented a classifier solution using a VAE in the intrusion detection field, but it is a VAE (not CVAE) with a different architecture to the one presented here. They use the KDD 99 dataset. The authors of [4] apply a CVAE to a semi-supervised image classification problem. In [8] they used a recurrent neural network (RNN) with a CVAE to perform anomaly detection on one Apollo's dataset. It is applied to generic multivariate time-series. The architecture is different to the one presented and the results are not related to NIDS. Similarly [9] employs an RNN with a VAE to perform anomaly detection on multivariate time-series coming from a robot. Data and results are not applicable to NIDS.

There are works that present results applying deep learning models to classification in the intrusion detection field. In [10] a neural network is used for detecting DoS attacks in a simulated IoT network, reporting an accuracy of 99.4%. The work in [11] presents a classifier which detects intrusions in an in-vehicle Controller Area Network (CAN), using a deep neural network pre-trained with a Deep Belief Network (DBN). The authors of [12] use a stacked autoencoder to detect multilabel attacks in an IEEE 802.11 network with an overall accuracy of 98.6%. They use a sequence of sparse auto-encoders but they do not use variational autoencoders. Ma et al. [13] implemented an intrusion classifier combining spectral clustering and deep neural networks in an ensemble algorithm. They used the NSL-KDD dataset in different configurations, reporting an overall accuracy of 72.64% for a similar NSL-KDD configuration to the one presented in this paper.

Using other machine learning techniques, there is also an important body of literature applying classification algorithms to the NSL-KDD dataset. It is important to mention that comparison of results in this field is extremely difficult due to: (1) the great variability of the different available datasets and algorithms applied; (2) the aggregation of classification labels in different sets (e.g., 23 labels can be grouped hierarchically into five or two subsets or categories); (3) diversity of reported performance metrics and (4) reporting results in unclear test datasets. This last point is important to mention, because for example, for the NSL-KDD dataset, 16.6% of samples in the test dataset correspond to labels not present in the training dataset. This is an important property of this dataset and creates an additional difficulty to the classifier. From this, it is clear how the performance of the classification may be different if the prediction is based on a subset of the training or test datasets, rather than the complete set of test data.

The difficulties presented above are shown in detail in [14]. In [15], applying a multilayer perceptron (MLP) with three layers to the NSL-KDD dataset, they achieved an accuracy of 79.9% for test data, for a 5-labels intrusion scenario. For a 2-labels (normal vs. anomaly) scenario they provided an accuracy of 81.2% for test data. In [16] they provided, for a 2-labels scenario and using self-organizing maps (SOM), a recall of 75.49% on NSL-KDD test data. The authors of [17] reported employing AdaBoost with naive Bayes as weak learners, an F1 of 99.3% for a 23-labels scenario and an F1 of 98% for a 5-labels scenario; to obtain these figures they used 62,984 records for training (50% of NSL-KDD), where 53% are normal records and the remaining 47% are distributed among the different attack types; test results are based on 10-fold cross-validation over the training data, not on the test set. Bhuyan et al. [2] explained

the reasons for creating the NSL-KDD dataset. They gave results for several algorithms. The best accuracy reported was 82.02% with naive Bayes tree using Weka. They use the full NSL_KDD dataset for training and testing, for the 2-labels scenario.

ID-CVAE's ability to recover missing features is unique in the literature. There are other applications of generative models to NIDS, but none of them reports capabilities to perform feature recovery. In [18,19], the authors used a generative model—a Hidden Markov Model—to perform classification only. The work in [18] does not report classification metrics and [19] provides a precision of 93.2% using their own dataset. In [20] they resorted to a deep belief network applied to the NSL-KDD dataset to do intrusion detection. They reported a detection accuracy of 97.5% using just 40% of the training data, but it is unclear what test dataset is used. Xu et al. [21] employed continuous time Bayesian networks as detection algorithm, using the 1998 DARPA dataset. They achieved good results on the 2-labels scenario; the metric provided is a ROC curve. Finally, [22] presents a survey of works related to neural networks architectures applied to NIDS, including generative models; but no work on feature recovery is mentioned.

Using a different approach, [6] proposes a method to recover missing (incomplete) data from sensors in IoT networks using data obtained from related sensors. The method used is based on a probabilistic matrix factorization and it is more applicable to the recovery of continuous features. Related to NIDS for IoT, specifically wireless sensor networks, Khan et al. [23] presents a good review of the problem, and [24,25] show details of some of the techniques applied.

3. Work Description

In the following sections, we present the dataset used for this work and a description of the variational Bayesian method that we have employed.

3.1. Selected Dataset

We have used the NSL-KDD dataset as a representative dataset for intrusion detection. The NSL-KDD [14] dataset is a derivation of the original KDD 99 dataset. It solves the problem of redundant samples in KDD 99, being more useful and realistic. NSL-KDD provides a sufficiently large number of samples. The distribution of samples among intrusion classes (labels) is quite unbalanced and provides enough variability between training and test data to challenge any method that tries to reproduce the structure of the data.

The NSL-KDD dataset has 125,973 training samples and 22,544 test samples, with 41 features, being 38 continuous and three categorical (discrete valued) [15]. Six continuous variables were discarded since they contained mostly zeros. We have performed an additional data transformation: scaling all continuous features to the range [0–1] and one-hot encoding all categorical features. This provides a final dataset with 116 features: 32 continuous and 84 with binary values ({0, 1}) associated to the three one-hot encoded categorical features. It is interesting to note that the three categorical features: *protocol*, *flag*, and *service* have respectively three, 11 and 70 distinct values.

The training dataset contains 23 possible labels (normal plus 22 labels associated with different types of intrusion); meanwhile, the test dataset has 38 labels. That means that the test data has anomalies not present at training time. The 23 training and 38 testing labels have 21 labels in common; two labels only appear in training set and 17 labels are unique to the testing data. Up to 16.6% of the samples in the test dataset correspond to labels unique to the test dataset, and which were not present at training time. This difference in label distribution introduces an additional challenge to the classifiers.

As presented in [14], the training/testing labels are associated to one of five possible categories: NORMAL, PROBE, R2L, U2R and DoS. All the above categories correspond to an intrusion except the NORMAL category, which implies that no intrusion is present. We have

considered these five categories as the final labels driving our results. These labels are still useful to fine-grain characterize the intrusions, and are still quite unbalanced (an important characteristic of intrusion data) yet contain a number of samples, in each category, big enough to provide more meaningful results.

We use the full training dataset of 125,973 samples and the full test dataset of 22,544 samples for any result we provide concerning the training and test NSL-KDD datasets. It is also important to mention that we do not use a previously constructed (customized) training or test datasets, nor a subset of them, what may provide better-alleged results but be less objective and also miss the point to have a common reference to compare results.

3.2. Methodology

In Figure 1 we compare ID-CVAE and VAE architectures. In the VAE architecture [26], we try to learn the probability distribution of data: $\boldsymbol{X}$, using two blocks: an encoder and a decoder block. The encoder implements a mapping from $\boldsymbol{X}$ to a set of parameters that completely define an associated set of intermediate probability distributions: $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{X})$. These intermediate distributions are sampled, and the generated samples constitute a set of latent variables: $\boldsymbol{Z}$, which forms the input to the next block: the decoder. The decoder block will operate in a similar way to the encoder, mapping from the latent variables to a new set of parameters defining a new set of associated probability distributions: $\boldsymbol{p}(\widehat{\boldsymbol{X}}/\boldsymbol{Z})$, from which we take samples again. These final samples will be the output of our network: $\widehat{\boldsymbol{X}}$.

The final objective is to approximate as much as possible the input and output of the network: $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$. But, in order to attain that objective, we have to map the internal structure of the data to the probability distributions: $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{X})$ and $\boldsymbol{p}(\widehat{\boldsymbol{X}}/\boldsymbol{Z})$.

The probability distributions $\boldsymbol{p}(\widehat{\boldsymbol{X}}/\boldsymbol{Z})$ and $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{X})$ are conditional probability distributions, as they model the probability of $\widehat{\boldsymbol{X}}$ and $\boldsymbol{Z}$ but depend on their specific inputs: $\boldsymbol{Z}$ and $\boldsymbol{X}$, respectively

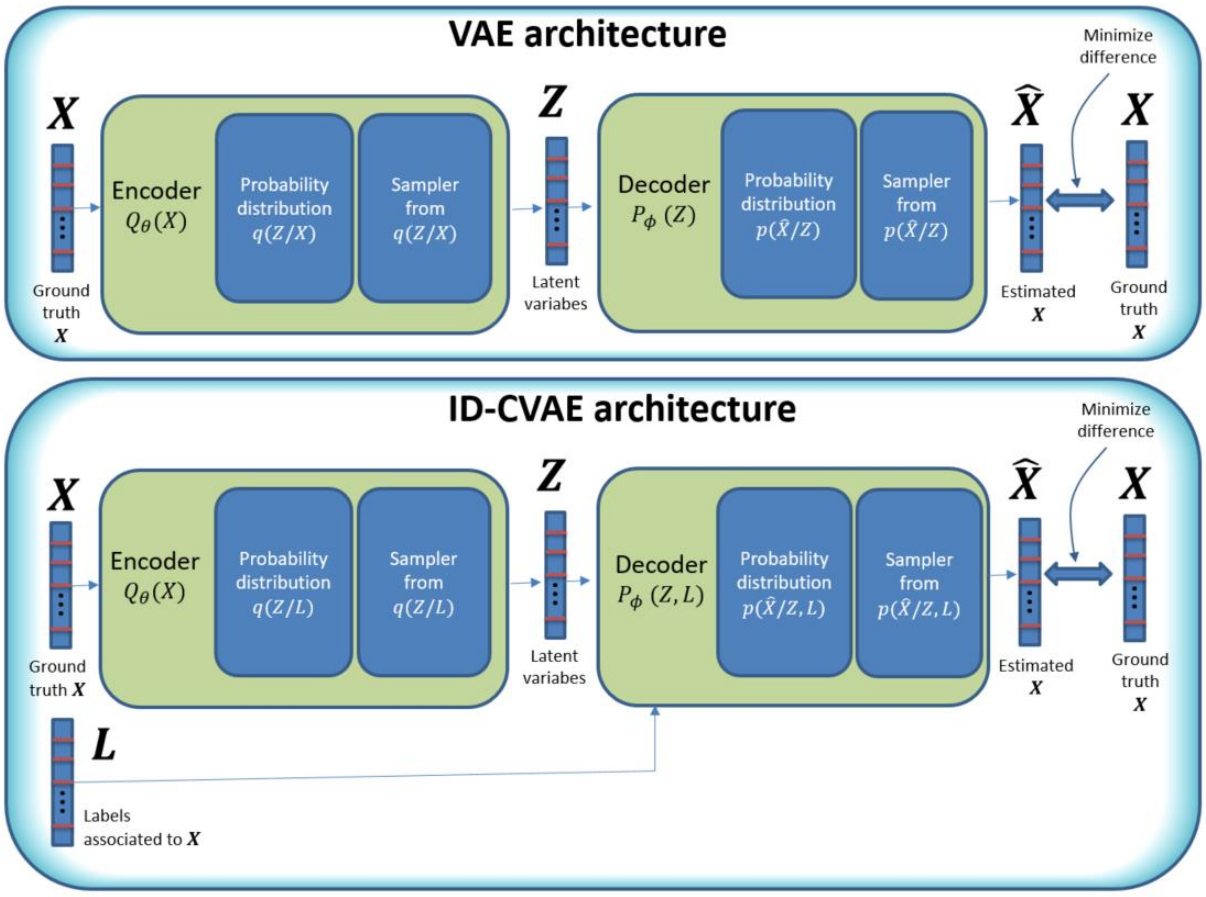


Figure 1. Comparison of ID-CVAE with a typical VAE architecture.

In a VAE, the way we learn the probability distributions: $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{X})$ and $\boldsymbol{p}(\widehat{\boldsymbol{X}}/\boldsymbol{Z})$, is by using a variational approach [26], which translates the learning process to a minimization process, that can be easily formulated in terms of stochastic gradient descent (SGD) in a neural network.

In Figure 1, the model parameters: $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, are used as a brief way to represent the architecture and weights of the neural network used. These parameters are tuned as part of the VAE training process and are considered constant later on.

In the variational approach, we try to maximize the probability of obtaining the desired data as output, by maximizing a quantity known as the Evidence Lower Bound (ELBO) [22]. The ELBO is formed by two parts: (1) a measure of the distance between the probability distribution $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{X})$ and some reference probability distribution of the same nature (actually a prior distribution for $\boldsymbol{Z}$), where the distance usually used is the Kullback-Leibler (KL) divergence; and (2) the log likelihood of $\boldsymbol{p}(\boldsymbol{X})$ under the probability distribution $\boldsymbol{p}(\widehat{\boldsymbol{X}}/\boldsymbol{Z})$, that is the probability to obtain the desired data ($\boldsymbol{X}$) with the final probability distribution that produces $\widehat{\boldsymbol{X}}$.

All learned distributions are parameterized probability distributions, meaning that they are completely defined by a set of parameters (e.g., the mean and variance of a normal distribution). This is very important in the operation of the model, as we rely on these parameters, obtained as network nodes values, to model the associated probability distributions: $\boldsymbol{p}(\widehat{\boldsymbol{X}}/\boldsymbol{Z})$ and $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{X})$.

Based on the VAE model, our proposed method: ID-CVAE, has similarities to a VAE but instead of using exclusively the same data for the input and output of the network, we use additionally the labels of the samples as an extra input to the decoder block (Figure 1, lower diagram). That is, in our case, using the NSL-KDD dataset, which provides samples with 116 features and a class label with five possible values associated with each sample, we will have a vector of features (of length 116) as both input and output, and its associated label (one-hot encoded in a vector of length 5) as an extra input.

To have the labels as an extra input, leads to the decoder probability distributions being conditioned on the latent variable and the labels (instead of exclusively on the latent variable: $\boldsymbol{Z}$), while the encoder block does not change (Figure 1, lower diagram). This apparently small change, of adopting the labels as extra input, turns out to be an important difference, as it allows one to:

- Add extra information into the decoder block which is important to create the required binding between the vector of features and labels.
- Perform classification with a single training step, with all training data.
- Perform feature reconstruction. An ID-CVAE will learn the distribution of features values using a mapping to the latent distributions, from which a later feature recovery can be performed, in the case of incomplete input samples (missing features).

In Figure 2 we present the elements of the loss function to be minimized by SGD for the ID-CVAE model. We can see that, as mentioned before, the loss function is made up of two parts: a KL divergence and a log likelihood part. The second part takes into account how probable is to generate $\boldsymbol{X}$ by using the distribution $\boldsymbol{p}(\widehat{\boldsymbol{X}}/\boldsymbol{Z},\boldsymbol{L})$, that is, it is a distance between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$. The KL divergence part can be understood as a distance between the distribution $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{X})$ and a prior distribution for $\boldsymbol{Z}$. By minimizing this distance, we are really avoiding that $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{X})$ departs too much from its prior, acting finally as a regularization term. The nice feature about this regularization term is that it is automatically adjusted, and it is not necessary to perform cross-validation to adjust a hyper-parameter associated to the regularization, as it is needed in other models (e.g., parameter $\boldsymbol{\lambda}$ in ridge regression)

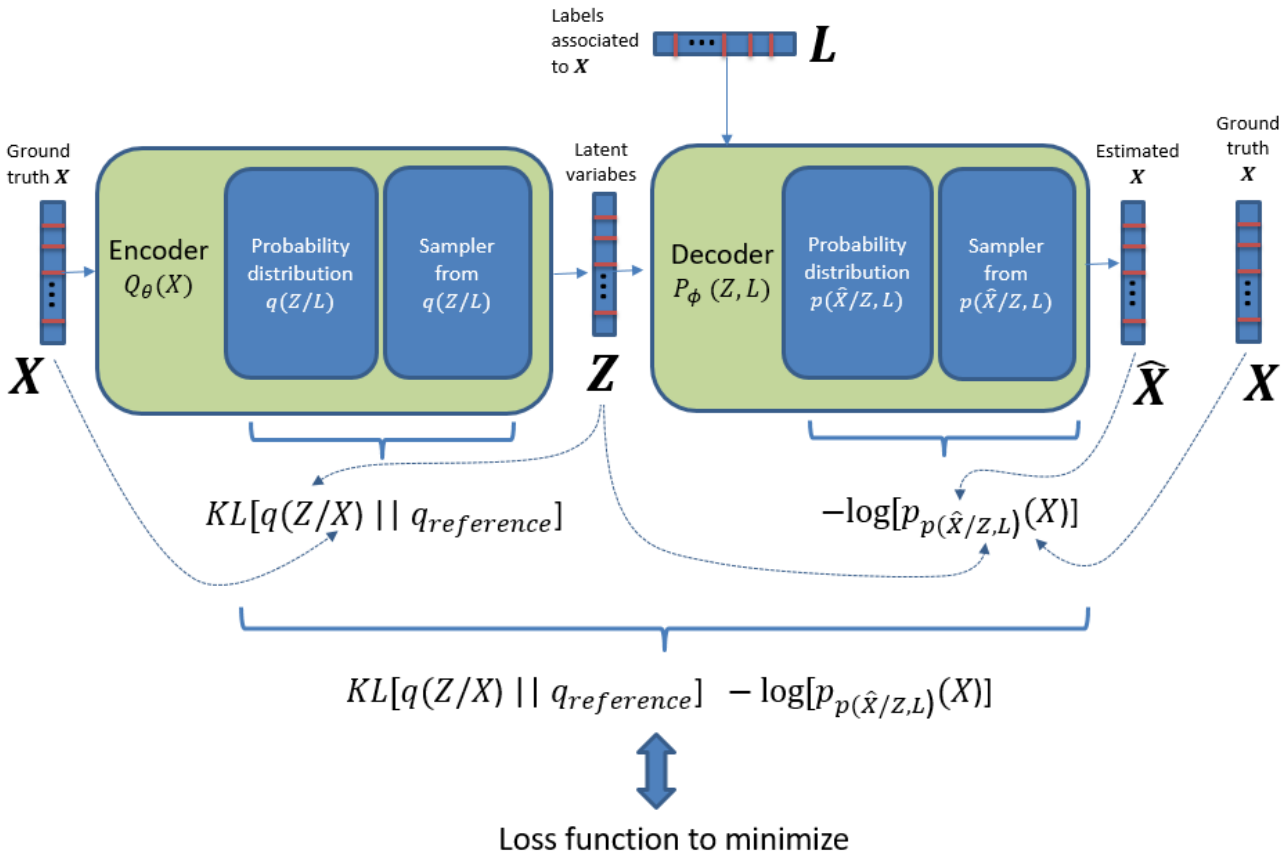


Figure 2. Details on the loss function elements for the ID-CVAE model.

3.3. Model Details

The details of the ID-CVAE model are presented in Figure 3. We employ a multivariate Gaussian as the distribution for $\boldsymbol{q(Z/X)}$, with a mean $\boldsymbol{\mu(X)}$ and a diagonal covariance matrix: $\boldsymbol{\Sigma(X)} \rightarrow \boldsymbol{\sigma_i^2(X)}$, with different values along the diagonal. We have a standard normal $\boldsymbol{N(0,I)}$ as the prior distribution for $\boldsymbol{Z}$.

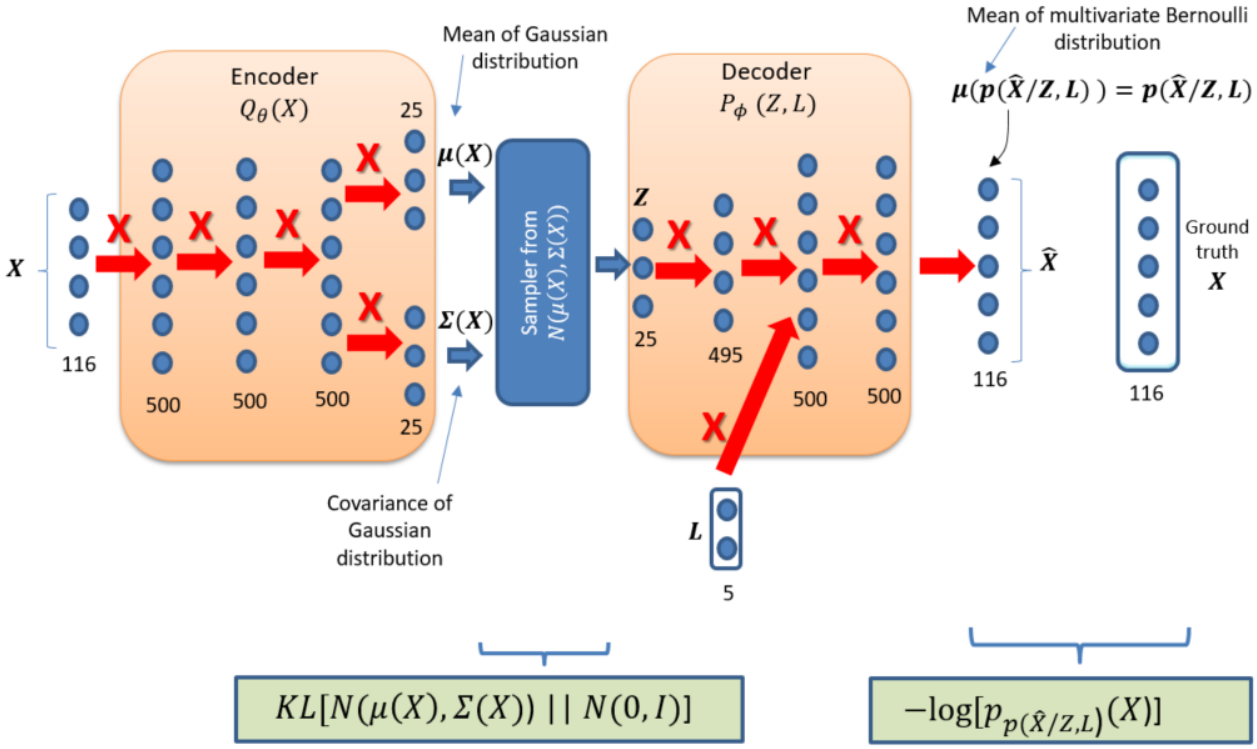


Figure 3. ID-CVAE model details.

