Deep-FDA: Using Functional Data Analysis and Neural Networks to Characterize Network Services Time Series

Daniel Perdices, Jorge E. López de Vergara, Javier Ramos

***Abstract*—In network management, it is important to model baselines, trends, and regular behaviors to adequately deliver network services. However, their characterization is complex, so network operation and system alarming become a challenge. Several problems exist: Gaussian assumptions cannot be made, time series have different trends, and it is difficult to reduce their dimensionality. To overcome this situation, we propose Deep-FDA, a novel approach for network service modeling that combines functional data analysis (FDA) and neural networks. Specifically, we explore the use of functional clustering and functional depth measurements to characterize network services with time series generated from enriched flow records, showing how this method can detect different separated trends. Moreover, we augment this statistical approach with the use of autoencoder neural networks, improving the classification results. To evaluate and check the applicability of our proposal, we performed experiments with synthetic and real-world data, where we show graphically and numerically the performance of our method compared to other state-of-the-art alternatives. We also exemplify its application in different network management use cases. The results show that FDA and neural networks are complementary, as they can help each other to improve the drawbacks that both analysis methods have when are applied separately.**

***Index Terms*—functional data analysis; network monitoring and management; autoencoders; service characterization; time series; baseline model.**

I. Introduction

In recent times, network service monitoring has become a task of paramount importance for both network managers and researchers. Service monitoring allows characterizing the traffic to obtain baselines that are useful for the detection of performance faults and security problems. Given the growing complexity of large corporations' networks that comprise several thousands of subnets, manual network inspection and characterization using all the generated monitoring data has become an unfeasible task. Thus, it is necessary to find an automatic way to identify when any of these network segments is not behaving as expected. To this end, the services that are provided on top of such networks have to be characterized.

To make the most of monitoring data, we need to build baseline models [1]–[3], which provide a forecast of the time series of the next period (e.g. hour, day, week, month), that allow for behavioral differentiation and accurate prediction, which is an actual challenge in network management [4]. Additionally, the huge amount and diversity of data obtained from service monitoring calls for the application of more capable statistical techniques and machine learning approaches in order to build reliable models for service characterization [5]. Such models may be used to increase the operational intelligence and ease the life of network managers and security teams when troubleshooting problems, thus reducing their response time to incidents.

Specifically, two of the most relevant applications of baseline models are capacity planning and alarming. The motivation of the first case is straightforward. In order to proactively provision resources for services, we need to know in advance the expected values of service related metrics (i.e. load) to ensure that the service is delivered without problems. This approach is an improvement over traditional reactive systems that are typically based on short-time predictions, which can lead to temporal stress periods where systems are saturated due to underprovisioning, until service is scaled up. On the other hand, alarming is an important task in network management, where operators need be notified about abnormal behaviors that may be caused by a variety of reasons, such as network attacks or device failure. Having a baseline estimate that characterizes the network services allows operators to detect anomalies automatically.

Traditionally, baseline approaches rely on several assumptions such as process stationarity or data Gaussianity, which are not always applicable [6]–[8] given the fast paced changing nature of current services and network deployments. Even checking some of the hypotheses is an arduous task since different trends, such as workdays vs holidays, must be separated beforehand. These multiple trends usually lead operators to manually establish the baselines or thresholds, which is a solution that clearly does not scale for large networks with thousands of services. In this light, several methods have been proposed to characterize how a given service works. The straightforward solution is the application of clustering techniques, such as k-means, to group the values of a variable or a set of variables in order to identify the different behaviors.

Manuscript received May 1st, 2020; revised September 21st, 2020; revised December 11th, 2020; accepted January 17th, 2021.

All authors are with Universidad Autónoma de Madrid, Spain. Daniel Perdices and Jorge López de Vergara are also with Naudit HPCN, Spain.