For the distribution $\boldsymbol{p}(\widehat{\boldsymbol{X}}/\boldsymbol{Z},\boldsymbol{L})$ we use a multivariate Bernoulli distribution. The Bernoulli distribution has the interesting property of not requiring a final sampling, as the output parameter that characterizes the distribution is the mean that is the same as the probability of success. This probability can be interpreted as a [0–1] scaled value for the ground truth $\boldsymbol{X}$, which has been already scaled to [0–1]. Then, in this case, the output of the last layer is taken as our final output $\widehat{\boldsymbol{X}}$.

The selection of distributions for $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{X})$ and $\mathbf{p}(\widehat{\boldsymbol{X}}/\boldsymbol{Z},\boldsymbol{L})$ is aligned with the ones chosen in [26], they are simple and provide good results. The boxes at the lower part of Figure 3 show the specific choice of the loss function. This is a particular selection for the generic loss function presented in Figure 2.

An important decision is how to incorporate the label vector in the decoder. In our case, to get the label vector inside the decoder we just concatenate it with the values of the second layer of the decoder block (Figure 3). The position for inserting the $\boldsymbol{L}$ labels has been determined by empirical results (see Section 4.1) after considering other alternatives positions.

In Figure 3, a solid arrow with a nearby X designates a fully connected layer. The numbers behind each layer designate the number of nodes of the layer. The activation function of all layers is ReLU except for the activation function of last encoder layer that is Linear and the activation function of last decoder layer which is Sigmoid. The training has been performed without dropout.

4. Results

This section presents the results obtained by applying ID-CVAE and some other machine learning algorithms to the NSL-KDD dataset. A detailed evaluation of results is provided.

In order to appreciate the prediction performance of the different options, and considering the highly unbalanced distribution of labels, we provide the following performance metrics: accuracy, precision, recall, F1, false positive rate (FPR) and negative predictive value (NPV). We base our definition of these performance metrics on the usually accepted ones [2].

Considering all metrics, F1 can be considered the most important metric in this scenario. F1 is the harmonic mean of precision and recall and provides a better indication of prediction performance for unbalanced datasets. F1 gets its best value at 1 and worst at 0.

When doing either classification or feature reconstruction we will face a multi-class classification problem. There are two possible ways to give results in this case: aggregated and One-vs.-Rest results. For One-vs.-Rest, we focus in a particular class (label) and consider the other classes as a single alternative class, simplifying the problem to a binary classification task for each particular class (one by one). In the case of aggregated results, we try to give a summary result for all classes. There are different alternatives to perform the aggregation (micro, macro, samples, weighted), varying in the way the averaging process is done [27]. Considering the results presented in this paper, we have used the weighted average provided by scikit-learn [27], to calculate the aggregated F1, precision and recall scores.

4.1. Classification

We can use ID-CVAE as a classifier. Figure 4 shows the process necessary to perform classification. The process consists of two phases (in order): training and prediction phase.

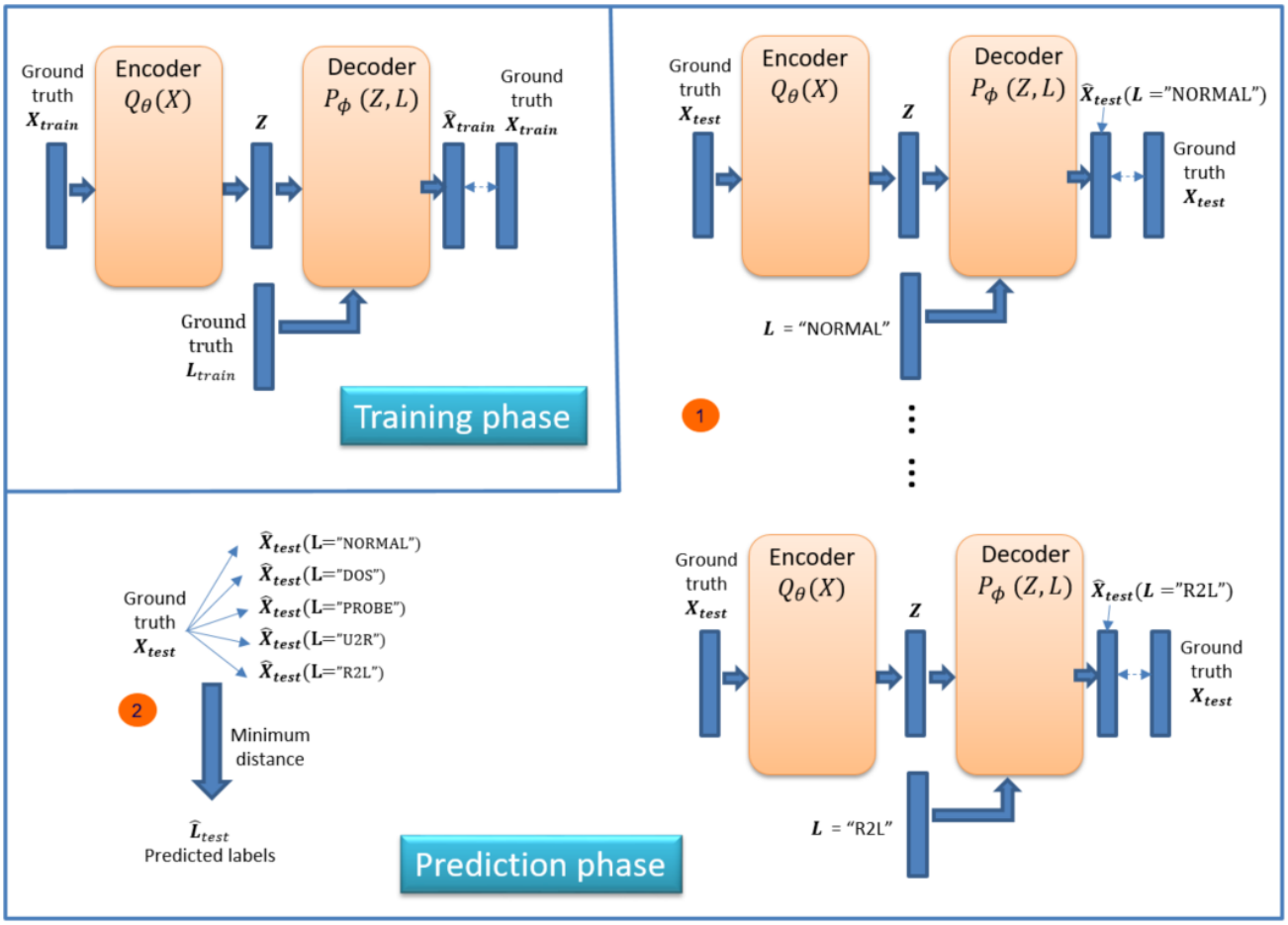


Figure 4. Classification framework.

In the training phase, we train the model using a training dataset together with its associated labels. We train the model as presented in Section 3, trying to minimize the difference between the recovered and ground truth training dataset: $\hat{\boldsymbol{X}}_{train}$ vs. $\boldsymbol{X}_{train}$.

In the prediction phase the objective is to retrieve the predicted labels for a new test dataset: $\boldsymbol{X}_{test}$. The prediction phase is made up of two steps (Figure 4). In the first step we apply the previously trained model to perform a forward pass to obtain a recovered test dataset

($\widehat{\boldsymbol{X}}_{test}$). For this step, we use two inputs: the original test dataset plus a label vector with a single value. That is, we use as label input (L vector) a single label value (e.g., NORMAL, DOS, R2L) for all samples in the test dataset. We need to run this step as many times as there are distinct values in the label, each time changing the value of the L vector. This allows having a set of recovered test datasets, each one using a different label value.

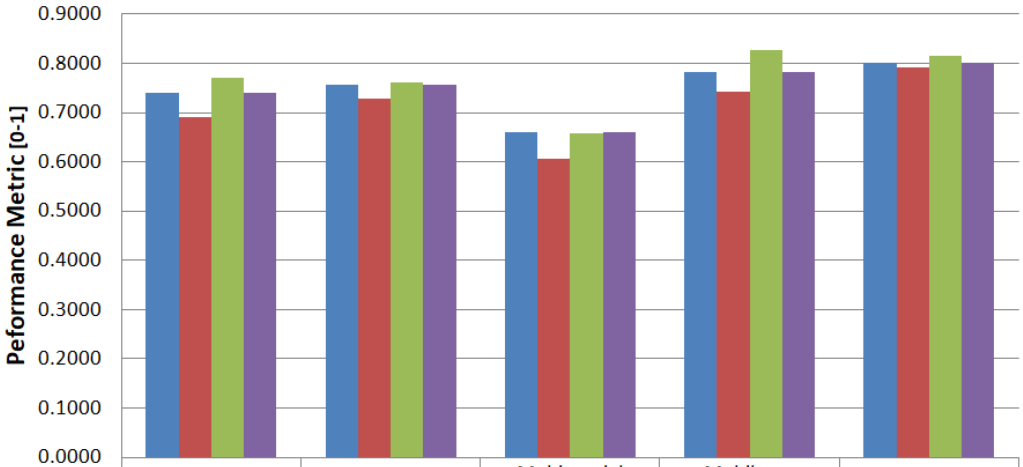
In step two of the prediction phase, we calculate the distance between the ground truth test dataset and each of the recovered ones, choosing for each sample the label associated with the minimum distance. Several distances can be used and we have selected the Euclidean distance.

The intuition behind this process is that the network learns how to recover the original features better when using the correct label as input. Therefore, we choose the label that generates the recovered features closer to the original ones.

For ID-CVAE, the classification process requires a single training stage followed by as many test stages as distinct values we try to predict. The training stage is the one demanding more time and resources, while the test stage is very light and fast. On the contrary, if we use a VAE to perform classification, we will require as many training and test stages as there are distinct label values.

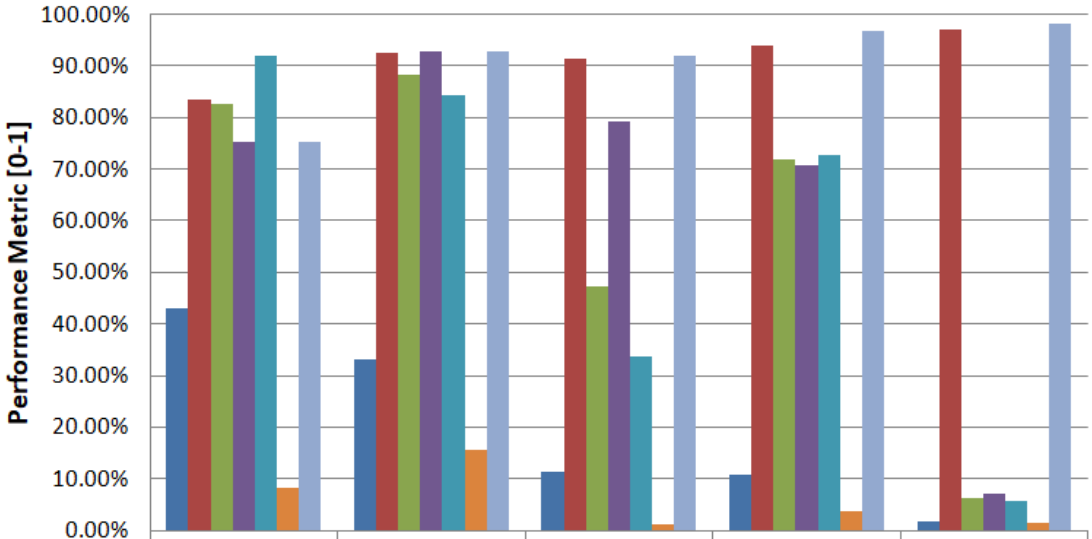
When applying the above-described process to the NSL-KDD test dataset we obtain the classification results presented in Figure 5. In Figure 5 we compare performance metrics for ID-CVAE with results obtained when applying conventional supervised algorithms: random forest, linear support vector machine (SVM), multinomial logistic regression and an MLP with two layers (200, 50). The results provided in Figure 5 are aggregated results. It can be seen that ID-CVAE presents the best overall results. In particular, ID-CVAE obtains an F1 score of 0.79 and an accuracy and recall of 0.80 which are the highest among the algorithms. The good results of ID-CVAE, compared with the alternative classifiers, indicate that ID-CVAE can better deals with the unbalanced and highly noisy data used in an NIDS. This behavior could be explained by the two-step process used to perform the classification, where the first step performs a stochastic data modeling and the second applies a discriminative approach to choose the best classification label. From the results, it seems that a combination of generative and discriminative methods is more appropriate for this kind of data.

Figure 6 shows one vs. rest detailed performance metrics for 5-labels classification using the ID-CAVE algorithm. We can observe how the frequency distribution for the labels is highly unbalanced (row “Frequency” in Figure 6). We get an F1 score greater than 0.8 for the most frequent labels. The behavior of lower frequency labels is quite noisy due to the nature of the training and test datasets. The accuracy obtained is always greater than 0.83 regardless of the label.



	Random Forest	Linear SVC	Multinomial Log. Reg.	Multilayer Perceptron	ID-CVAE
Accuracy	0.7391	0.7560	0.6602	0.7831	0.8010
F1	0.6909	0.7295	0.6066	0.7429	0.7908
Precision	0.7708	0.7622	0.6570	0.8262	0.8159
Recall	0.7391	0.7560	0.6602	0.7831	0.8010

Figure 5. Classification performance metrics (aggregated) vs. different classifiers.



	NORMAL	DOS	R2L	PROBE	U2R
Frequency	43.08%	33.08%	11.33%	10.74%	1.77%
Accuracy	0.8342	0.9266	0.9149	0.9385	0.9698
F1	0.8268	0.8839	0.4721	0.7176	0.0633
Precision	0.7519	0.9276	0.7937	0.7076	0.0703
Recall	0.9182	0.8441	0.3359	0.7278	0.0575
FPR	0.0818	0.1559	0.0112	0.0362	0.0137
NPV	0.7519	0.9276	0.9210	0.9671	0.9830

Figure 6. Classification performance metrics (One vs. Rest) vs. intrusion label.

Table 1 presents the confusion matrix for the classification of five labels. The confusion matrix provides a sample count according to actual (ground-truth) and predicted labels. The table also provides totals and percentages along rows and columns. From Table 1, the number of correctly classified and misclassified samples can be easily obtained. In this table, we can see that the classification of the label R2L presents more difficulties due to the number of

misclassifications. The U2R label has the worst results, which is expected given the very low frequency of this label (1.77%).

Table 1. Classification confusion matrix.

		Prediction						
		DoS	**Normal**	**Probe**	**R2L**	**U2R**	**Total**	**Percentage (%)**
Ground Truth	**DoS**	6295	916	61	162	24	7458	33.08%
	Normal	119	8917	610	36	29	9711	43.08%
	Probe	368	252	1762	18	21	2421	10.74%
	R2L	4	1430	32	858	230	2554	11.33%
	U2R	0	345	25	7	23	400	1.77%
	Total	6786	11860	2490	1081	327	22544	100.00%
	Percentage (%)	30.10%	52.61%	11.05%	4.80%	1.45%	100.00%	

In Table 2 we present the empirical results to determine the position for inserting the labels in the decoder. The insertion in the second layer provides the best classification results, we can also observe that the position of insertion is an important element to consider when defining the architecture of the model.

Table 2. Impact of layer used in the decoder to insert the labels.

Model	Accuracy	F1	Precision	Recall
Labels inserted in first layer of decoder	0.7791	0.7625	0.7888	0.7791
Labels inserted in second layer of decoder (ID-CVAE)	0.8010	0.7908	0.8159	0.8010
Labels inserted in third layer of decoder	0.7547	0.7389	0.7584	0.7547

All results presented in this Section (label frequency, confusion matrix, and performance metrics) are calculated using the full NSL-KDD test dataset

4.2. Feature Reconstruction

ID-CVAE can perform feature reconstruction. Figure 7 shows the process that consists of three phases (in order): training, prediction and reconstruction phase.

The objective here will be to reconstruct missing features in an intrusion detection test dataset, i.e., one with unknown labels. To do that, we start by training an ID-CVAE with the dataset with missing features (shown in Figure 7 as $\boldsymbol{X}^{-f}$), but using the complete dataset as reference (Figure 7, training phase).

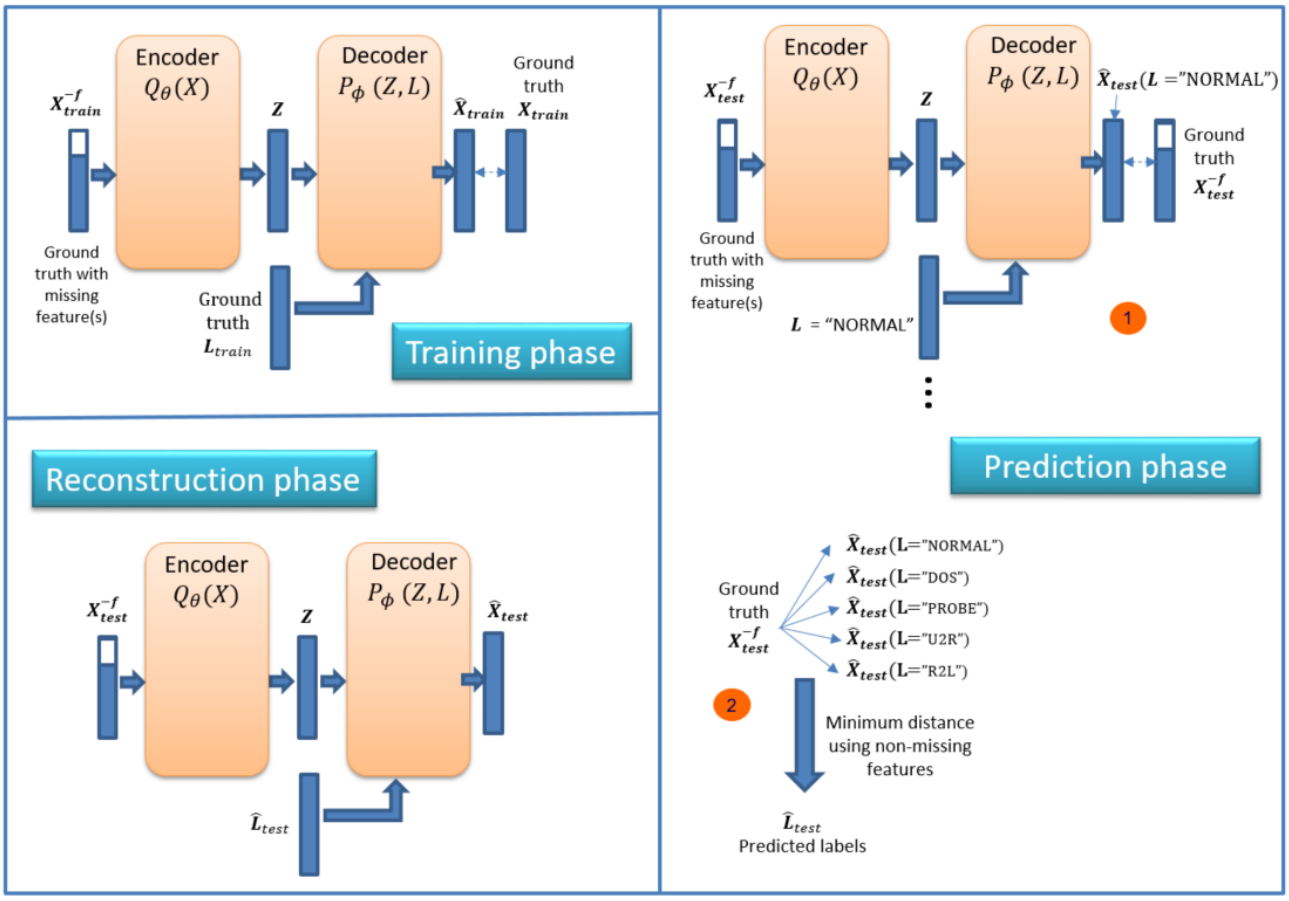


Figure 7. Feature reconstruction framework.

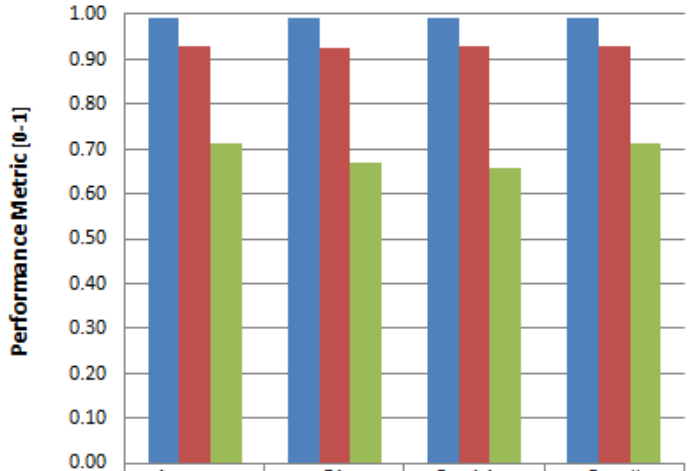
Then we need to perform a prediction phase (Figure 7). The input to this phase will be again the dataset with missing features. We do the prediction in a similar way as presented in Section 4.1, to perform classification; the only difference is that we will have a recovered test dataset with more features than the original one (used as reference). That creates a possible problem to perform the distance calculation between them. In order to solve this problem, we use only the non-missing features in both datasets to perform the distance calculation.

Once the predicted labels are obtained, we use these labels together with the original test dataset (with missing features) as inputs to our model, having as output the recovered test dataset (with all features), as can be seen in Figure 7 (reconstruction phase).

It is important to note that we only need the training dataset for the training phase. This training dataset contains the full set of features, from which we extract a reduced set $\boldsymbol{X}^{-f}_{train}$ (a subset of the features from the training data) that will be used to train the model. After the training phase, we will only have access to a test dataset with missing features $\boldsymbol{X}^{-f}_{test}$ (the same missing features provided at the training phase). The ground truth $\boldsymbol{X}_{test}$ (with all features) is always unknown, and we try to approximate it in two phases: first, we use an initial reconstruction of the test dataset to predict the most probable associated labels $\hat{\boldsymbol{L}}_{test}$, and, second, using these labels we perform a final reconstruction $\hat{\boldsymbol{X}}_{test}$, that tries to recover the unknown ground truth $\boldsymbol{X}_{test}$.

When we apply the feature reconstruction process to the NSL-KDD Test dataset, we obtain the results given in Figure 8. This figure provides performance metrics when recovering some missing features; in particular: *protocol*, *flag* and *service* features; all of them categorical. As expected, the achievable prediction accuracy is related to the number of values of the categorical feature. The features *protocol* and *flag* with three and 11 values, respectively, have an accuracy of 99% and 92%, while the feature *service* with 70 values has an accuracy of 71%,

which is not a bad result considering the large number of values of this feature. It would be necessary to obtain more training data to achieve a better result in the recovery of features with many values, due to the greater uncertainty associated with a label with more values. The remaining metrics (F1, precision, and recall) follow a similar pattern (Figure 8).



	Accuracy	F1	Precision	Recall
Missing feature: protocol	0.9902	0.9903	0.9904	0.9902
Missing feature: flag	0.9284	0.9240	0.9293	0.9284
Missing feature: service	0.7142	0.6678	0.6584	0.7142

Figure 8. Performance metrics (aggregated) for predicting missing features of NSL-KDD test dataset.

Figures 9–11 provide detailed results on reconstruction performance for each of the three recovered features (*protocol*, *flag*, and *service*). Each figure provides the name and frequency distribution for feature's values, together with one vs. rest reconstruction metrics. The reconstruction metrics are given for each value of the reconstructed feature (one vs. rest). We can see that in all cases the frequency distribution of values is remarkably unbalanced (column "Frequency" in Figures 9–11). The unbalanced distribution creates additional problems to the recovery task. There are cases where this unbalanced scenario is so strong that the algorithm cannot recover certain values of the reconstructed feature. This is the case in Figure 10, where several rare values of the *flag* feature (S1, S2, and RSTOS0) have an F1 equal or close to zero, implying that we cannot predict any positive occurrence of these values. This happens for values with extremely low frequency (less than 0.09%).

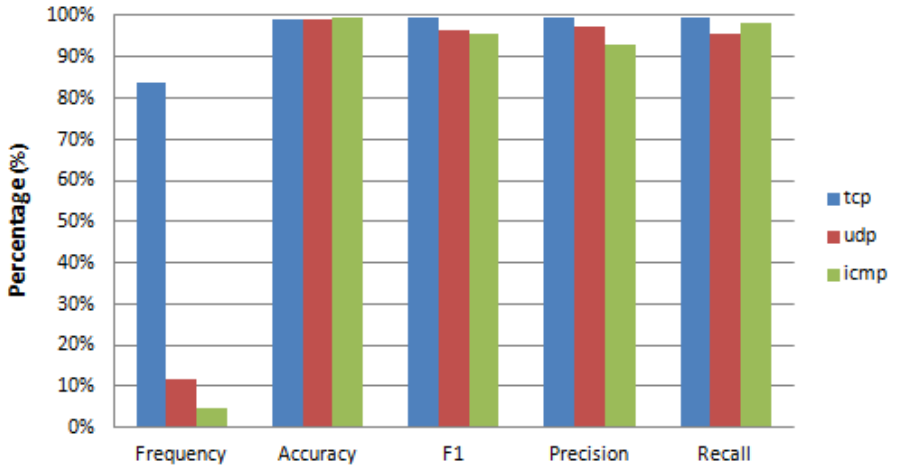


Figure 9. Performance metrics (One vs. Rest) for reconstruction of all features values when feature: 'protocol' is missing.

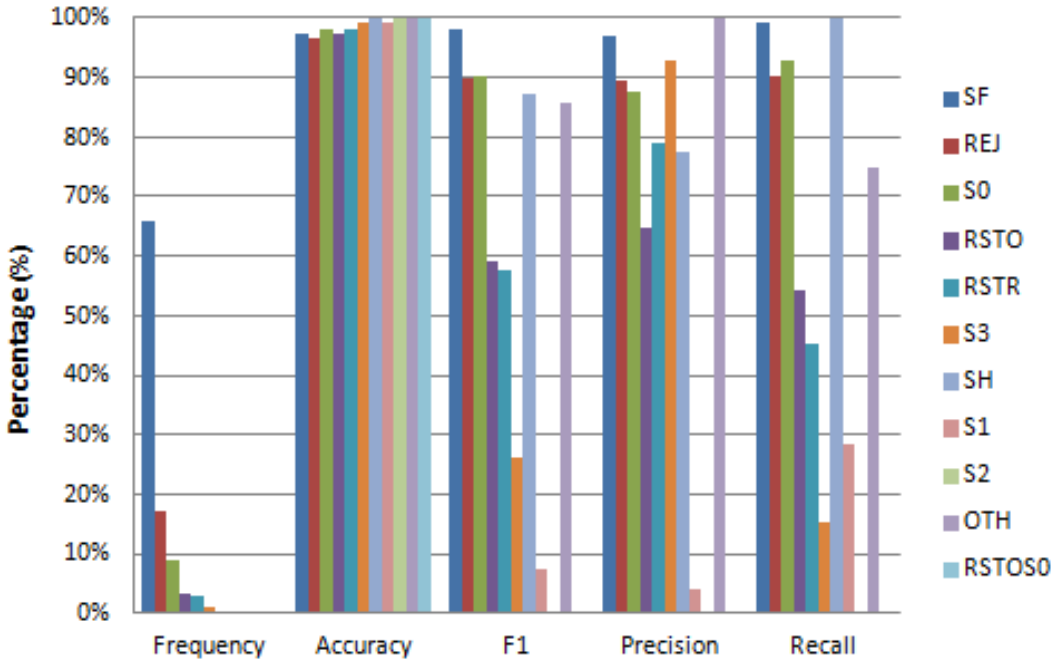


Figure 10. Performance metrics (One vs. Rest) for reconstruction of all features values when feature: 'flag' is missing.

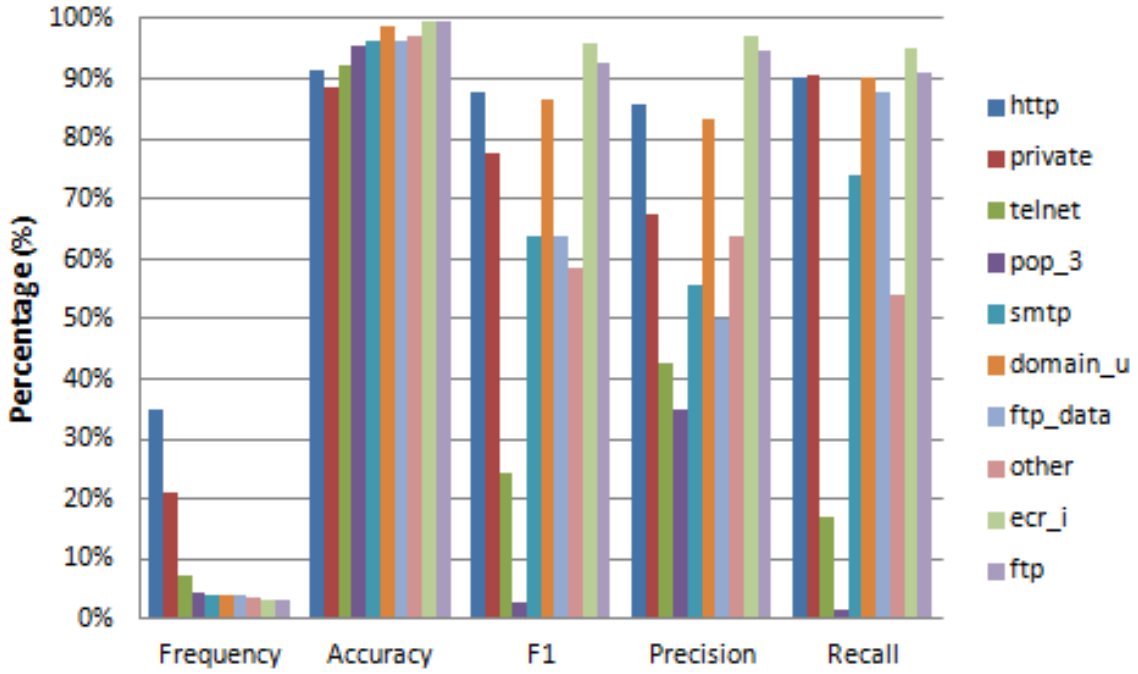


Figure 11. Performance metrics (One vs. Rest) for reconstruction of all features values when feature: '*service*' is missing.

Figures 9 and 10 present results for the recovery metrics for all values of the *protocol* and *flag* features (with three and 11 values, respectively) and Figure 11 presents results for the 10 most frequent values of the *service* feature, which has 70 values in total.

In Figure 9, when recovering the *protocol* feature we can achieve an F1 score of not less than 0.96 for any value of the feature, regardless of its frequency.

While recovering the *flag* feature (Figure 10), we obtain an accuracy always greater than 0.97 and an F1 score greater than 0.9 for the most frequent values.

Similarly, when recovering the *service* feature, we get an accuracy greater than 0.89 for the 10 most frequent values of this feature and a noisy F1 score with a higher value of 0.96.

In Table 3, we present the confusion matrix for the recovery of the three values of the feature: 'protocol'. This is information similar to that given for the classification case (Section 4.1). Table 4 also shows detailed performance metrics such as those provided for the classification case.

We only present detailed data (as in Tables 3-4) for the case of recovery of the feature: 'protocol'. Similar data could be presented for the other two discrete features, but their large number of values would provide too much information to be useful for analysis.

The description of the data presented in Tables 3-4 is similar to the data presented in Table 1 and Figure 6.

Table 3. Confusion matrix for reconstruction of all features values when feature: '*protocol'* is missing.

		Prediction				
		icmp	**tcp**	**udp**	**Total**	**Percentage (%)**
Ground Truth	**icmp**	1022	19	2	1043	4.63%
	tcp	13	18791	76	18880	83.75%
	udp	7	79	2535	2621	11.63%
	Total	1042	18889	2613	22544	100.00%
	Percentage (%)	4.62%	83.79%	11.59%	100.00%	

Table 4. Detailed performance metrics for reconstruction of all features values when feature: '*protocol'* is missing.

Label value	Frequency	Accuracy	F1	Precision	Recall	FPR	NPV
tcp	83.75%	0.9917	0.9950	0.9948	0.9953	0.0267	0.9757
udp	11.63%	0.9927	0.9687	0.9701	0.9672	0.0039	0.9957
icmp	4.63%	0.9982	0.9803	0.9808	0.9799	0.0009	0.9990

So far, we have only covered the reconstruction of discrete features. However, we have also done the experiment to recover all continuous features using only the three discrete features to perform the recovery. We obtained a Root Mean Square Error (RMSE) of 0.1770 when retrieving the 32 continuous features from the discrete features. All performance metrics are calculated using the full NSL-KDD test dataset.

4.3. Model Training

These are lessons learned about training the models: The inclusion of drop-out as regularization gives worse results. Having more than two or three layers for the encoder or

decoder does not improve the results, making the training more difficult. It is important to provide a fair number of epochs for training the models, usually 50 or higher.

We have used Tensorflow to implement all the ID-CVAE models, and the python package scikit-learn [27] to implement the different classifiers. All computations have been performed on a commercial PC (i7-4720-HQ, 16 GB RAM).

5. Conclusions and Future Work

This work is unique in presenting the first application of a conditional VAE (CVAE) to perform classification on intrusion detection data. More important, the model is also able to perform feature reconstruction, for which there is no previous published work. Both capabilities can be used in current NIDS, which are part of network monitoring systems, and particularly in IoT networks [1].

We have demonstrated that the model performs extremely well for both tasks, being able for example to provide better classification results on the NSL-KDD Test dataset than well-known algorithms: random forest, linear SVM, multinomial logistic regression and multi-layer perceptron.

The model is also less complex than other classifier implementations based on a pure VAE. The model operates creating a single model in a single training step, using all training data irrespective of their associated labels. While a classifier based on a VAE needs to create as many models as there are distinct label values, each model requiring a specific training step (one vs. rest). Training steps are highly demanding in computational time and resources. Therefore, reducing its number from n (number of labels) to 1 is an important improvement.

When doing feature reconstruction, the model is able to recover missing categorical features with three, 11 and 70 values, with an accuracy of 99%, 92%, and 71%, respectively. The reconstructed features are generated from a latent multivariate probability distribution whose parameters are learned as part of the training process. This inferred latent probability distribution serves as a proxy for obtaining the real probability distribution of the features. This inference process provides a solid foundation for synthesizing features as similar as possible to the originals. Moreover, by adding the sample labels, as an additional input to the decoder, we improve the overall performance of the model making its training easier and more flexible.

Extensive performance metrics are provided for multilabel classification and feature reconstruction problems. In particular, we provide aggregated and one vs. rest metrics for the predicted/reconstructed labels, including accuracy, F1 score, precision and recall metrics.

Finally, we have presented a detailed description of the model architecture and the operational steps needed to perform classification and feature reconstruction.

Considering future work, after corroborating the good performance of the conditional VAE model, we plan to investigate alternative variants as ladder VAE [28] and structured VAE [29], to explore their ability to learn the probability distribution of NIDS features.

Acknowledgments: This work has been partially funded by the Ministerio de Economía y Competitividad del Gobierno de España and the Fondo de Desarrollo Regional (FEDER) within the project “Inteligencia distribuida para el control y adaptación de redes dinámicas definidas por software, Ref: TIN2014-57991-C3-2-P”, and the Project “Distribucion inteligente de servicios multimedia utilizando redes cognitivas adaptativas definidas por software”, Ref: TIN2014-57991-C3-1-P, in the Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento.

Author Contributions: Manuel Lopez Martín conceived and designed the original models and experiments. Belen Carro and Antonio Sanchez-Esguevillas have supervised the work, guided the experiments and critically reviewed the paper to produce the manuscript. Jaime Lloret

provided background information on related experiments in IoT and co-guided the course of research.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

[1] Zarpelo, B.B.; Miani, R.S.; Kawakani, C.T.; de Alvarenga, S.C. A survey of intrusion detection in Internet of Things. J. Netw. Comput. Appl. 2017, 84, 25–37, doi:10.1016/j.jnca.2017.02.009.

[2] Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Network Anomaly Detection: Methods, Systems and Tools. In IEEE Communications Surveys & Tutorials; IEEE: Piscataway, NJ, USA, 2014; Volume 16, pp. 303–336, doi:10.1109/SURV.2013.052213.00046.

[3] Aggarwal, C.C. Outlier Analysis; Springer: New York, NY, USA, 2013; pp. 10–18, ISBN 978-1-4614-639-5.

[4] Kingma, D.P.; Rezende, D.J.; Mohamed, S.; Welling, M. Semi-supervised learning with deep generative models. In Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14), Montreal, QC, Canada, 8–13 December 2014, Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; MIT Press: Cambridge, MA, USA, 2014; pp. 3581–3589.

[5] Sohn, K.; Yan, X.; Lee, H. Learning structured output representation using deep conditional generative models. In Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15), Montreal, QC, Canada, 7–12 December 2015, Cortes, C., Lee, D.D., Sugiyama, M., Garnett, R., Eds.; MIT Press: Cambridge, MA, USA, 2015; pp. 3483–3491.

[6] Fekade, B.; Maksymyuk, T.; Kyryk, M.; Jo, M. Probabilistic Recovery of Incomplete Sensed Data in IoT. IEEE Int. Things J. 2017, 1, doi:10.1109/JIOT.2017.2730360.

[7] An, J.; Cho, S. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. Seoul National University, Seoul, Korea, SNU Data Mining Center, 2015–2 Special Lecture on IE, 2015.

[8] Suh, S.; Chae, D.H.; Kang, H.G.; Choi, S. Echo-state conditional Variational Autoencoder for anomaly detection. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016, pp. 1015–1022, doi:10.1109/IJCNN.2016.7727309.

[9] Sölch, M. Detecting Anomalies in Robot Time Series Data Using Stochastic Recurrent Networks. Master's Thesis, Department of Mathematics, Technische Universitat Munchen, Munich, Germany, 2015.

[10] Hodo, E.; Bellekens, X.; Hamilton, A. Threat analysis of IoT networks using artificial neural network intrusion detection system. In Proceedings of the 2016 International Symposium on Networks, Computers and Communications (ISNCC), Yasmine Hammamet, Tunisia, 11–13 May 2016; pp. 1–6, doi:10.1109/ISNCC.2016.7746067.

[11] Kang, M.-J.; Kang, J.-W. Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. PLoS ONE 2016, 11, e0155781, doi:10.1371/journal.pone.0155781.

[12] Thing, V.L.L. IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 19–22 March 2017, pp. 1–6, doi:10.1109/WCNC.2017.7925567.

[13] Ma, T.; Wang, F.; Cheng, J.; Yu, Y.; Chen, X. A Hybrid Spectral Clustering and Deep

Neural Network Ensemble Algorithm for Intrusion Detection in Sensor Networks. Sensors 2016, 16, 1701.

[14] Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp.1–6, doi:10.1109/CISDA.2009.5356528.

[15] Ingre, B.; Yadav, A. Performance analysis of NSL-KDD dataset using ANN. In Proceedings of the 2015 International Conference on Signal Processing and Communication Engineering Systems, Guntur, India, 2–3 January 2015; pp. 92–96, doi:10.1109/SPACES.2015.7058223.

[16] Ibrahim, L.M.; Basheer, D.T.; Mahmod, M.S. A comparison study for intrusion database (KDD99, NSL-KDD) based on self-organization map (SOM) artificial neural network. In Journal of Engineering Science and Technology; School of Engineering, Taylor's University: Selangor, Malaysia, 2013; Volume 8, pp. 107–119.

[17] Wahb, Y.; ElSalamouny, E.; ElTaweel, G. Improving the Performance of Multi-class Intrusion Detection Systems using Feature Reduction. arXiv 2015, arXiv:1507.06692.

[18] Bandgar, M.; dhurve, K.; Jadhav, S.; Kayastha, V.; Parvat, T.J. Intrusion Detection System using Hidden Markov Model (HMM). IOSR J. Comput. Eng. (IOSR-JCE) 2013, 10, 66–70.

[19] Chen, C.-M.; Guan, D.-J.; Huang, Y.-Z.; Ou, Y.-H. Anomaly Network Intrusion Detection Using Hidden Markov Model. Int. J. Innov. Comput. Inform. Control 2016, 12, 569–580.

[20] Alom, M.Z.; Bontupalli, V.; Taha, T.M. Intrusion detection using deep belief networks. In Proceedings of the 2015 National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 June 2015, pp. 339–344.

[21] Xu, J.; Shelton, C.R. Intrusion Detection using Continuous Time Bayesian Networks. J. Artif. Intell. Res. 2010, 39, 745–77.

[22] Hodo, E.; Bellekens, X.; Hamilton, A.; Tachtatzis, C.; Atkinson, R. Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey. arXiv 2017, arXiv:1701.02145.

[23] Khan, S.; Lloret, J.; Loo, J. Intrusion detection and security mechanisms for wireless sensor networks. Int. J. Distrib. Sens. Netw. 2017, 10, 747483.

[24] Alrajeh, N.A.; Lloret, J. Intrusion detection systems based on artificial intelligence techniques in wireless sensor networks. Int. J. Distrib. Sens. Netw. 2013, 9, 351047.

[25] Han, G.; Li, X.; Jiang, J.; Shu, L.; Lloret, J. Intrusion detection algorithm based on neighbor information against sinkhole attack in wireless sensor networks. Comput. J. 2014, 58, 1280–1292.

[26] Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. ArXiv e-prints, arXiv: 1312.6114v10 [stat.ML], 2014.

[27] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 2011, 12, 2825–2830.

[28] Sønderby, C.K.; Raiko, T.; Maaløe, L.; Sønderby, S.K.; Winther, O. Ladder Variational Autoencoders. ArXiv e-prints, arXiv:1602.02282v3 [stat.ML], 2016.

[29] Johnson, M.J.; Duvenaud, D.; Wiltschko, A.B.; Datta, S.R.; Adams, R.P. Structured VAEs: Composing Probabilistic Graphical Models and Variational Autoencoders. ArXiv e-prints, arXiv:1603.06277v1 [stat.ML], 2016.

PAPER 4

Deep learning model for multimedia Quality of Experience prediction based on network flow packets

Manuel Lopez-Martin, Belen Carro, Jaime Lloret, Santiago Egea, Antonio Sanchez-Esguevillas

***Abstract*— Quality of Experience (QoE) is the overall acceptability of an application or service, as perceived subjectively by the end user. In particular for Video Quality (VQ) the QoE is dependent of video transmission parameters. To monitor and control these parameters is critical in modern network management systems, but it would be better to be able to monitor the QoE itself (both in terms of interpretation and accuracy) rather than the parameters on which it depends. In this paper we present the first attempt to predict video QoE based on information directly extracted from the network packets using a deep learning model. The QoE detector is based on a binary classifier (good or bad quality) for seven common classes of anomalies when watching videos (blur, ghost...). Our classifier can detect anomalies at the current time instant and predict them at the next immediate instant. This classifier faces two major challenges: first, a highly unbalanced dataset with a low proportion of samples with video anomaly, and second, a small amount of training data, since it must be obtained from individual viewers under a controlled experimental setup. The proposed classifier is based on a combination of a Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Gaussian Process (GP) classifier. Image processing which is the common domain for a CNN has been expanded to QoE detection. Based on a detailed comparison, the proposed model offers better performance metrics than alternative machine learning algorithms, and can be used as a QoE monitoring function in edge computing.**

***Index Terms*—Quality of Experience; Convolutional Neural Network; Deep Learning; Recurrent Neural Network**

M. Lopez, B. Carro, A. Sanchez and S. Egea are with Universidad de Valladolid
J. Lloret is with Universitat Politecnica de Valencia

I. INTRODUCTION

QoE is defined by ITU-T as "the overall acceptability of an application or service, as perceived subjectively by the end user". The ability to evaluate the QoE in a communication system, and especially in a system involved in video transmission, is critical. One of the main objectives of modern network management systems is to monitor and guarantee end-user Quality of Experience (QoE), hence the importance of an accurate QoE monitoring system. This need is even greater with highly configurable networks (e.g. Software Defined Networks (SDN) and edge computing), where precise and reliable information about end-user quality perception is needed to dynamically reconfigure network resources [1,2].

Edge computing is a way to streamline the flow of traffic between cloud computing services

and particular devices (e.g. IoT) and provide real-time local data analysis at the edge of the network, near the source of the data. The capabilities provided by edge computing can be improved if they are leveraged using real-time QoE estimates. This is even more valuable for video transmission networks whose real-time nature makes more important a rapid reaction to QoE degradation [1,2,3,4].