This work has been partially supported by the European Commission under the project H2020 METRO-HAUL (Project ID: 761727), by the Spanish State Research Agency under the project AgileMon (AEI PID2019-104451RB-C21) and by the Spanish Ministry of Science, Innovation and Universities under the program for the training of university lecturers (Grant number: FPU19/05678).

Cite as: D. Perdices, J. E. López de Vergara, J. Ramos, "Deep-FDA: Using Functional Data Analysis and Neural Networks to Characterize Network Services Time Series," *IEEE Transactions on Network and Service Management*, 2021. DOI:10.1109/TNSM.2021.3053835

Once we have grouped similar behaviors together, we can rely on classical methodologies [9] to provide more accurate and significant models.

However, traditional multivariate clustering algorithms are not suitable when working with traffic time series, given that the model treats some inherent characteristics, such as the high correlation between consecutive points, as redundant information. Moreover, the periodicity of the repeated patterns and events is a key characteristic when working with time series and the multivariate approaches do not take into account such structure. On the other hand, these time series can be considered as functional data [10], [11] solving these redundancy issues either by adjusting the basis or using such conditions as part of the structural hypothesis.

Besides direct approaches, such as functional k-means [12], another approach is to reduce the dimensionality of the problem and project the time series to a finite dimension space in order to capture relevant characteristics. Then, a clustering over such relevant characteristics can be performed. Traditionally, this process is done by means of Principal Components Analysis (PCA) or, in this case, by means of Functional PCA (FPCA) [13], where the latter approach has the advantage of working properly with high dimensional data, i.e. we can have fewer curves than the length of the time grid of the series.

In recent years, with the development of advanced artificial neural network techniques and the boom of deep learning [14], mechanisms similar to FPCA have been used to obtain better embeddings. For instance, autoencoder (AE) neural networks are based on a similar idea, where an input space of dimension d is zipped into an embedding with a lower space dimension K and then unzipped again to obtain an output space of dimension d. The idea behind this type of neural network is that the embedding captures the relevant information, allowing us to differentiate behaviors in a much more controlled space. As this type of networks is typically fed with multivariate data and do not perform correctly when working directly with time series, we propose the use of functional basis as input to perform a functional clustering approximation so both methods can be compared.

In this work, we combine FDA-based techniques and neural networks to characterize network services based on time series classification, exploring the feasibility of using both methods together to make the most of each one and overcome their limitations. For instance, neural networks present a huge amount of hyperparameters that must be tuned in order to obtain accurate results. Typically, this problem is addressed by parameter space exploration techniques that try to minimize the loss function or other similar metric, varying the configuration of the hyperparameters of the neural net. This is a highly demanding task, both in terms of time and computing power and, in some scenarios, it produces similar results to the ones obtained with FPCA, as we will show in following sections.

Our approach for network service modeling, called DeepFDA, is based on the use of functional clustering and neural networks, and we apply it on concurrent flow time series obtained from aggregated enriched network flow records. Using our characterization method for every segment in a large network allows for an automatic baselining process. With these baselines, we can identify deviations from expected behavior, especially when the subnets expose multimodal behavior, and trigger alarms that are inspected on-demand by network managers instead of manually inspecting thousands of network segments, thus easing network managers' life. Our proposal solves the problems stated before to obtain baseline models that characterize the network behavior, given that: (i) it can be applied to non-Gaussian distributions; (ii) it works properly with time series with multiple trends; (iii) it defines a way in which time series can be inputted to an AE neural network, allowing its application to this type of data, reducing their dimensionality.

To the best of our knowledge, this work has several novel contributions to the state of the art, which are: (i) the use of advanced preprocessing techniques, such as companding for network time series; (ii) the mixture functional model used to describe network time series, which is able to adapt to more complex situations than other models; (iii) the adaption of clustering benchmarking metrics to the functional setting, using functional distances instead of vector distances; and last but not least (iv) the FDA-based approach to neural network inputs, where we incorporate functional data structure in neural networks showing significant results, such as the FPCA relation with autoencoders.