Fig. 1 shows an abstract view of data distribution and processing services for IoT applications. The cloud/central services are responsible for application management and overall coordination. The end devices (IoT devices) produce and consume operational data and commands. Finally, the distribution/edge processing services (middle layer in Fig. 1) are intended to facilitate communication, increase availability and performance and add distributed services closer to the end devices. This middle layer can host services that would otherwise be difficult to deploy at the cloud location (slow and unreliable access) or at the IoT devices (lacking processing capabilities). The QoE predictor proposed here is intended to be deployed as a quality monitoring service at the edge processing layer in Fig. 1.

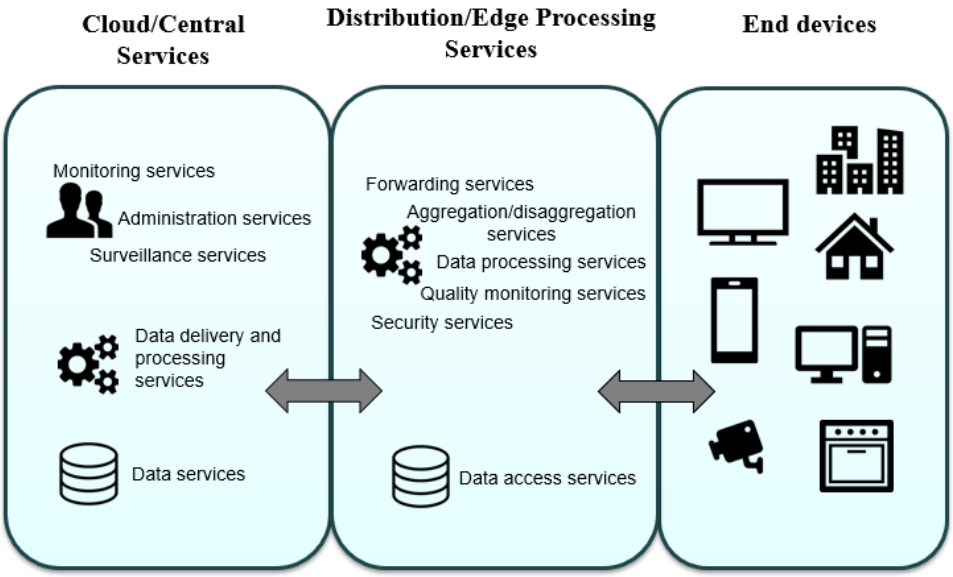


Fig. 1. Abstract view of data distribution and processing services for IoT applications

As the demand for video services increases in parallel with the storage and processing capabilities at the edge layer [3], it is now possible to host highly demanding video processing services in this layer, which allows to offer new network capacities based on automatic and intelligent analysis of video transmissions and QoE-aware network management and video traffic prioritization and scheduling [1,2,3,6]. Hence the importance of more robust and accurate QoE predictors that can make better use of the new processing platforms (e.g. GPUs) at the edge layer [3]. Our proposed predictor is based in a deep learning model that is especially suitable for these new platforms.

At present, the usual way to evaluate QoE is either to carry out experiments with individuals as testers or to calculate it indirectly from Quality of Service (QoS) network parameters (jitter, delay, packet loss,..) [4,5,6]. Another approach, recently being actively explored is applying machine learning (ML) to video QoE estimation. The resulting QoE detector is able to predict QoE directly from information contained in the transmitted videos, the network packets or end-user recorded events (e.g. related web activity). This approach is the one taken on this work in order to build a video QoE detector from network packets information using deep learning

models. This is also the most advanced and precise approach [6] that shifts the focus of video quality assessment from QoS (system oriented) to QoE (user oriented).

Building a QoE detector raises important challenges. First, it is difficult to construct a training dataset, since it is obtained in a controlled experiment with several individuals who have to evaluate the quality of the video. This makes it very difficult to acquire large datasets, which are normally needed to train a classification algorithm. Secondly, the training datasets are highly unbalanced, as the number of errors observed in the videos is normally much smaller than the number of non-anomaly events. And third, the subjective judgment of quality, assumed by QoE, necessarily implies noisy results (even using Mean Opinion Score (MOS)), which can make it even more difficult to assess the performance of the algorithms.

Additionally to all former considerations, other important objective in our case was to have a QoE detector which could be integrated into a network management system to monitor network quality (as observed by the end-user), allowing at the same time an efficient network reconfiguration and control (in our case an SDN network). Therefore, QoE detector could identify the QoE score of the video transmitted at the current time-interval, but also be able to anticipate (predict) the quality score for the next time-interval. Since, this prediction can be crucial to anticipate actions on network resources.

Having in mind these challenges, the proposed QoE detector consists of a deep learning classifier that is based on the combination of a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN) with a final Gaussian Process (GP) classifier. The classifier implements a binary classification (good or bad quality) for seven usual classes of video anomalies (blur, ghost, columns, chrominance, blockness, color bleeding and black pixel [6]) that can happen when watching the videos.

In order to design the final model, we have tried several alternative architectures and different ML models. We present a complete analysis of the results obtained from these alternative models. The impact of several algorithms' hyper-parameters and design decisions has also been analyzed. Similarly, the process for generating and transforming the training data is presented in detail.

QoE detector utilizes a training dataset created specifically for this work. The dataset was obtained from a controlled experiment in which several individual viewers evaluated video transmissions in a time interval of 1-second and under different network configurations.

The main contributions of this work are: (1) First application of deep learning models to video QoE prediction. (2) Prediction based on network packet information. (3) Network flows treated as pseudo-images that allow applying a CNN. (4) Excellent prediction performance for not extremely unbalanced labels with a small dataset.

The structure of the paper is the following: Related work is presented in Section II. The work performed is described in Section III. The results are discussed in Section IV and finally, Section V provides discussion and conclusions.

II. RELATED WORK

There is no similar work in the literature presenting a deep learning solution to video QoE assessment based on information contained in network packets and trained with end-user QoE evaluations in a controlled experiment.

Nevertheless, there is a solid work done on automatic Video Quality Assessment (VQA) based on the identification and processing of parameters extracted from the video. In [6] a thorough review of QoE modeling and methodologies is presented. Authors in [4,5] propose an analytical expression for video QoE calculation based on several parameters: jitter, delay,

bandwidth, loss packets and zapping time for IPTV video transmissions.

There is also relevant literature on ML models that are applied to features extracted from the videos (or network packets) in order to rank its quality, usually in accordance with quality assessments obtained from end-users. In this line, in [7] they use QoS parameters to predict QoE using a dataset built from subjective end-user scores, and applying machine learning algorithms based on Support Vector Machine (SVM) and Decision Trees.

A survey of ML techniques used to capture the relationship between QoS parameters and QoE scores is provided in [8], where most of the common machine learning algorithms (Linear Discriminant Analysis, Random Forest (RF), SVM, Naïve Bayes, K-Nearest Neighbors) are applied to the automatic identification of QoE from QoS network parameters.

Considering Content Delivery Networks (CDN), [9] gives a review of the reasons why developing an objective method of quality assessment based on video transmission parameters is extremely difficult due to the complex relationships between these parameters, the user's perception and even the nature of the content. Furthermore, the authors propose the application of ML algorithms (Decision Trees, Naïve Bayes and Logistic Regression) to predict the QoE based on transmission parameters (bitrates, latency,..) and end-user engagement attributes (playtime, number of visits,..).

The prediction of streaming video QoE is proposed in [10] applying several regression models such as Ridge and Lasso Regression, and ensemble methods such as Random Forest (RF), Gradient Boosting (GB) and Extra Trees (ET).

None of the above references apply the new deep learning models and they do not provide a short-term QoE prediction based on network packet information. The QoE score generally provided is a single score in contrast to the simultaneous prediction of seven QoE anomalies/errors, which is provided in this paper. Comparison of performance results between these works is not significant, since the datasets used and the areas of application are too different. In this context, the present work has to be considered as an alternative option available in this topic area.

III. WORK DESCRIPTION

This section presents the experimental configuration employed to generate the training data, the necessary data preparation and a description and comparison of the prediction models applied.

A. Experimental setup: data generation

To generate the data that the QoE prediction models will use, it was necessary to establish an experimental setup that would allow identifying the QoE of the video transmissions while recording the associated network flow packets. The resulting data are multivariate time series, in time-steps of 1-second, which contain the network packets transmitted in each time-step plus the presence or absence of seven video transmission errors in that time-step.

The topology of the experimental setup included three components: (1) A video transmission server, which allowed us to vary the characteristics of the video. (2) The clients, where the video streams are visualized by the end user to label them with QoE errors. And, (3) a packet analyzer (Wireshark) that extracts the network parameters on the end user's side. This configuration allows us to vary several network and video features (jitter, delay, bandwidth, packet loss, bitrate ...) and test their impact on network packets and their associated visual

effects. We used several network protocols (HTTP, RTP and UDP) to increase the variety of video transmissions.

B. Data preparation

The data generated, as described in the previous section, is further processed to extract aggregate information associated with each time-step, in 1-second intervals. The new features formed by these aggregates are organized into samples, finally forming a time-series of vectors (samples).

To build the training dataset, an ad-hoc application was developed as a feature extractor. The feature extractor identifies packets belonging to a specific multimedia transmission, extracting certain IP header information from the packets, namely the size of the application layer and the inter-arrival time between consecutive packets. Later, these two features are expanded in a collection of 40 statistical attributes that includes means, standard deviations, root mean squares, maximums, minimums and percentiles. In addition, the number of packets transferred in the ingoing and outgoing directions is counted and also included as a feature. All these features are normalized to the range [0-1] with a previous log normalization for features with high values ranges.

Finally, the QoE information provided by the end users in terms of the possible errors observed in each time-step is appended to the collection of attributes as labels. We have evaluated seven QoE errors: columns, blur, ghost, chrominance, blockness, color bleeding and black pixel. The resulting vector time-series are described in Fig. 2.a.

To train with the least possible number of samples, we perform an additional transformation of the data in Figure 2.a to arrange it in small elementary flows used for training (see Figure 2.b).

Figure 2 shows the complete process to obtain and transform the training data.

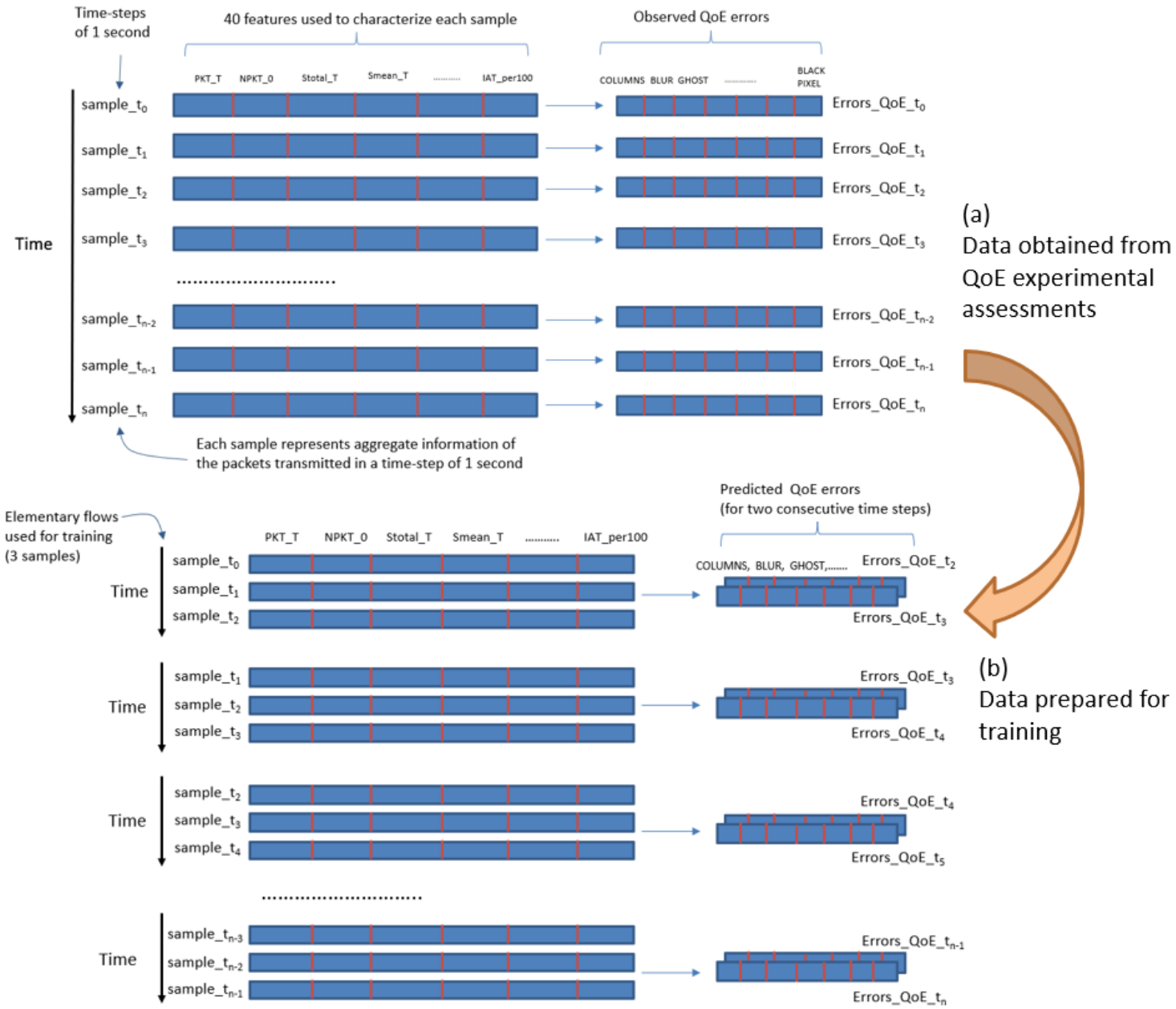


Fig. 2. Training data formed by aggregate data samples (a) and final configuration of the training data, arranged to be used by the models (b).

Fig. 2.b shows the training data ready to be finally used by the models. It can be seen that the data is arranged in small "elementary" flows of 3 samples (corresponding to an elapsed time of 3 seconds). These elementary flows form small vector time-series which are the data entry for training and prediction. The flows are obtained according to a sliding window of width 3 and offset 1 applied to the data in Fig. 2.a. The offset causes the successive flows to have one overlapping sample. For each elementary flow, the models will be trained with QoE errors for the current time-step and the next time-step. In this way, at prediction time, we will be able to detect which errors are occurring in the current time-step and predict errors in the next time interval.

Following the arrangement shown in Fig. 2.b, we finally obtain 2078 elementary flows, which we then divide into 1766 training flows and 312 test flows (15% of total flows). These will be the final data sets that will be used to train and validate all models presented in this

paper.

The result vectors for QoE errors are binary vectors (labels), with 1 indicating that an anomaly was present and 0 otherwise.

It is also important to mention the strong imbalance in the distribution of label values. Fig. 3 gives the frequency distribution for QoE label values. It can be observed that the absence of anomalies is much more frequent than the opposite. This fact adds an additional difficulty to the models and has to be considered in the analysis of results.

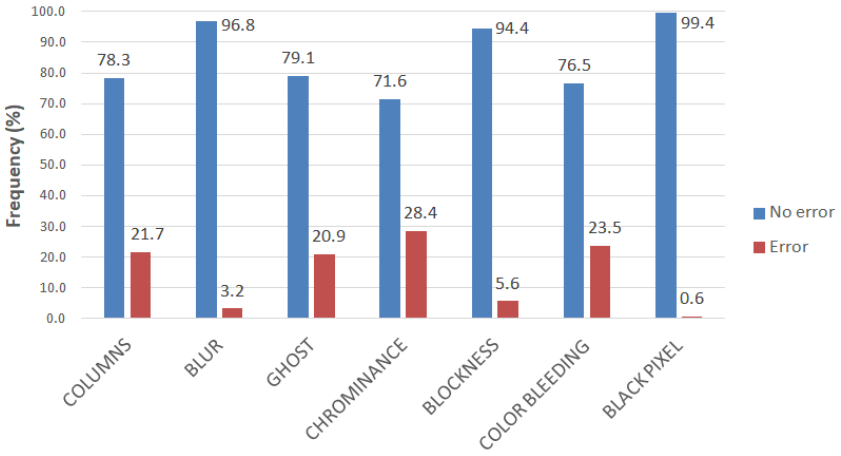


Fig. 3. Frequency distribution for QoE label values.

C. Models for QoE prediction

In this section, we present the different prediction models applied for this work.The prediction models studied can be organized into three groups: (1) classical ML models (Random Forest, Logistic Regression...), (2) deep learning models (CNN and RNN) and (3) a combination of deep learning models plus a Gaussian Process (GP) classifier.

The results obtained by the third group are the best, followed by the second group and the first group of models producing the worst results. Nevertheless, the results difference is not so much significative between the second and third group, but it is rather significative between these two groups and the first group. Results details are provided in section IV.

For the first group, the models studied have been: Linear Support Vector Classifier (SVC), SVC with an RBF kernel, Logistic Regression, Multilayer Perceptron (MLP), RF, Gradient Boosting Method (GBM) and Adaboost. For GBM and Adaboost we used decision trees as elementary classifiers.

Focusing on the second and third groups of algorithms, in order to find the best model for QoE prediction, we have explored several deep learning architectures. A model combining a CNN [11], RNN [12] and GP Classifier [13] has provided the best prediction performance. The implementation of the RNN network has been based on an LSTM [14], which is a special type of RNN.

A CNN [11] is a Neural Network (NN) that applies several convolutional filters to image-formatted data. The filter's weights are learned using Stochastic Gradient Descent (SGD). Each filter applied to the image generates a new scaled-down image that is arranged along a new dimension. The generated multi-dimensional image produces a new data representation that contains invariant properties of the original image at different scales and levels of

abstraction.

With a similar strategy to [15], to apply a CNN to a time-series of feature vectors, we assimilate the data to an associated pseudo-image (elementary flow), to which a CNN can be applied. An additional advantage of using deep learning models (mainly the use of the CNN network) is to provide automatic feature engineering (representation) of network flows.

An RNN [12] is an NN intended to treat time-series data, by iterating the NN with the sequential input data and an additional internal state. The internal state is updated at each iteration step. The internal state values are the output of the RNN. The output can be sampled at the end of the iteration process or at each step of the iteration producing as many outputs as iteration steps.

Considering all the architectures for the second and third group of models, we have arrived to three canonical models: Model 1, uses only RNN layers. Model 2, applies a sequence of CNN and RNN layers, and Model 3, adds a GP Classifier to a sequence of CNN and RNN layers. When applying the GP Classifier [13] we have used Model 2 (already trained) as our initial network, then using one of the last layers of this network as the input to the GP Classier. With this configuration, we have trained the GP Classifier by adjusting the lengthscale and variance parameters of the RBF kernel used in this case.

The GP Classifier is based on the so called Laplace approximation [13], which tries to approximate with a Gaussian function a non-Gaussian posterior formed by applying a logistic link function to the output of an intermediate latent function. The resulting squashed outcomes produced by the link function are associated to classification probabilities. The kernel [13] chosen has been a Radial Basis Function (RBF) kernel with two adjustable parameters: lengthscale and variance. To train the GP Classifier consists on tuning these two parameters plus an additional noise variance parameter associated with the likelihood of the model.

Considering the difficulties to apply a multi-label GP classifier, we have applied 14 independent GP binary classifiers (one per label). The GP classifier is a non-parametric algorithm that makes full use of the data available. Interestingly, the increase in performance obtained with the addition of the GP classifier is not so important as to discard the use of the simpler deep learning network, especially given the additional memory and time processing required by adding the GP classifier.

In Fig. 4, the best architectures obtained for Model 2 and 3 are presented (Model 1 being a subset of Model 2). The description of the architecture, given in Fig. 4, follows the notation provided in [15] where more details about the layers and their connections can be found. As a summary, a 2-dimensional image-formatted time-series of samples is presented to the network where a sequence of two CNN layers (first two layers in Fig. 4) extracts new features which are added to a new additional dimension (third dimension). This new dimension is flattened out again to two dimensions before entering into two additional LSTM layers. The LSTM layers are intended to process the time sequence information creating a final embedding of the data into a one-dimensional vector that is delivered to a fully connected final network that eventually generates the expected predictions. The second LSTM layer produces a single output (a vector) while the first one provides as many outputs as iteration steps; these multiple outputs are associated to an extra temporal dimension (a matrix in Fig. 4). The final layers are two fully connected layers.

For all architectures, the training was done with 200 epochs. An epoch is a single pass of all the samples in the training dataset. We have used Rectified Linear Units (ReLU) for the activation functions, except sigmoid activation for the last layer. The loss function employed was Binary Cross-Entropy and the optimization was performed with mini-batch Stochastic Gradient Descent (SGD) with Adam.

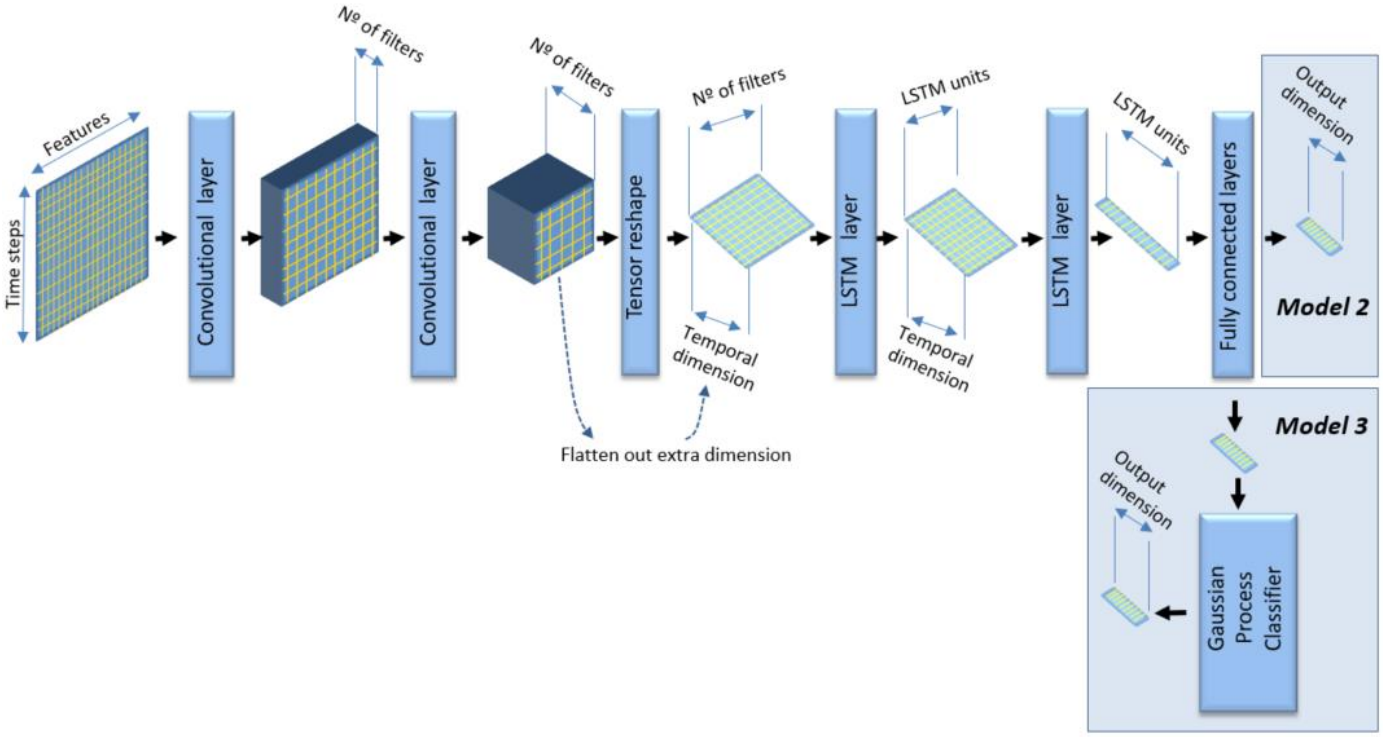


Fig. 4. Architecture for the best deep learning network proposed for this work (Model 2) and alternative combination with GP (Model 3). More details in [15].

IV. RESULTS

This section presents a discussion of results for the different models under evaluation. We focus the analysis on the impact of design decisions, mainly aimed at the architecture of the models and the length of the elementary network flows used for the training.

We provide the following performance metrics: accuracy, precision, recall, F1-score and Area Under the ROC Curve (AUC). F1-score and AUC are particularly suitable considering the highly unbalanced distribution of QoE errors.

For the definition of accuracy, F1, precision, recall and AUC we follow [15]. All performance metrics given in this section are obtained with a test dataset not used at any time during training.

A. Analysis of results

For this problem, we have 14 distinct QoE anomalies (labels) to be detected (7 are anomalies detected at the current time-step and other 7 are anomalies for the next time-step). The anomalies can occur simultaneously, being a multi-label classification problem, considering all the labels, but with a separate error probability calculation for each label. There are two possible ways to give results in this case: aggregated and one-by-one results.

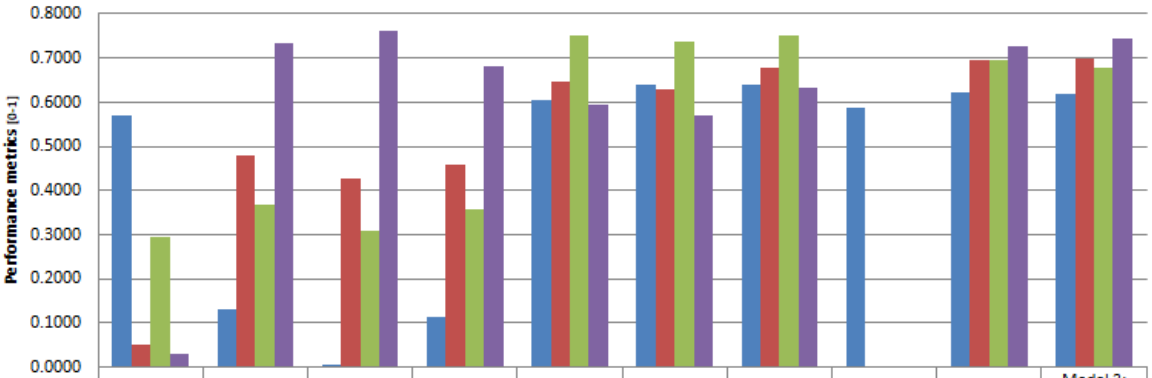
For one-by-one, we focus in a particular class (label) in isolation of the other labels, simplifying the problem to a binary classification task for each particular label (one by one). In the case of aggregated results, we try to give a summary result for all labels. There are different alternatives to perform the aggregation (micro, macro, samples, weighted), varying in the way the averaging process is done.

In this section, we provide the performance metrics obtained for all the models analyzed for this work. The models are described in a previous section (Section III.C). The performance metrics for all the models are presented in Fig. 5. The performance metrics in Fig. 5 are aggregated metrics using a weighted average.

In Fig. 5, we can see that Models 2 and 3 present the best results in terms of F1-score. Of these models, the best accuracy is given in Model 2 (0.6218) and the best F1-score is obtained for Model 3 (0.6987). The slightly better result of Model 3 has to be balanced with the greater needs of memory and processing time for this model. We see that, in general, the metrics obtained are not high, but these metrics are formed by adding the results of 14 labels. In addition, there are three highly unbalanced labels (blur, blockness and black pixel) that significantly worsen the results. This can be seen in Fig. 6, where the performance metrics are calculated for each label separately.

Considering the other models, Random Forest offers fairly good results but the final F1-score (0.6787) is below the best models due to poor results in the recall metric. Focusing on the F1-score, which is our preferred metric for aggregated results (Fig. 5), Model 3 provides a 3% increase over Random Forest. To calculate AUC scores it is necessary to obtain prediction probabilities, which can be cumbersome and not accurate for some models. For example, SVC requires an additional Platt scaling or some other alternative method to retrieve prediction probabilities. Therefore, the AUC score is not considered in Fig. 5 for aggregated results for all models.

It is important to highlight that the worst result in Fig. 5 is for Model 1. This model is formed exclusively by an RNN network and does not include a CNN in its architecture. This is a rather unexpected result as the inclusion of a CNN was not anticipated as something so critical, considering the time-series nature of the predictors.



	MLP	Linear SVC	SVC - RBF Kernel	Logistic Regression	GBM	Adaboost	Random Forest	Model 1: RNN	Model 2: CNN+RNN	Model 3: CNN+RNN+ GP
Accuracy	0.5705	0.1314	0.0064	0.1122	0.6058	0.6410	0.6378	0.5865	0.6218	0.6186
F1	0.0512	0.4799	0.4286	0.4585	0.6476	0.6301	0.6787	0	0.6965	0.6987
Precision	0.2938	0.3676	0.3098	0.3570	0.7495	0.7356	0.7513	0	0.6949	0.6792
Recall	0.0305	0.7327	0.7618	0.6814	0.5942	0.5693	0.6330	0.0000	0.7271	0.7424

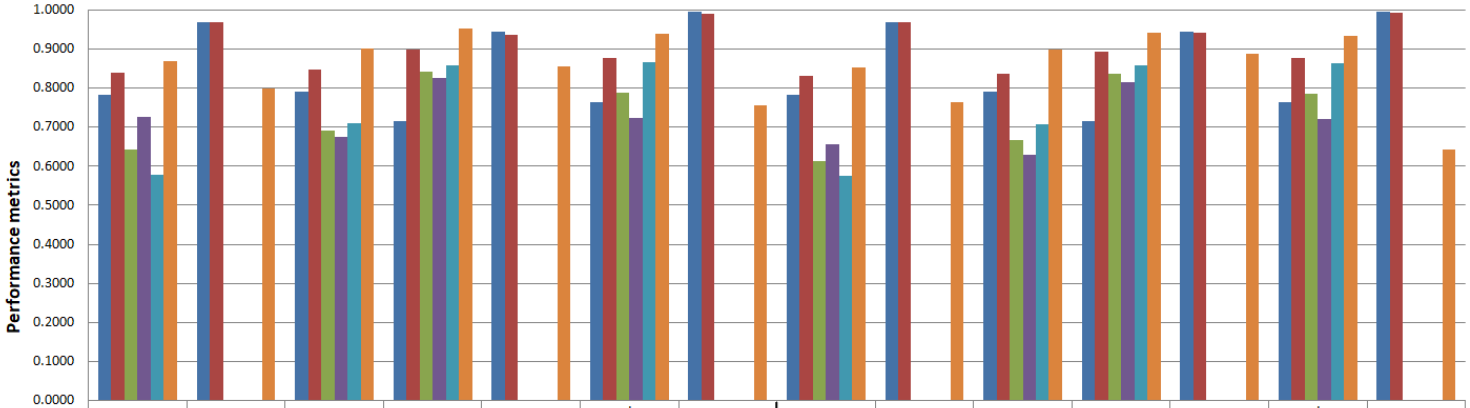
Fig. 5. Performance metrics (aggregated) for QoE classification for all models

In Fig. 6 is provided the one-by-one results for the classification of the 7 labels for the current and next time-steps. The AUC score and the frequency of the non-error value are given for all the labels, in addition to the metrics: accuracy, F1, precision and recovery. The results are obtained with Model 2.

It is important to note in Fig. 6 that for some labels the results are very good (chrominance

and color bleeding), for others are not bad (columns and ghost) and very bad for the rest (blur, blockness and black pixel). For these latter labels, the detector always assigns the most frequent value, thus making the accuracy almost identical to the frequency of the most frequent value (non-error value). This happens for the labels with the most unbalanced values.

The F1 and AUC values are ranked in a similar order (Fig.6). It is interesting to note that the performance metrics get worse at the next time-step vs. the current time-step, as expected, but the reduction in performance is quite small, which is good news as it confirms the implicit initial hypothesis that the prediction of QoE was possible.



	CURRENT TIME-STEP							NEXT TIME-STEP						
	columns	blur	ghost	chrominance	blockness	color bleeding	black pixel	columns	blur	ghost	chrominance	blockness	color bleeding	black pixel
Frequency	0.7835	0.9676	0.7909	0.7157	0.9440	0.7647	0.9943	0.7835	0.9676	0.7909	0.7157	0.9440	0.7647	0.9943
Accuracy	0.8397	0.9679	0.8462	0.8974	0.9359	0.8782	0.9904	0.8301	0.9679	0.8365	0.8942	0.9423	0.8782	0.9936
F1	0.6429	0	0.6923	0.8416	0	0.7889	0	0.6131	0	0.6667	0.8358	0	0.7865	0
Precision	0.7258	0	0.6750	0.8252	0	0.7245	0	0.6563	0	0.6296	0.8155	0	0.7216	0
Recall	0.5769	0.0000	0.7105	0.8586	0.0000	0.8659	0.0000	0.5753	0.0000	0.7083	0.8571	0.0000	0.8642	0.0000
AUC	0.8689	0.7987	0.9004	0.9520	0.8550	0.9379	0.7551	0.8538	0.7629	0.8998	0.9427	0.8878	0.9339	0.6435

Fig. 6. Performance metrics (one-by-one) for QoE classification. Results obtained with Model 2.

In conclusion, the classifier achieves excellent prediction results for the non-extremely unbalanced labels. These results are more significant considering the small number of training samples available and the noisy nature of error detection (subjective component).

B. Impact of time-series length

Considering the small amount of data available, the length of the elementary flows used for training is an important parameter to be considered.

Increasing the number of samples used in each elementary flow reduces the number of training flows and imposes a tougher requirement in the amount of information needed to perform prediction.

Interestingly, the best performance metrics are obtained for three samples per elementary flow. The performance does not increase when increasing the number of samples per flow beyond three, implying that prediction is conditioned on the amount of previous information, but information too distant in time is not only less useful but indeed a problem. Similarly, reducing the number of samples below three also decreases the prediction performance.

V. CONCLUSION

This work shows that it is possible to apply new deep learning models whose origin focused mainly on the areas of video, audio and language processing for the prediction of QoE of transmitted videos. We have extended the work in [15], demonstrating that a CNN can be applied to a time-series of samples (formed by aggregated information from network packets) to predict QoE for video transmission. As far as we know, there is no previous application of a deep learning model to QoE prediction, being this work the first contribution to this area.

The proposed model can be integrated into a network management system to monitor network quality (as observed by the end-user), which is an essential part of a self-adapting network (e.g. SDN, edge computing...). The model is applicable to a real-time environment (in time-steps of 1-second) and is able to predict video QoE for current and near-future video transmissions.

The best proposed model includes a combination of CNN and RNN networks, being the CNN network the most critical piece. We have extended the study with the inclusion of a GP Classifier that, being a non-parametric model, could make full use of the scarce data available. This inclusion provides a slight improvement in the prediction results, but has to be considered in parallel with the increase in memory and processing time required.

In future work, we plan to investigate the application of generative models (e.g. variational autoencoders) to create synthetic data and explore one-shot learning advances.

ACKNOWLEDGMENTS

This work has been funded by the Ministerio de Economía y Competitividad del Gobierno de España and the Fondo de Desarrollo Regional (FEDER) within the project "Inteligencia distribuida para el control y adaptación de redes dinámicas definidas por software, Ref: TIN2014-57991-C3-2-P", and, also by the Ministerio de Economía y Competitividad in the Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento with the projects “Distribucion inteligente de servicios multimedia utilizando redes cognitivas adaptativas definidas por software, Ref: TIN2014-57991-C3-1-P" and “Red Cognitiva Definida por Software Para Optimizar y Securizar Tráfico de Internet de las Cosas con Informacion Critica”, Ref TIN2017-84802-C2-1-P.

REFERENCES

[1] J. Wang, J. Pan, and F. Esposito, “Elastic urban video surveillance system using edge computing,” *Proceedings of the Workshop on Smart Internet of Things (SmartIoT '17).* ACM, New York, NY, USA, 2017, Article 7

[2] K. Bilal, A. Erbad, "Edge computing for interactive media and video streaming," *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, Valencia, Spain, May 8-11, 2017, pp. 68-73

[3] G. Ananthanarayanan et al., "Real-Time Video Analytics: The Killer App for Edge Computing," *Computer*, vol. 50, no. 10, 2017, pp. 58-67.

[4] J. Lloret et al., “A QoE management system to improve the IPTV network,” *International Journal of Communication Systems*, 24, 2011, pp. 118–138.

[5] M. Garcia et al., "A QoE Management System for Ubiquitous IPTV Devices," *2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, Sliema, 2009, pp. 147-152.
[6] Y. Chen, K. Wu and Q. Zhang, "From QoS to QoE: A Tutorial on Video Quality Assessment," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, 2015, pp. 1126-1165.
[7] V. Menkovski, G. Exarchakos and A. Liotta, "Machine Learning Approach for Quality of Experience Aware Networks," *2010 International Conference on Intelligent Networking and Collaborative Systems*, Thessaloniki, 2010, pp. 461-466.
[8] S. Aroussi, A. Mellouk, "Survey on machine learning-based QoE-QoS correlation models," *2014 International Conference on Computing, Management and Telecommunications (ComManTel)*, Da Nang, 2014, pp. 200-204.
[9] A. Balachandran et al., "Developing a predictive model of quality of experience for internet video," *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (SIGCOMM '13)*. ACM, New York, NY, USA, 2013, pp. 339-350.
[10] C. G. Bampis, A. C. Bovik, "Learning to Predict Streaming Video QoE: Distortions, Rebuffering and Memory," *arXiv*:1703.00633 [cs.MM], 2017.
[11] W. Rawat, Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," *Neural Computation*, vol. 29, no. 9, 2017, pp. 2352–2449.
[12] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," *arXiv*:1506.00019 [cs.LG], 2015.
[13] C. K. I. Williams, D. Barber, "Bayesian classification with Gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, 1998, pp. 1342-1351.
[14] S. Hochreiter, J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*," vol. 9, no. 8, 1997, pp. 1735–1780.
[15] M. Lopez-Martin et al., "Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things," *IEEE Access*, vol. 5, 2017, pp. 18042-18050.

BIOGRAPHIES

MANUEL LOPEZ-MARTIN [M'91, SM'12] (mlopezm@ieee.org) is a research associate and Ph.D. candidate at Universidad de Valladolid (Spain). His research activities include the application of machine learning models to data networks. He received his M.Sc. in Telecommunications Engineering in 1985 from UPM-Madrid and his M.Sc. in Computer Sciences in 2013 from UAM-Madrid. He has worked as data-scientist at Telefonica and has more than 25 years of experience in the development of IT software projects.

BELEN CARRO (belcar@tel.uva.es) received a Ph.D. degree in the field of broadband access networks from the Universidad de Valladolid, Spain, in 2001. She is Professor and Director of the Communications Systems and Networks (SRC) laboratory at Universidad de Valladolid, working as Research Manager in NGN communications and services, VoIP/QoS and machine learning. She has supervised a dozen Ph.D. students and has extensive research publications experience, as author, reviewer and editor.

JAIME LLORET [M'07, SM'10] (jlloret@dcom.upv.es) received his M.Sc. in Physics in 1997, his M.Sc. in Electronic Engineering in 2003 and his Ph.D. in telecommunication engineering in 2006. He is Associate Professor at the UPV and Chair of the Research Institute IGIC. He has authored more than 450 research papers and book chapters. He is EiC of the "Ad

Hoc and Sensor Wireless Networks” and "Networks Protocols and Algorithms". He is ACM Senior and IARIA Fellow.

SANTIAGO EGEA (santiago.egea@alumnos.uva.es) received a Telecommunication Engineering Degree from Polytechnic University of Cartagena, Murcia, Spain. Currently, He is a PhD student at University of Valladolid, Valladolid, Spain. He is member of the Communications Systems and Networks (SRC) lab. His research interests include Signal Processing and Machine Learning specifically applied to telecommunication networks.

ANTONIO SANCHEZ-ESGUEVILLAS [M’07, SM’07] (antoniojavier.sanchez@uva.es) received a Ph.D. degree in the field of QoS over IP networks from the University of Valladolid, Spain, in 2004. He has managed innovation at Telefonica, has been Adjunct Professor and Honorary Collaborator at the University of Valladolid, supervising several Ph.D. students. He has coordinated very large international R&D projects, has over 50 international publications and several patents. His current research interests include digital services and machine learning.

PAPER 5

Variational data generative model for intrusion detection

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

Dpto. TSyCeIT, ETSIT, Universidad de Valladolid, Paseo de Belén 15, Valladolid 47011, Spain ;

mlopezm@acm.org ; belcar@tel.uva.es ; antoniojavier.sanchez@uva.es;

The authors declare that there is no conflict of interest regarding the publication of this paper

Abstract

A Network Intrusion Detection System (NIDS) is a system which detects intrusive, malicious activities or policy violations in a host or hosts network. It is very important for any detection system the ability to access balanced and diversified data to train the system. Intrusion data rarely have these characteristics, since samples of network traffic are strongly biased to normal traffic, being difficult to access traffic associated with intrusion events.

Therefore, it is important to have a method to synthesize intrusion data with a probabilistic and behavioral structure similar to the original one. In this work we provide such a method.

Intrusion data have continuous and categorical features, with a strongly unbalanced distribution of intrusion labels. That is the reason why we generate synthetic samples conditioned to the distribution of labels. That is, from a particular set of labels, we generate training samples associated with that set of labels, replicating the probabilistic structure of the original data that comes from those labels.

We use a generative model based on a customized Variational Autoencoder (VAE), using the labels of the intrusion class as an additional input to the network. This modification provides an advantage, as we can readily generate new data using only the labels, without having to rely on training samples as canonical representatives for each label, which makes the generation process more reliable, less complex and faster. We show that the synthetic data are similar to the real data, and that the new synthesized data can be used to improve the performance scores of common machine learning classifiers.

Keywords: Intrusion detection system; Variational methods; Generative model; Neural Networks.

1. Introduction

The objective of a Network Intrusion Detection System (NIDS) is to automate the detection of policy and security violations and malicious activities against a host that is part of a network. As the importance and volume of data exchange increases, it is more relevant to increase the performance of NIDS. Most current NIDS are based on supervised machine learning models which require labeled data to perform model training.

It is fundamental for any detection system the possibility of accessing balanced and diversified data to train the system. Intrusion data rarely have these characteristics, as the network traffic samples are strongly biased to the normal type of traffic, being difficult to access traffic associated to anomalous intrusion events. Therefore, it would be very interesting to be able to synthesize intrusion data with a structure similar to the real data. In this way, we avoid investing significant resources in tasks such as obtaining additional intrusion data or manually simulating attacks.

The generation of synthetic data that resemble real data, being similar (in a probabilistic sense) but not identical, has long been an objective in the area of image processing [1, 2]. In this area the features used to train the models are all continuous (pixel intensities) and, additionally, it is relatively easy to appreciate if a generative model is working well, as we (humans) are quite good at identifying whether the images generated corresponds to the class of images we want. Similar works have been carried out more recently in the generation of text/sentences [3, 4]. In this case the features are all discrete, and we can appreciate directly, in a similar way, if the generated text corresponds to a particular topic.

In the intrusion detection area the generative data process has its own difficulties, due to several reasons: (1) the features used to identify an intrusion type (label) are both continuous and categorical, (2) the class labels are highly unbalanced, and (3) we cannot directly appreciate whether a new synthetic sample of a particular class really correspond to that class (intrusion label). The first and second imply that we need to synthesize, at the same time, discrete and continuous features, each having its own problematic. The third requires developing alternative techniques to show the similarity of original and synthetic data. We need to identify if two populations of samples (real and synthetic) belong to the same class. Taking into account that the samples represent multivariate high-dimensional vectors, with continuous and discrete values, complex joint probability distribution and with non-Gaussian marginals (or another easily parameterized distribution), it is very difficult to apply methods based on information theory (e.g. KL divergence) or multivariate extensions of goodness-of-fit tests to identify the similarity between the probability distributions of the two populations (Section 4.1). That is why we have applied other alternative and original approaches to evaluate similarity: (1) Extended histograms of the original and synthesized features, and, (2) demonstrate that classification results are similar when using either the original or synthesized data as training or testing data for several classifiers (e.g. Random Forest, Multinomial Logistic Regression…).

The problem mentioned above is not found when generating synthetic images or text, since, as already discussed, we can discriminate samples that belong to specific objects (images) or topics (text). In the case of samples related to intrusion detection, there is not such good discriminator. To appreciate the complex and unclear relationship between the distributions of values of a high-dimensional sample with the label associated to that sample, we could consider the difficulties imposed by adversarial examples [5] to a neural network and how the addition of small perturbations to images can mislead a perfectly tuned neural network, resulting in misclassified images, even when these perturbations do not affect the discrimination capacity of humans.

In this work we provide a method to generate data of similar probabilistic structure to intrusion detection data, having both continuous and categorical features and being strongly unbalanced to some of their associated labels. We generate the data conditioned to the specific class (label) to which we want the data to belong. That is, from a particular set of labels we generate training samples associated to that set of labels, reflecting real data that comes from those labels.

We call the method Variational Generative Model (VGM). We use a generative model based on a Conditional Variational Autoencoder (VAE), using the intrusion class labels as input. This modification provides an advantage, as we can readily generate new data using only the labels, without having to rely on specific training samples that represent or are associated to specific labels. Furthermore, the new synthesized data can be used as new additional training data to improve classification results for common machine learning classifiers. These results also confirm that the synthesized data have similar structure to the original but not been identical which allows to improve the performance of a classifier.

The problem presented here can be considered similar to the one faced by classification with an imbalanced dataset, which is mainly addressed with four strategies [6, 7, 8]: resampling, cost-sensitive, algorithmic and ensemble. To compare VGM with equivalent approaches, we focus on resampling. Resampling can be achieved creating new minority class samples (over-sampling) or reducing the number of majority class samples (under-sampling). An effective way to perform over-sampling is by creating new synthetic data that resembles the original data. The state-of-the-art (SOTA) algorithms in synthetic over-sampling are based on SMOTE [9] and its numerous variants [10, 11]; being its main idea to create new samples close in 'distance' to existing samples that belongs to some specific minority class. The different variants consider alternative approaches to calculate the distance function and the proximity to majority class samples (borderline). To avoid possible over-fitting due to synthetic data, there is the possibility to combine over-sampling and under-sampling methods [12, 13]. Another interesting method is ADASYN[14], which is similar to SMOTE, but giving more weight to samples that are harder to learn (closer to other majority class samples). Finally, there is the possibility to perform ensemble sampling with methods similar to EasyEnsemble [15].

Our method (VGM) is a generative method to synthesize new data belonging to any class label. The main difference between VGM and SMOTE (and its variants) is that VGM is based in a latent probability distribution learned from data, instead of being based in a predefined 'distance' function. VGM does not need to assume any 'distance' function, or to impose rules on the importance of proximity to majority class samples, which would be additional hyper-parameters to explore.

We provide a comparison of the proposed model with seven SOTA synthetic data generation algorithms (SMOTE, ADASYN…), showing that synthetic data generated by VGM provides better performance metrics (average accuracy and F1) when several common classifiers are trained with this data instead of data from other alternative generation algorithms. To train the VGM model, we need original data from a well-known intrusion detection dataset, for which we have chosen the NSL-KDD data set [16]. We have explored several architectures for VGM, considering different number of layers, nodes, regularization, loss functions and probability distribution for the output layer. We present the different options and the results obtained.

As a summary, the contributions of this paper are: (1) It is the first application of a conditional variational autoencoder to generate synthetic data in the intrusion detection field. (2) We present original methods to show similarity of real and synthetic data. (3) VGM provides more useful synthetic samples than comparable SOTA over-sampling algorithms, corroborated by better performance (accuracy, F1) produced by various classifiers when using synthetic data generated by VGM.

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

2. Related works

As far as we know, there is no previous application of a variational generative model based on neural networks to generate data of similar probabilistic structure to intrusion detection data. Therefore, the work presented in this paper is original in essence. There are a number of works for the application of variational generative models to generate images [1, 2, 17] and text [3, 4], but there is none in the area of intrusion detection.

There is no work, similar to the present one, generating both continuous and categorical features. In [18] is presented a model that handles a discrete distribution for the "latent" layer, but it is applied to images with continuous features. In [19] the authors provide a solution using a VAE in the intrusion detection field, but it is used exclusively to implement a classifier, not to generate synthetic data according with the intrusion class as in the present work

There is a vast number of works applying classification algorithms to NIDS [20, 21]. This paper is not related specifically with any classification technique, but we will show (Section 4.2) that the synthetic data generated with our model improves results obtained with different classifiers. Therefore, it is interesting to provide a summary of results on classification using the NSL-KDD dataset, which will help to put into perspective the results presented in this work. It is important to mention that comparison of results in this field is difficult due to: (1) diversity of reported performance metrics; (2) the aggregation of classification labels in different sets (e.g. 23 labels can be grouped hierarchically in different subsets or categories: 23, 5 or 2 final labels) making it difficult to compare results for different subsets; (3) reporting results on unclear test datasets. This last point is important to mention, because for example, for the NSL-KDD dataset, 16.6% of samples in the test dataset correspond to labels not present at the training dataset. This is an important property of this dataset and creates an additional difficulty to the classifier. These difficulties are shown in detail in [16].

Classification results for NSL-KDD are provided in several works. In [22] is achieved an accuracy of 79.9% for test data, for the 5-labels intrusion scenario. In [23] they provide, for the 2-labels scenario a recall of 75.49% on test data. Authors in [20] explain the reasons to create the NSL-KDD data set, providing results for several algorithms, being 82.02% the best accuracy reported when using the full NSL_KDD dataset for training and testing, for the 2-labels scenario.

There is large literature related to algorithms dealing with imbalanced datasets [6, 7, 8], and in particular presenting algorithms for synthetic over-sampling [9, 10, 11], adaptive over-sampling [14], over-sampling followed by under-sampling [12], ensemble sampling [15] and specific combinations of methods [13].

3. Work description

In this section we present the dataset used for this work, a description of the variational method employed and details on different variants of the method.

3.1. Selected dataset

We have chosen the NSL-KDD [16] dataset as our reference dataset. NSL-KDD is an

enhanced version of the original KDD-99 dataset, solving the problem of redundant records present in KDD-99. We consider that this dataset is useful for this work, as we are mainly interested in generation of synthetic data, for which NSDL-KDD provides a sufficient number of samples. Additionally, the distribution of samples among intrusion classes (labels) is quite unbalanced and provides enough variability between training and test data to challenge any method that tries to reproduce the structure of the data.

The NSL-KDD dataset provides 125973 training samples and 22544 test samples, with 41 features, being 38 continuous and 3 categorical (discrete valued). Each training sample has a label output from 23 possible labels (normal plus 22 labels associated to different types of anomaly). The test data has the same number of features (41) and output labels from 38 possible values. That means that the test data has anomalies not presented at training time. The 23 training and 38 testing labels have 21 labels in common; 2 labels only appear in training and 17 labels are unique to the testing data.

Around 16% of the samples in the test dataset correspond to labels unique to the test dataset, and which were not present at training time. The existence of new labels at testing introduces an additional challenge to the learning methods, which is important to verify the robustness of the classifiers, but not for the purpose of this study, which is to synthesize samples associated to existing labels. Therefore, it seems more practical and useful to aggregate labels by categories. As presented in [16], the original labels are associated to 5 categories: NORMAL, PROBE, R2L, U2R and DoS, with the latter four corresponding to an anomaly. The meaning of the 5 categories is as follows:

- NORMAL: There is no attack
- Denial of Service (DoS): The intention of these attacks is to interrupt some service.
- PROBE: They intend to gain information about the target host.
- User to Root (U2R): U2R attacks try to obtain root access to the system.
- Remote to Local (R2L): Unauthorized access from a remote machine.

For this work we have used these 5 categories as the labels driving our data generation model.

We have performed an additional data transformation: scaling all NSL-KDD continuous features to the range [0,1] and one-hot encoding all categorical features. This provides a final dataset with 116 features: 32 continuous and 84 with values in {0,1} associated to the three one-hot encoded categorical features.

It is important to note that the 3 categorical features: protocol, flag and service have respectively 3, 11 and 70 distinct values. We will show later the accuracy obtained when synthesizing these features (having as reference the original ones), and how the different number of values impacts on the results.

Working with the NSL-KDD dataset, we provide all results using the full training dataset of 125973 samples and the full test dataset of 22544 samples. It is also important to mention that we do not use a previously customized training or test datasets, neither a subset of them, what may provide better alleged results but being less objective and also missing the point to have a common reference to compare.

3.2. Method explained

In Figure 1 we present a diagram comparing VGM and VAE architectures. In a VAE architecture [1] we model the internal structure of data with an initial neural network (encoder) that approximates the parameters of a probability distribution. This probability distribution is used to draw samples that are the input to a second neural network (decoder) that approximates the parameters of a second probability distribution from which the samples drawn are the final

output of the VAE.

To train a VAE we use the same data for input and output, trying to implement the identity function but with an intermediate layer with a lower dimension. Contrary to an Autoencoder [24] that is used as a dimensionality reduction mechanism, with a VAE we will not obtain a deterministic lower dimension representation of the data, but a set of parameters that define an associated set of probability distributions that represent the data. In other words, instead of mapping an input sample in $\mathcal{R}^m$ to a deterministic output (latent variable) in $\mathcal{R}^n$, where usually $n < m$, what we do in a VAE is to map samples in $\mathcal{R}^m$ to n probability distributions. The samples drawn from these n probability distributions form a new stochastic feature vector in $\mathcal{R}^n$. The new latent variable is not deterministic but stochastic (variable $\boldsymbol{Z}$ in Figure 1).

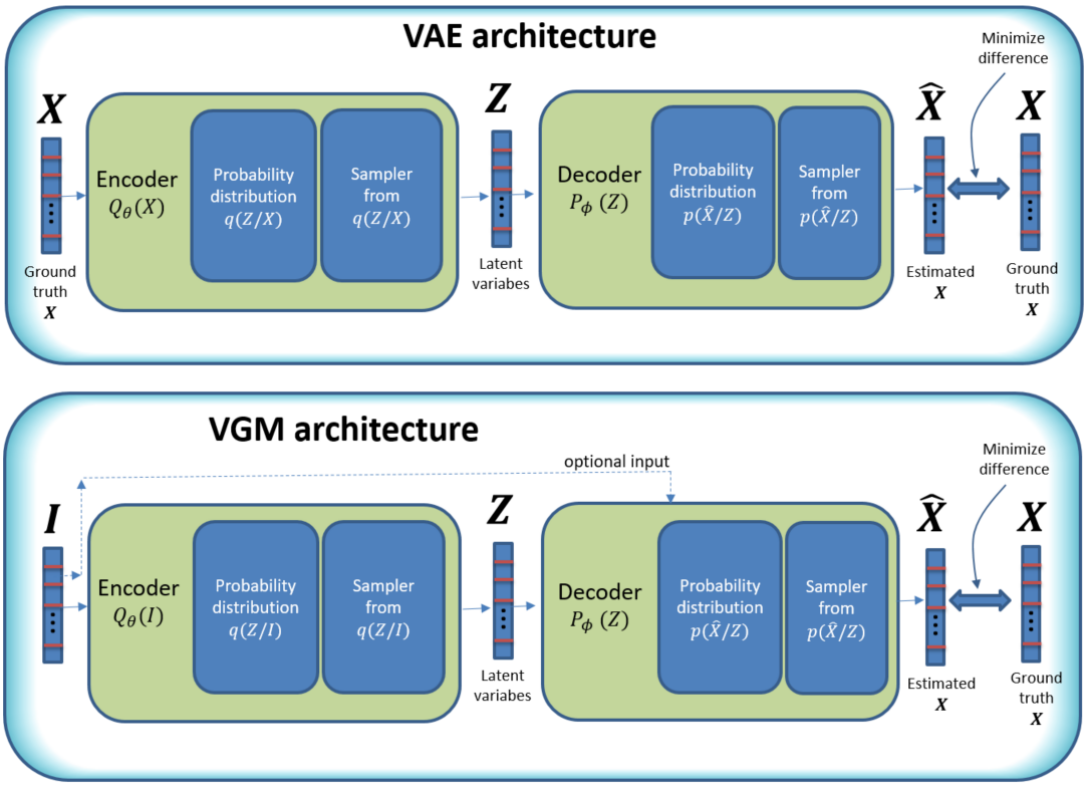


Figure 1.. Comparison of VGM with a typical VAE architecture

The interest of a VAE is that, after the model is trained, it is sufficient to sample the intermediate layer (latent variables) to generate new data that resembles the data used for training.

The left part of the diagram of VAE in Figure 1 corresponds to the encoder which depends on the input data ($\boldsymbol{X}$) and some model parameters ($\boldsymbol{\theta}$) and produces the "latent" probability distributions ($\boldsymbol{q(Z/X)}$) and stochastic latent variable ($\boldsymbol{Z}$). To the right of the diagram we have the decoder which depends both on the latent variable ($\boldsymbol{Z}$) and some other model parameters ($\boldsymbol{\phi}$) and produces a new set of output probability distributions ($\boldsymbol{p(\hat{X}/Z)}$) from which we sample the final output ($\boldsymbol{\hat{X}}$).

The model parameters: $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, are used as a brief way to represent the architecture and weights of the neural network used. These parameters are tuned as part of the VAE training process and are considered constant later on.

The probability distributions $\boldsymbol{p(\hat{X}/Z)}$ and $\boldsymbol{q(Z/X)}$ are conditional probability distributions, and they are parameterized, which means that they are completely defined by a set of

parameters (e.g. the mean and variance for a normal distribution).

The final objective is to produce an output $\widehat{\boldsymbol{X}}$ with a minimum difference to the input $\boldsymbol{X}$. There are different ways to achieve this objective, one is to use sampling methods as Markov Chain Monte Carlo (MCMC), but the path taken by VAE is different. VAE uses a variational approach that tries to maximize the log likelihood of X by maximizing a quantity known as the Evidence Lower Bound (ELBO) [1]. The ELBO is formed by two parts: (1) a measure of the distance between the probability distribution $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{X})$ and a reference probability distribution of the same nature (actually a prior distribution for Z), where the distance usually employed is the Kullback-Leibler (KL) divergence, and (2) the log likelihood of $\boldsymbol{p}(\boldsymbol{X})$ under the probability distribution $\boldsymbol{p}(\widehat{\boldsymbol{X}}/\boldsymbol{Z})$, which is the probability to obtain the desired data ($\boldsymbol{X}$) with the probability distribution that produces $\widehat{\boldsymbol{X}}$.

Using the ELBO, we reduce the problem to an optimization problem based on a maximization of the ELBO, which allows using neural networks with stochastic gradient descent (SGD) as the optimizer. The only problem remains on how to incorporate the sampling process, required by the model, with the way SGD operates. To do this, the innovation of VAE is to use what is called the "reparameterization trick" [1]. Using this trick, all the variables involved are connected through differentiable layers on which SGD can operate.

Based on the VAE model, our proposed method (VGM) is similar to a VAE but instead of using the same vector of features for the input and output of the network, we add more flexibility allowing to have a different input to the network and to add an optional additional input on the decoder block (Figure 1, lower diagram). We represent this generic input in Figure 1 with the letter $\boldsymbol{I}$. The input $\boldsymbol{I}$ can be instantiated in two possible ways: as the vector of sample features, as in VAE, or as the vector of labels associated to the samples. That is, the input $\boldsymbol{I}$ can be either X or L, where L is the vector of labels.

In the VGM architecture, in case we use the vector of features as input to the encoder (as in VAE) then we will employ an additional input to the decoder formed by the vector of labels. In this way we will always have the vector of labels as an input to the network, either as input to the encoder or decoder blocks.

To use the labels as input of the generative process is an important difference as it allows generating new synthesized samples using exclusively the labels assigned to these samples. As already pointed out, for intrusion detection data, the generative data process is more difficult, as the features are both continuous and categorical, and we cannot appreciate directly if the synthesized data samples have features with a similar structure to the original ones, that is the reason why using directly the labels is important to be sure we are using meaningful information to characterize the generated samples.

In Section 3.3 we will present different variants to the generic architecture for VGM shown in Figure 1. In Figure 2 we present the elements of the loss function to be minimized by SGD for the VGM model. We can see that, as mentioned before, the loss function is made up of two parts: a KL divergence and a log likelihood part. The second part takes into account how probable is to generate $\mathbf{X}$ by using the distribution $\boldsymbol{p}(\widehat{\mathbf{X}}/\mathbf{Z})$, that is, it is a distance between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$. The KL divergence part can be understood as a distance between the distribution $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{I})$ and a prior distribution for $\boldsymbol{Z}$, that we identify as $q_{reference}$ in Figure 2. By minimizing this distance, we are really avoiding that $\boldsymbol{q}(\boldsymbol{Z}/\boldsymbol{I})$ departs too much from its prior, acting finally as a regularization term. The nice feature about this regularization term is that it is automatically adjusted, and it is not necessary to perform cross-validation to adjust a hyper-parameter associated to the regularization, as it is needed in other models (e.g. ridge regression, soft-margin support vector machines…).

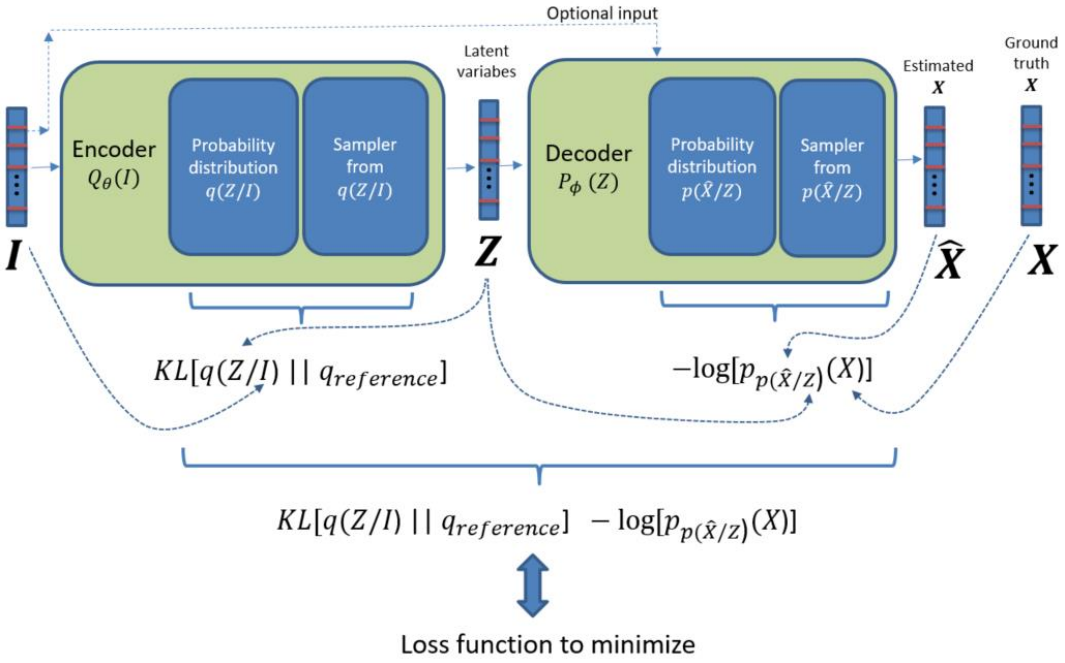


Figure 2. Details on the loss function elements for the VGM model.

3.3. Variants of the model

We have explored three options for the general VGM model discussed in the previous section.

3.3.1. Option A. Model with Gaussian and Bernoulli distributions

In this option we have the vector of labels as input to the network and the vector of features as output. Figure 3 presents option A.

We use a multivariate Gaussian as the distribution for $\boldsymbol{q(Z/L)}$, with a mean $\boldsymbol{\mu(L)}$ and a diagonal covariance matrix: $\boldsymbol{\Sigma(L)} \rightarrow \boldsymbol{\sigma_i^2(L)}$, with different values along the diagonal. We have a standard normal $\boldsymbol{N(0, I)}$ as the prior distribution for Z.

For the distribution $\mathbf{p(\hat{X}/Z)}$ we use a multivariate Bernoulli distribution. The nice property about the Bernoulli distribution is that we do not require doing sampling on it, as the output parameter that characterizes the distribution is the mean that is the same as the probability of success. Then, in this case, the output of the last layer is taken as our final output $\boldsymbol{\hat{X}}$.

The selection of distributions for $\boldsymbol{q(Z/L)}$ and $\mathbf{p(\hat{X}/Z)}$ is similar to the distributions selected in [1], since they are simple and provide good results.

In Figure 3 we show also (in squared boxes) the elements of the loss function to be minimized for this option.

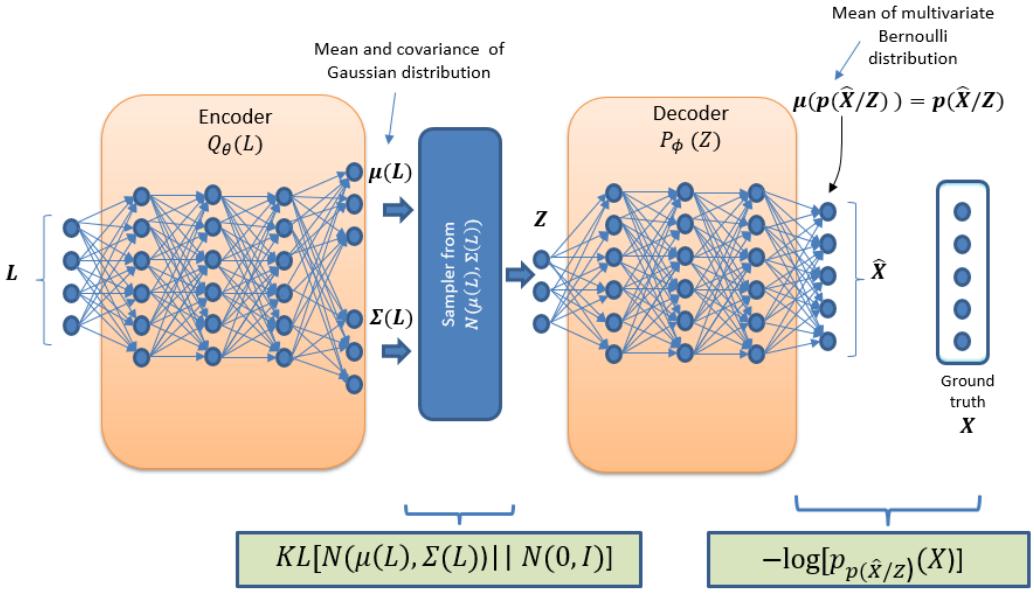


Figure 3. Option A. VGM model with Gaussian and Bernoulli distributions.

3.3.2. Option B. Model with Gaussian and Bernoulli distributions plus RMSE for continuous features

Option B is presented in Figure 4. This option is similar to option A, but we have divided the output layer according to the separation between discrete and continuous features. We treat the discrete features as in Option A, and for the continuous features we change the loss function to the root mean square error (RMSE) between original and generated continuous features, instead of the log-likelihood, as in option A.

We have tried this change in the loss function to see whether we could obtain an improvement by separating the behavior of continuous and discrete features. The change can be also justified by considering that minimizing an RMSE loss function is equivalent to minimizing the negative log-likelihood of an implicit Gaussian distribution for the last decoder layer (in accordance with ELBO theory).

In Figure 4, in squared boxes are the elements of the loss function to be minimized for this option.

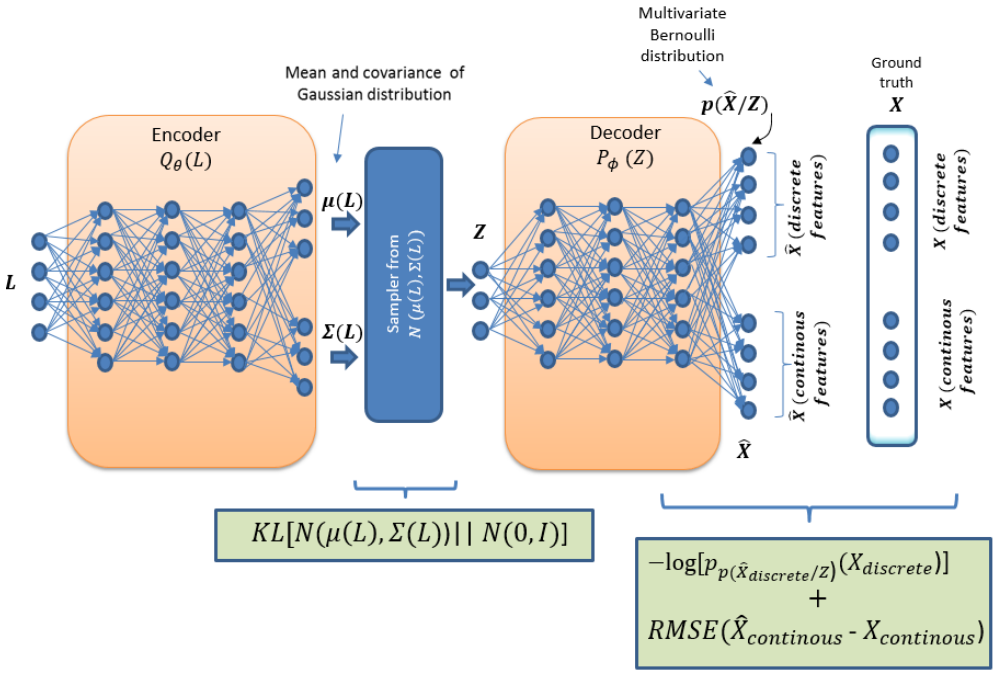


Figure 4.
Option B. VGM model with Gaussian and Bernoulli distributions plus RMSE for continuous features.

3.3.3. Option C. Model based on Conditional VAE with Gaussian and Bernoulli distributions

The last option is presented in Figure 5. This option is similar to option A, but with an important difference. Instead of using the labels as the input for the encoder we use it as an additional input to the decoder. We leave the features vector (***X***) as input to the encoder. The architecture is similar to a normal VAE except for the inclusion of the label vector as a supplementary input to the decoder.

In order to get the label vector inside the decoder network we just concatenate it with the values of the first layer of the decoder block (Figure 5). The one-hot encoded label vector has a length of 5 that is too short to have a significative impact in the output, which is why we replicate the label vector 40 times before we concatenate it with the first decoder layer. Then, what is represented in Figure 5 as the vector L is really a vector of length 200 (the label vector replicated 40 times). The number of times we replicate the label vector can be considered as a hyperparameter of the model: this number modifies the results but not in a very significant way; it can be considered as a part of the fine tuning of the model.

In Figure 5, in squared boxes are the elements of the loss function to be minimized for this option.

This option has produced the best results.

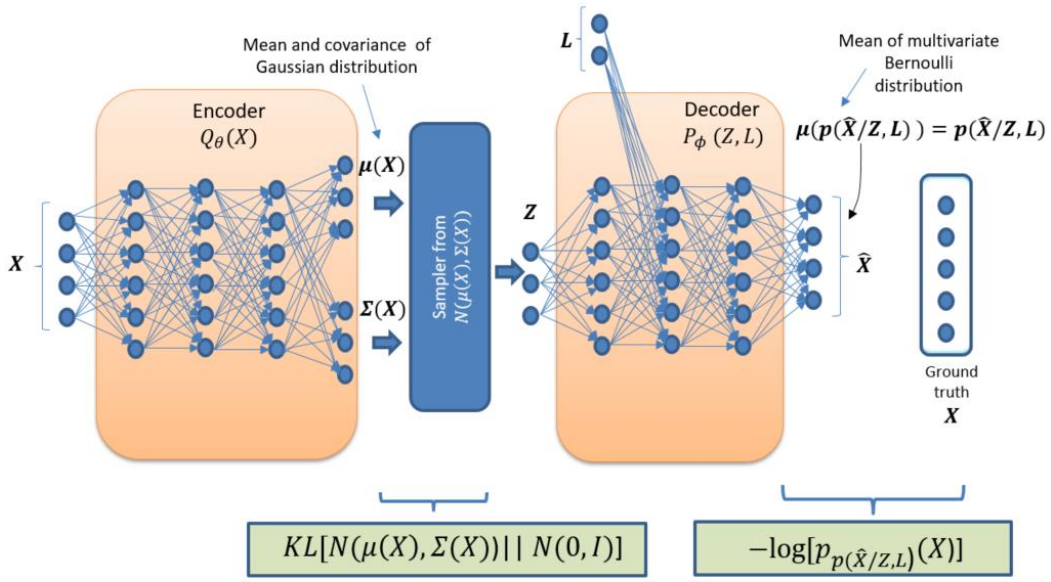


Figure 5. Option C. VGM model including the labels in the decoder with Gaussian and Bernoulli distributions. Training phase.

The process of training and data generation is different for this model when compared with previous models. In previous models, both the training and generation phases are done with the labels as input and the feature vectors as outputs, using the encoder and decoder blocks for the two phases.

For this model, after the training phase is done, we will employ only the decoder block of the trained model to generate new samples (Figure 6). To achieve that, we will provide two inputs to the decoder block: a vector of selected labels and a vector of random values sampled from a standard normal distribution with zero mean and unit variance. The output of the decoder block will be the desired generated samples. These samples are constructed using the probability distributions of the features associated to the labels given as input.

All the models presented, for all options, have the same objective: to be able to generate new data relying exclusively on the labels at the generation phase. This objective is also maintained in this option, because at generation phase we need only the labels and a vector of standard normal random values which are independent of everything else.

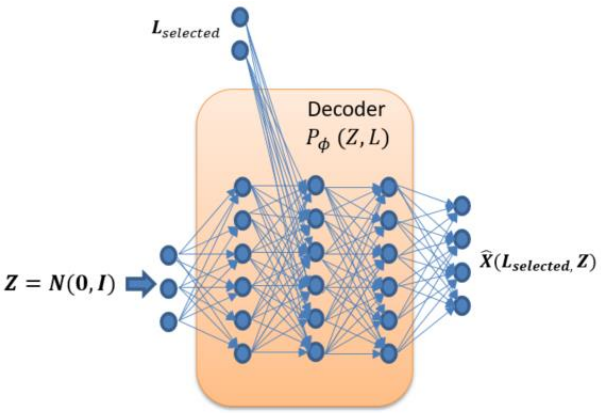


Figure 6 Option C. VGM model including the labels in the decoder with Gaussian and Bernoulli distributions. Generation phase.

4. Results

The objective of this section is first to prove that the synthetic data is similar but not identical to the original data (Section 4.1), and, this similarity is maintained when the data is conditionally partitioned by its class label. And, secondly, to show that the new synthetic data can be used as new training data, improving the results obtained with several prediction algorithms (Section 4.2)

4.1. Structure of generated data

In this section we will show that the synthetic generated data have similar probabilistic structure to the original data. Verifying this similarity is a hard problem since it involves comparing the probability distributions of multivariate vectors (116 features) with non-Gaussian marginals (discrete and continuous features) and complex joint probability distributions. The challenge is twofold: obtain the joint probability distributions and compare them. Methods based on information theory (e.g. Kullback-Leibler (KL) divergence) require an estimate of joint probabilities that is very difficult for high-dimensional variables [25], and many of them are not practically applicable for multivariate distributions (e.g. mutual information and KL divergence) [26]. Other methods based in multivariate extensions of goodness-of-fit tests are also difficult to apply considering the high dimensionality and non-Gaussian marginal distributions [27, 28, 29]

Considering the difficulties mentioned above, we have developed several approaches to verify the similarity: (1) extended histograms of the original and synthesized features; and (2) classification results obtained from the application of original and synthesized data to several classification algorithms.

Considering the first approach, Figure 7 presents extended histograms for the original NSL-KDD training dataset (upper diagram) and a synthesized dataset created with the same labels as the original one (lower diagram). To visualize the data in Figure 7, we use [30] which makes possible to visualize and compare the distributions of large datasets. The columns of the diagrams correspond to features. The rightmost 4 columns are the categorical variables, respectively: protocol (3 values), service (70 values), flag (11 values) and label (5 values). All features values are ordered in accordance with the alphabetical order of the label. The rows are divided in 100 slots associated to 100 bins where the continuous features have been mapped. The colors in the slots represent where the mean value is for that slot, with a different color to show the dispersion (similar to a box-plot)

We can observe, in Figure 7, that both diagrams present a similar distribution over the features. The intention of the diagram is to show the general similarity of distributions, providing an overall impression of similarity when comparing feature to feature from original and synthetic data. This is the reason why we do not give the names of the features in the diagram, since we are not interested here in a comparison of particular features.

It is important to note that the synthesized data corresponds to data generated from a forward pass of the model (Option C, Section 3.3.3), and each time we generate a new set of synthesized data this dataset will be different, due to the stochastic nature of the layer of latent variables.

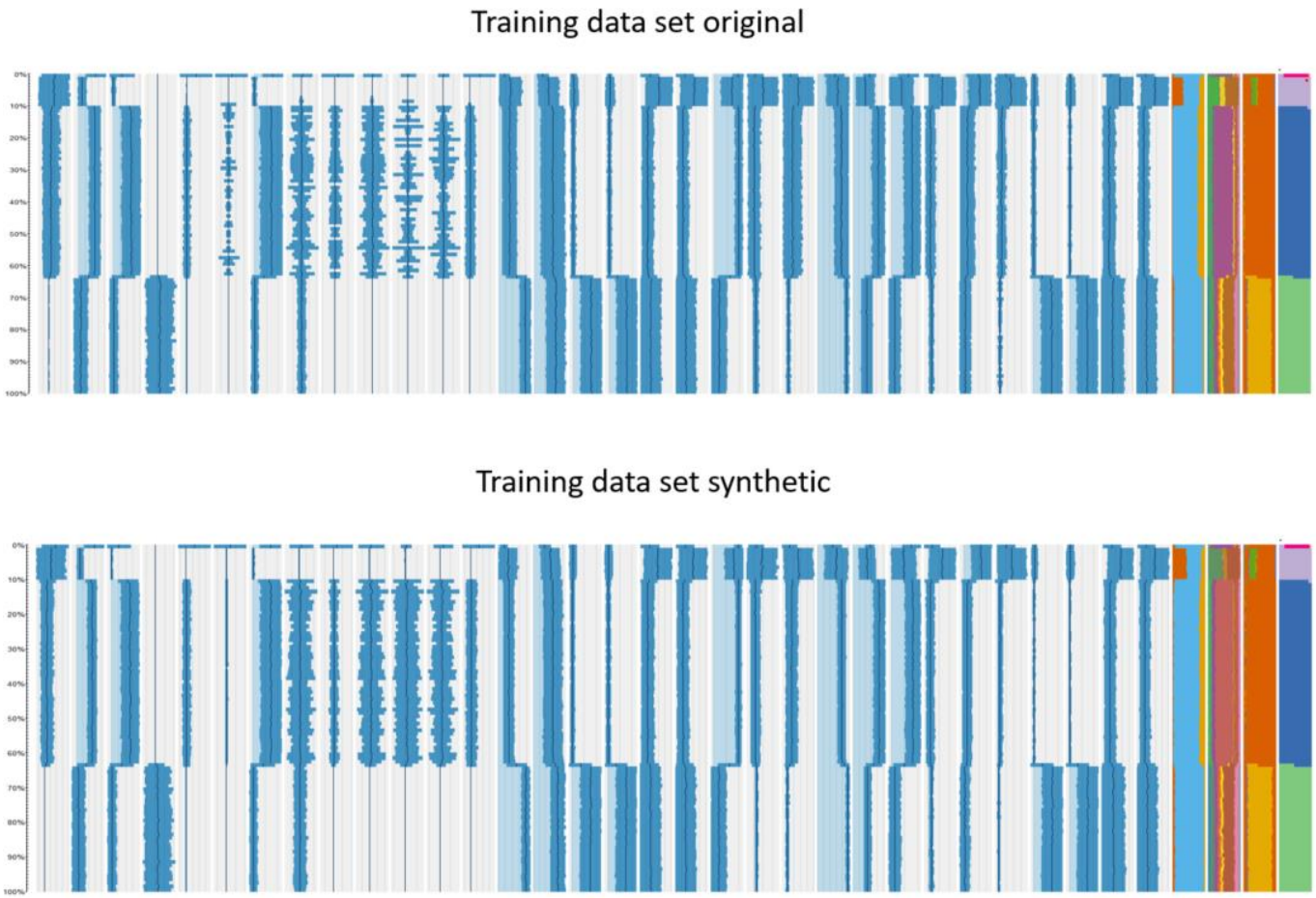


Figure 7. Extended histogram for original NSL-KDD training dataset and a synthesized one generated from the same labels as the original.

Regarding the second approach to check the similarity between original and synthesized features, in Table 1 we present the accuracy obtained in reproducing the three discrete features when using the NSL-KDD Training and Test dataset as original data. In each case, we compare the accuracy between the original dataset and a synthesized dataset composed of the same labels as the original. We see that the accuracy depends on the number of different values of the feature, and, in general, it is quite high. Option C offers the best results, providing a reconstruction accuracy for discrete features that is greater than 90% for most features. We can see that even when the much smaller NSL-KDD test dataset is used to train VGM, we still get very good accuracy results (the three columns on the right in Table 1). The values in Table 1 are color-coded; where the greenest is better and the redder is worse (comparison of values is applied column-wise). We base our definition of accuracy in the usually accepted one [20].

	Accuracy for discrete features, results with NSL-KDD Training dataset			Accuracy for discrete features, results with NSL-KDD Test dataset		
	flags	protocol	service	flags	protocol	service
Option A	0.82711	0.81517	0.46315	0.65822	0.83747	0.44451
Option B	0.81456	0.81517	0.44973	0.65277	0.83747	0.43320
Option C	0.99371	0.99660	0.88315	0.90046	0.98603	0.74676

Table 1. Accuracy when reproducing discrete features

We can see in Table 1 that the results differ for the column: flags. This is due to the fact that NSL-KDD has differences between the training and test datasets, as presented in Section 3.1.

This difference provides an additional challenge to the data generation process.

Following the same strategy, in Table 2, we present the classification results applying original and synthesized data to several classifiers. We have tested four different classifiers: Random Forest, Logistic Regression, Linear SVM and Multilayer Perceptron (MLP). In all cases, we used the NSL-KDD Training dataset as real data and several synthesized datasets created from a similar label's distribution, as synthetic data.

The objective here is to show that we have similar accuracies when doing prediction with either the original or the synthetic datasets, that is, referring to Table 2, the objective is to have similar values on columns with the same color for each particular option and classifier (see Table 2). We see that this is the case for most tests with all options; the results are quite similar when using either the original or synthesized data as training or prediction data, what provides additional arguments on their similarity. It is also interesting to note the poor results of almost all the options when using Linear SVM and training with synthetic data; the only option that behaves well in this case is Option C, which performs well and robustly in all tests. Option C presents the best results in Table 2, since the accuracy values remain very similar for each classifier.

The classifiers are set-up with their defaults parameters (no tuning), since we are not using the classifiers to show best performance, but to show if they provide similar performance when using the original and generated data, and we do not want to modify the results by tuning the parameters.

	Accuracy: Random Forest				Accuracy: Linear SVM			
Training with....	Real dataset	Real dataset	Synthetic dataset	Synthetic dataset	Real dataset	Real dataset	Synthetic dataset	Synthetic dataset
Prediction with....	Synthetic dataset	Real dataset	Real dataset	Synthetic dataset	Synthetic dataset	Real dataset	Real dataset	Synthetic dataset
Option A	0.98497	0.99986	0.85887	0.85964	0.97502	0.98433	0.20608	0.97613
Option B	0.93022	0.99986	0.86099	0.97973	0.93713	0.98433	0.09812	0.93495
Option C	0.98400	0.99986	0.94993	1.00000	0.98790	0.98433	0.93729	0.99963

	Accuracy: Logistic Regression				Accuracy: MLP			
Training with....	Real dataset	Real dataset	Synthetic dataset	Synthetic dataset	Real dataset	Real dataset	Synthetic dataset	Synthetic dataset
Prediction with....	Synthetic dataset	Real dataset	Real dataset	Synthetic dataset	Synthetic dataset	Real dataset	Real dataset	Synthetic dataset
Option A	0.98509	0.99317	0.80487	0.98516	0.97419	0.99904	0.83356	0.97669
Option B	0.94000	0.99317	0.69931	0.93935	0.93632	0.99904	0.87717	0.94002
Option C	0.99300	0.99317	0.95297	0.99974	0.99753	0.99904	0.96736	0.99998

Table 2. Prediction accuracy with different classifiers and datasets (original and synthetic)

To expand the data provided in Table 2, we present, in Table 3, the prediction details using a contingency table for the 5 predicted labels, when we use Option C with MLP as a classifier. We can see that the predicted and actual percentages of labels are very similar for the three most frequent labels, the least frequent being the most prone to errors, as expected.

		Training with real & Prediction with synthetic						
		Prediction						
		DOS	NORMAL	PROBE	R2L	U2R	Total	%
Ground Truth	DOS	45712	180	35	0	0	45927	36.46%
	NORMAL	78	67074	59	131	1	67343	53.46%
	PROBE	0	314	11321	21	0	11656	9.25%
	R2L	22	105	0	868	0	995	0.79%
	U2R	1	8	3	18	22	52	0.04%
	Total	45813	67681	11418	1038	23	125973	100%
	%	36.37%	53.73%	9.06%	0.82%	0.02%	100%	

		Training with real & Prediction with real						
		Prediction						
		DOS	NORMAL	PROBE	R2L	U2R	Total	%
Ground Truth	DOS	45927	0	0	0	0	45927	36.46%
	NORMAL	9	67299	11	22	2	67343	53.46%
	PROBE	2	29	11625	0	0	11656	9.25%
	R2L	0	33	0	962	0	995	0.79%
	U2R	0	13	0	0	39	52	0.04%
	Total	45938	67374	11636	984	41	125973	100%
	%	36.47%	53.48%	9.24%	0.78%	0.03%	100%	

		Training with synthetic & Prediction with real						
		Prediction						
		DOS	NORMAL	PROBE	R2L	U2R	Total	%
Ground Truth	DOS	45751	120	51	5	0	45927	36.46%
	NORMAL	622	65002	446	1186	87	67343	53.46%
	PROBE	22	436	11181	16	1	11656	9.25%
	R2L	2	47	10	912	24	995	0.79%
	U2R	0	12	0	9	31	52	0.04%
	Total	46397	65617	11688	2128	143	125973	100%
	%	36.83%	52.09%	9.28%	1.69%	0.11%	100%	

		Training with synthetic & Prediction with synthetic						
		Prediction						
		DOS	NORMAL	PROBE	R2L	U2R	Total	%
Ground Truth	DOS	45927	0	0	0	0	45927	36.46%
	NORMAL	0	67343	0	0	0	67343	53.46%
	PROBE	0	1	11655	0	0	11656	9.25%
	R2L	0	31	0	964	0	995	0.79%
	U2R	1	1	0	0	50	52	0.04%
	Total	45928	67376	11655	964	50	125973	100%
	%	36.46%	53.48%	9.25%	0.77%	0.04%	100%	

Table 3. Contingency table for predictions when employing different training and test data sets.

Finally, to prove that synthetic and original data are similar, but not identical, we subtract (element-wise) the original and synthetic datasets, calling the resulting dataset as difference-dataset. If the similarity between the datasets is true, the values of the difference-dataset should have zero mean (no reproduction bias) and a relatively small standard deviation (not too small to make both datasets indistinguishable or too large to make them completely unrelated). Table 4 shows the mean and standard deviation for 20 continuous and discrete features of the above-mentioned difference-dataset. We can observe, in both cases, that the behavior is the expected one. The synthetic features exhibit variability (no exact copy) and are centered on expected values (no bias). Similarly, Figure 8 gives the distribution of values for several continuous and discrete features (upper and lower diagrams, respectively) of the difference-dataset. The Y-axis of the histograms corresponds to the number of repetitions of a given value, and the X-axis represents the values of the difference-dataset. These values are, in all cases, in the interval [-1,1], due to the [0,1] scaling and one-hot encoding of the continuous and discrete features, respectively (Section 3.1).

		Difference values Original vs. Synthetic	
	Continous features	mean	std
1	duration	-0.0027	0.1706
2	src_bytes	-0.0033	0.1289
3	dst_bytes	0.0029	0.1466
4	wrong_fragment	0.0001	0.1138
5	hot	-0.0004	0.0354
6	num_failed_logins	-0.0002	0.0101
7	logged_in	0.0014	0.4817
8	num_compromised	-0.0008	0.0248
9	root_shell	-0.0015	0.0426
10	num_root	-0.0005	0.0238
11	num_file_creations	-0.0001	0.0119
12	num_access_files	-0.0003	0.0126
13	is_guest_login	0.0009	0.1227
14	count	0.0253	0.2577
15	srv_count	0.0049	0.2522
16	serror_rate	0.0080	0.3914
17	srv_serror_rate	0.0064	0.3939
18	rerror_rate	0.0061	0.4225
19	srv_rerror_rate	0.0075	0.4256
20	same_srv_rate	-0.0052	0.3355

		Difference values Original vs. Synthetic	
	Discrete features	mean	std
97	service=red_i	0.0000	0.0080
98	service=remote_job	0.0006	0.0249
99	service=rje	0.0007	0.0261
100	service=shell	0.0003	0.0266
101	service=smtp	-0.0094	0.3352
102	service=sql_net	0.0019	0.0441
103	service=ssh	0.0020	0.0546
104	service=sunrpc	0.0020	0.0633
105	service=supdup	0.0043	0.0660
106	service=systat	0.0027	0.0694
107	service=telnet	-0.0043	0.2017
108	service=tftp_u	0.0000	0.0049
109	service=tim_i	0.0000	0.0132
110	service=time	0.0029	0.0864
111	service=urh_i	0.0000	0.0089
112	service=urp_i	-0.0006	0.1002
113	service=uucp	-0.0118	0.1532
114	service=uucp_path	0.0034	0.0868
115	service=vmnet	0.0021	0.0875
116	service=whois	0.0043	0.0813

Table 4. Mean and standard deviation of difference between values (original vs. synthetic) for several features.

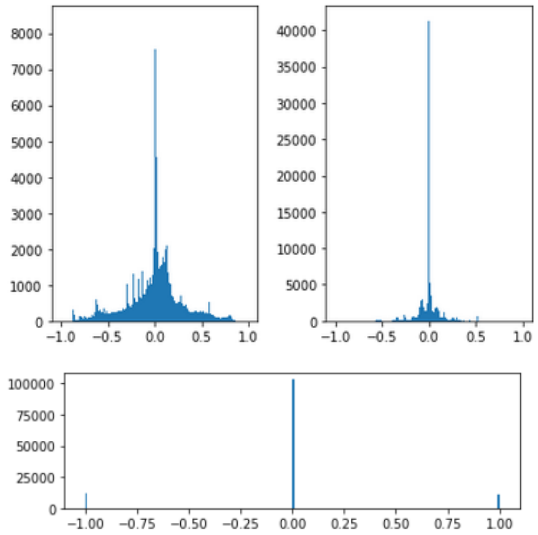


Figure 8. Distribution of value difference between original and synthetic datasets for several continuous and discrete features (upper and lower diagrams, respectively).

4.2. Improvement in classification results

The purpose of this section is to show that the new synthesized data can be used to improve classification results for common machine learning classifiers. That means that the synthetic data can be used as new training data. These results additionally confirm that the synthesized data have similar structure to the original but including enough variability to improve the performance of a classifier.

In Table 5, we present the accuracy obtained with four different classifiers: Random Forest, Logistic Regression, Linear SVM and Multilayer Perceptron (MLP); where for training data we use the NSL-KDD Training dataset alone or with additional synthesized samples, and for test data we use, in all cases, the NSL-KDD Test dataset.

It is important to see the difference between the results of Tables 2 and 5, as they try to prove two different things. Table 2 presents the results when the NSL-KDD training dataset (original or synthesized) is used both for training and prediction, differentiating only if the data set used is original or synthesized. In contrast, Table 5 shows the results when the NSL-KDD test dataset is used exclusively for prediction (without synthetic data), while a combination of original training data and synthetic data is used for the training. For each particular classifier, the six columns in Table 5 correspond to accuracy results in different scenarios: (1) using only the original NSL-KDD training dataset, (2) using the original NSL-KDD dataset repeated twice, (3) using the original NSL-KDD dataset repeated three times, (4) using the original NSL-KDD training dataset plus an additional synthetic dataset made up from the same labels as the original one, (5) using the original NSL-KDD training dataset plus two additional synthetic datasets made up from the same labels as the original one, and (6) using the original NSL-KDD training dataset plus a synthetic dataset made up from a proportion of labels such that the final proportion of labels is balanced in the complete dataset.

	Accuracy: Random Forest						Accuracy: Linear SVM					
	Original Training dataset	Training (x2)	Training (x3)	Training + Synthetic	Training + Synthetic (x2)	Training + Synthetic (balanced)	Original Training dataset	Training (x2)	Training (x3)	Training + Synthetic	Training + Synthetic (x2)	Training + Synthetic (balanced)
Option A	0.7304	0.7303	0.7297	0.7318	0.7322	0.7319	0.7388	0.7389	0.7390	0.7324	0.7308	0.7414
Option B	0.7304	0.7303	0.7297	0.7318	0.7330	0.7338	0.7388	0.7389	0.7390	0.7414	0.7418	0.7357
Option C	0.7304	0.7303	0.7297	0.7339	0.7346	0.7361	0.7388	0.7389	0.7390	0.7549	0.7565	0.7723

	Accuracy: Logistic Regression						Accuracy: MLP					
	Original Training dataset	Training (x2)	Training (x3)	Training + Synthetic	Training + Synthetic (x2)	Training + Synthetic (balanced)	Original Training dataset	Training (x2)	Training (x3)	Training + Synthetic	Training + Synthetic (x2)	Training + Synthetic (balanced)
Option A	0.7362	0.7359	0.7370	0.7661	0.7663	0.7547	0.7752	0.7740	0.7694	0.7831	0.7865	0.7775
Option B	0.7362	0.7359	0.7370	0.7583	0.7592	0.7530	0.7752	0.7740	0.7694	0.7830	0.7950	0.7721
Option C	0.7362	0.7359	0.7370	0.7643	0.7701	0.7729	0.7752	0.7740	0.7694	0.7947	0.7925	0.7926

Table 5. Classification results when increasing the number of training samples with synthetic samples. Predictions are done with NSL-KDD Test dataset.

The values in Table 5 are color coded (same code as Table 1). We can appreciate that the increase in performance is clear when increasing the number of synthetic samples. It is also clear the difference between the first three columns and the next three, for almost all options and classifiers. This difference provides evidence that employing synthetic data increases classifiers performance, while simply repeating the original data does not provide any significative advantage. It is important to realize that this happens when using very different classifiers.

We can see in Table 5 that for Option C there is always an increase in performance for all classifiers when employing additional synthetic data. Moreover, using a balanced dataset (last columns) provides best results, at least for Option C, which is the Option we have chosen as our best model.

Finally, we compare VGM with seven SOTA synthetic over-sampling algorithms: (1) SMOTE [9], (2) SMOTE Borderline [10], (3) SMOTE+ENN [12, 7], (4) SMOTE+Tomek [12, 7], (5) ADASYN [14], (6) SMOTE-SVM [11] and (7) EasyEnsemble [15, 7].

Table 6 presents a comparison of several classification performance metrics: accuracy and F1 score [20], when different well-known classifiers are trained with synthetic data generated by the aforementioned over-sampling algorithms. To avoid bias due to specific effectiveness of synthetic data with some particular classifier, we repeat the experiment with four classifiers: random forest, multinomial logistic regression, linear SVM and MLP. In all cases, the classifiers are trained with a balanced dataset that is constructed using the different synthetic data generation algorithms. The base dataset used to generate the synthetic data has been the NSL-KDD Training dataset. All the prediction metrics (accuracy and F1) are obtained with the NSL-KDD Test dataset.

We can observe (Table 6) that VGM exhibits a better average performance than the other algorithms; some give better results for a specific classifier but in average VGM gives the best results. The results depend on both the classifier and the oversampling method. The intention here is to show that VGM provides average results as good as any SOTA oversampling method and in many cases better.

We used the weighted average provided by scikit-learn [31] to calculate F1 score. The values in Table 6 are color-coded in a manner similar to previous tables.

	Accuracy					F1				
	Random Forest	Logistic Regression	Linear SVM	MLP	Average Accuracy	Random Forest	Logistic Regression	Linear SVM	MLP	Average F1
VGM	0.7361	0.7729	0.7723	0.7926	0.7685	0.6908	0.7529	0.7557	0.7645	0.7410
SMOTE	0.7298	0.7717	0.7758	0.7795	0.7642	0.6840	0.7483	0.7634	0.7455	0.7353
SMOTE Borderline	0.7376	0.7597	0.7679	0.7738	0.7597	0.6920	0.7353	0.7496	0.7303	0.7268
SMOTE+ENN	0.7294	0.7533	0.7675	0.7795	0.7574	0.6819	0.7208	0.7517	0.7428	0.7243
SMOTE+Tomek	0.7324	0.7548	0.7657	0.7889	0.7605	0.6847	0.7254	0.7501	0.7573	0.7294
SMOTE SVM	0.7425	0.7629	0.7799	0.7798	0.7663	0.6993	0.7474	0.7723	0.7406	0.7399
Easy Ensemble	0.7373	0.7749	0.7721	0.7782	0.7656	0.6910	0.7528	0.7541	0.7579	0.7390
ADASYN	0.7312	0.7521	0.7512	0.7800	0.7536	0.6833	0.7236	0.7293	0.7458	0.7205

Table 6. Classification metrics when using training data generated by several over-sampling algorithms.

4.3. Model training

As part of the different experiments performed, we have learned that the inclusion of regularization by using drop-out provides worse results, similarly to increasing the number of layers for the encoder and decoder beyond 2 or 3 layers. We have seen also that results are sensitive to the number of training epochs, having better results when this number is over 50. All the models converged easily.

Table 7 presents the parameters used for the training of the different models.

We have used Tensorflow to implement all the VAE models, and the python package scikit-learn to implement the different classifiers. All computations have been performed in a commercial PC (i7-4720-HQ, 16GB RAM).

	Model parameters									
	Input dimension	Output dimension	Latent space dimension	Activation function last encoder layer	Activation function last decoder layer	Activation function other layers	Batch size	# epochs	Dropout	Nodes per layer
Option A	Labels: 5 (one-hot encoding)	116 (32 continuous and 84 discrete)	25	Linear	Sigmoid (all features)	ReLU	200	80	No	[500,500,500,25,500,500,500]
Option B	Labels: 5 (one-hot encoding)	116 (32 continuous and 84 discrete)	25	Linear	Sigmoid (discrete features) and Linear (continuous features)	ReLU	200	80	No	[500,500,500,25,500,500,500]
Option C	X: 116 Labels: 5 (one-hot encoded)	116 (32 continuous and 84 discrete)	25	Linear	Sigmoid (all features)	ReLU	200	80	No	[500,500,500,25,(300+200),500,500]

Table 7. Parameters used to train the models

5. Discussion and conclusion

This work is unique in presenting the application of a VAE as a generative model for intrusion detection. The model is able to synthesize data with both continuous and categorical features. We have demonstrated that the data generated is similar to the original data, and, at the same time, have enough variability to be effective in improving the detection performance of several classifiers when used together with the original data.

Other aspect unique to this work is the ability to synthesize the new samples from the intrusion labels to which the synthetic data should belong, with the advantage of not relying on

particular samples associated with the labels. This association is usually noisy and identifying a canonical set of samples associated with each label can be complex. Therefore, the proposed model streamlines the data generation process based on the intrusion label.

We have analyzed different VAE architecture variants for the proposed model, providing an extensive study on the alternatives. When considering all experiments carried out to determine the similarity of synthetic data to real intrusion detection data, and its capacity to be used as new training data we can conclude that the model based on conditional VAE with Gaussian and Bernoulli distributions presents the best results.

Also, we provide a comparison of our best model with seven common SOTA over-sampling algorithms (SMOTE, ADASYN…), showing that the synthetic data generated by our proposed model offer better metrics of average performance (accuracy, F1) when four common classifiers are trained with this data, compared to the results obtained with the data generated by the alternative algorithms. This indicates that the data generated with the proposed model is closer to the original data and can better reproduce the probability distribution of its features.

Acknowledgments

This work has been partially funded by the Ministerio de Economía y Competitividad del Gobierno de España and the Fondo de Desarrollo Regional (FEDER) within the project "Inteligencia distribuida para el control y adaptación de redes dinámicas definidas por software, Ref: TIN2014-57991-C3-2-P", in the Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento.

References

[1] Kingma DP, Welling M (2014) Auto-Encoding Variational Bayes. arXiv:1312.6114v10 [stat.ML]

[2] Goodfellow IJ, Pouget-Abadie J, Mirza M et al (2014) Generative Adversarial Networks. arXiv:1406.2661v1 [stat.ML]

[3] Miao Y, Yu L, Blunsom P (2015) Neural Variational Inference for Text Processing. arXiv:1511.06038 [cs.CL].

[4] Yang Z, Hu Z, Salakhutdinov R et al (2017) Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. arXiv:1702.08139 [cs.NE].

[5] Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and Harnessing Adversarial Examples. arXiv:1412.6572 [stat.ML].

[6] Galar M, Fernandez A, Barrenechea E et al (2012) A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 4, pp. 463-484

[7] He H, Garcia EA (2009) Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263-1284.

[8] Weiss G (2004) Mining with rarity: a unifying framework. ACM SIGKDD Explorations, vol. 6. no. 1. pp. 7-19.

[9] Chawla NV, Bowyer KW, Hall LO et al (2002) SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, vol. 16. pp. 321-357.

[10] Han H, Wen-Yuan W, Bing-Huan M, (2005) Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. Advances in intelligent computing, pp. 878-887.

[11] Nguyen HM, Cooper EW, Kamei K (2011) Borderline over-sampling for imbalanced data classification. International Journal of Knowledge Engineering and Soft Data Paradigms. vol. 3. no. 1. pp. 4-21.

[12] Batista G, Prati RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter. vol. 6. no. 1. pp. 20-29.

[13] Cieslak DA, Chawla NV, Striegel A (2006) Combating imbalance in network intrusion datasets. IEEE International Conference on Granular Computing, pp. 732-737.

[14] He H, Bai Y, Garcia EA et al (2008) ADASYN: Adaptive synthetic sampling approach for imbalanced learning. IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322-1328.

[15] Liu XY, Wu J, Zhou ZH (2009) Exploratory Undersampling for Class-Imbalance Learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 39, no. 2, pp. 539-550.

[16] Tavallaee M, Bagheri E, Lu W et al (2009) A Detailed Analysis of the KDD CUP 99 Data Set. Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009), pages 53-58.

[17] Gregor K, Danihelka I, Graves A et al (2015) DRAW: A Recurrent Neural Network For Image Generation. arXiv:1502.04623 [cs.CV].

[18] Jang E, Gu Sh, Poole B (2016). Categorical Reparameterization with Gumbel-Softmax. arXiv:1611.01144v2 [stat.ML].

[19] An J, Cho S (2015) Variational Autoencoder based Anomaly Detection using Reconstruction Probability. SNU Data Mining Center, 2015-2 Special Lecture on IE

[20] Bhuyan MH, Bhattacharyya DK, Kalita JK (2014) Network Anomaly Detection: Methods, Systems and Tools. IEEE Communications Survey & Tutorials, vol. 16, no. 1.

[21] Sommer R, Paxson V (2010) Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. IEEE Symposium on Security and Privacy.

[22] Ingre B, Yadav A (2015) Performance Analysis of NSL-KDD dataset using ANN. 2015 International Conference on Signal Processing and Communication Engineering Systems, Guntur, pp. 92-96.

[23] Ibrahim LM, Basheer DT, Mahmod MS (2013) A comparison study for intrusion database (KDD99, NSL-KDD) based on self-organization map (SOM) artificial neural network. Journal of Engineering Science and Technology, vol. 8, no. 1 pp. 107-119, School of Engineering, Taylor's University.

[24] Hinton GE, Zemel RS (1993) Autoencoders, minimum description length and Helmholtz free energy. Proceedings of the 6th International Conference on Neural Information Processing Systems, pp. 3-10.

[25] Bengio S, Bengio Y (2000) Taking on the curse of dimensionality in joint distributions using neural networks. IEEE Transactions on Neural Networks, vol. 11, no. 3, pp. 550-557.

[26] Siracusa MR, Tieu K, Ihler AT et al (2005) Estimating dependency and significance for high-dimensional data. Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005, vol. 5, pp. v/1085-v/1088.

[27] Dhar SS, Chakraborty B, Chaudhuri P (2014) Comparison of multivariate distributions using quantile–quantile plots and related tests. arXiv:1407.1212 [math.ST].

[28] Burke MD (1977) On the multivariate two-sample problem using strong approximations of the EDF". Journal of Multivariate Analysis. 7. pp. 491–511.

[29] Justel A, Peña D, Zamar R (1997) A multivariate Kolmogorov–Smirnov test of goodness of fit. Statistics & Probability Letters, vol.35, Issue 3, pp. 251-259.

[30] Tennekes M, Jonge E, Daas PJH (2013) Visualizing and Inspecting Large Datasets with

Tableplots. Journal of Data Science 11, pp. 43-58.
[31] Pedregosa F, Varoquaux G, Gramfort A et al (2011)Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12, pp. 2825-2830.

Manuel Lopez-Martin is a research associate and Ph.D. candidate at Universidad de Valladolid, Spain. His research activities include the application of machine learning models to data networks. He received his M.Sc. in telecommunications engineering in 1985 from Universidad Politécnica de Madrid (UPM), Spain and his M.Sc. in computer sciences in 2013 from Universidad Autonoma de Madrid. He has worked as a data scientist at Telefonica and has more than 25 years of experience in the development of IT software projects.

Belen Carro received a Ph.D. degree in the field of broadband access networks from the Universidad de Valladolid in 2001. She is a professor and director of the Communications Systems and Networks (SRC) laboratory at Universidad de Valladolid, working as a research manager in NGN communications and services, VoIP/QoS, and machine learning. She has supervised a dozen Ph.D. students and has extensive research publications experience as author, reviewer, and editor.

Antonio Sanchez-Esguevillas received a Ph.D. degree in the field of QoS over IP networks from Universidad de Valladolid in 2004. He has managed innovation at Telefonica and has been an adjunct professor and honorary collaborator at Universidad de Valladolid, supervising several Ph.D. students. He has coordinated very large international R&D projects and has over 50 international publications and several patents. His current research interests include digital services and machine learning.

POSTER MLSS-2018

Universidad de Valladolid

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

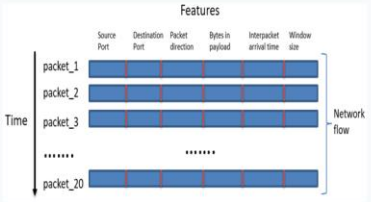
Application of deep learning architectures to prediction problems in data networking

M. Lopez-Martin, B. Carro
mlopezm@acm.org, belcar@tel.uva.es

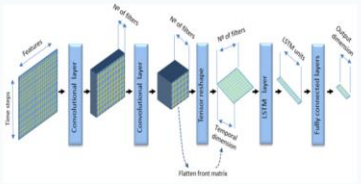
Network Traffic Classification

Apply a CNN+RNN model to detect type of service (e.g. HTTP, DNS,..) from network packets headers.

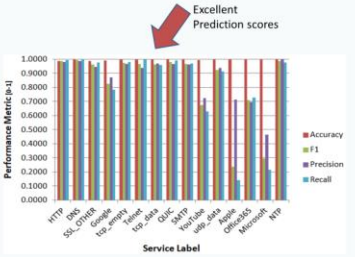
M Lopez-Martin, B Carro, A Sanchez-Esguevillas, J Lloret. "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things", ***IEEE Access*, vol. 5, pp. 18042-18050, 2017.**
doi: 10.1109/ACCESS.2017.2747560



Data packets preparation



Network details

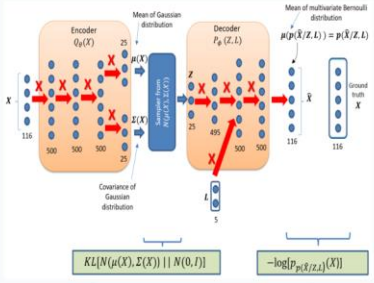


Prediction detailed results vs. type of service

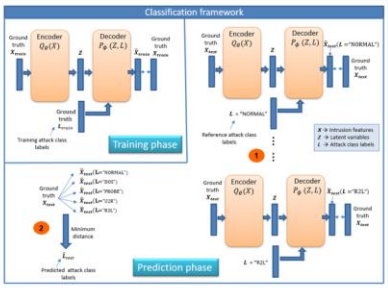
Intrusion detection

Generative model based on Conditional-VAE for intrusion detection

M Lopez-Martin, B Carro, A Sanchez-Esguevillas, J Lloret. "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT".
***Sensors* 17 (9), 2017**
doi:10.3390/s17091967



Network detail



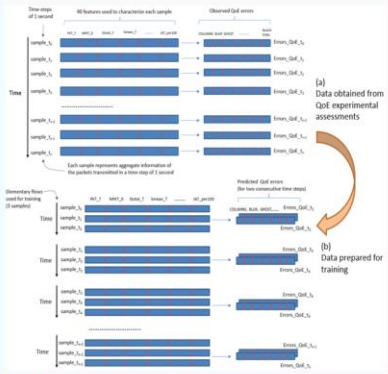
Classification framework

Detection metrics for different models

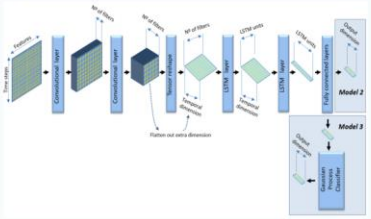
Video Quality of Experience prediction

Video QoE prediction with a DL model using only aggregated packets information

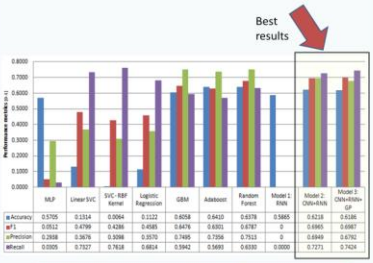
M. Lopez-Martin, B. Carro, J. Lloret, S. Egea, A. Sanchez-Esguevillas. "Deep learning model for multimedia Quality of Experience prediction based on network flow packets",
***IEEE Communications Magazine*, September 2018**
doi: 10.1109/MCOM.2018.1701156



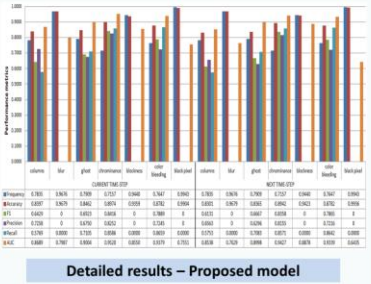
Data packets preparation



Network details



Prediction metrics for different models



Detailed results – Proposed model

Current areas of work

- Deep reinforcement learning
- DL + Gaussian Processes
- Kernel approximation + Shallow learning
- Few-shots learning