The rest of the paper is organized as follows: In Section II we review similar works and pinpoint the strengths of our proposal compared with the state of the art. Next, Section III presents the methodology used for both preprocessing the data and constructing and applying our proposed model. In Section IV, we evaluate our model and present the results for synthetic and real-world data and in Section V, we present the relevant lessons learned and take away messages derived from this work. To sum up, Section VI concludes this paper.

II. Related work

Few works have addressed the use of FDA applied to network related data. In [15], authors propose an architecture for network data processing using functional data and evaluate some application cases such as clustering, outlier detection and capacity planning. Furthermore, authors in [16] apply functional data techniques over a set of Key Performance Indicators (KPIs) to predict potential performance problems in radio cells. Authors in [17] explored co-clustering methods for functional data by an adaption of the expectation-maximization algorithm to functional data, showing the application of co-clustering to mobile networks. Although these works point out some interesting applications of the functional data analysis, they lack the service-oriented approach, the combination with machine learning and the functional multi-modal approach.

On the other hand, in recent years several works have used machine learning and neural network techniques applied to network data. For instance, authors in [18], present a survey on the application of machine learning techniques to different network areas, such as traffic prediction and classification or network security and routing.

Other approaches, like [19], present a traffic monitoring analytics system. Such system uses clustering techniques and

autoencoders over flow features extracted from incoming traffic to detect attacks and anomalies with a 76% recall. Similarly, in [20], authors apply variational autoencoders over NetFlow data to detect and cluster network anomalies. Other works like [21] use autoencoders, isolation forest and PCA techniques to detect anomalous packets in TCP data transfers by means of clustering.

The above works provide valuable insight into the application of machine learning techniques over network data, but they are focused on classifying either individual packets or flow records rather than classifying network aggregates using time series information. These approaches are suitable for anomalies or attacks detection, where some previous knowledge is available, but not for service or behavioral characterization. In this work, we present clustering techniques applied to time series of network aggregates, not suitable for fine-grained network monitoring, but more useful to network provisioning and performance assessment.

Recently, Hidden Markov Models (HMM) and Bayesian networks have also received much attention in network monitoring. Authors in [22] present a HMM to predict traffic volume in terms of flow counts using an auto regressive approach. In [23], a system is presented to segment time series of network delay using an Infinite HMM. Although the latest approach is helpful to detect different states of the network, they lose the time component, so it is not possible to know when the network will change from one state to other. Our aim here is different: we classify the whole daily time series and not time sub-intervals, since we are interested in characterizing frequent trends of the network operation without losing the time component.

Focusing on service characterization, authors in [24] model key service metrics using infrastructure measurements in a cloud environment by means of mixture density neural network. This type of network provides as output model the parameters for a mixture of Gaussian distributions. Using such a model, they can predict Service-Level Agreement (SLA) conformance. Although this work is somehow similar to our proposal, it focuses mainly on the prediction of conformance using mean values estimated from a distribution. In our case, we provide a model based on a mixture of stochastic processes (not necessary Gaussian processes), whereas they present a Gaussian mixture model for each time, leading to having precise models in each time moment, but not being able to see time-dependent behaviors such as correlation between events.

Finally, while all previous works addressed the classification or characterization problem by using either functional or machine learning approaches separately, regarding the joint use of functional data and neural networks there is little literature. For instance, in [25], a method for using functional data as input for neural networks is presented. In such work, several methods for functional processing such as FPCA or projection on smooth bases are presented. A similar approach is presented in [26], focusing on multi-layer perceptrons. While these works present some interesting foundational ideas that are used in this paper, they rely on simple neural networks that are far from modern deep neural network models. Moreover, there are some contributions of the neural networks to the functional data analysis; more explicitly, radial basis function neural networks [27] exposed a kernel method that is inherently based on a functional data representation of the data. Nevertheless, its application to networking has been limited [28] and it is restricted to this type of basis representation, where we propose a more general idea of using any functional orthonormal basis representation.

III. Methodology

The methodology of Deep-FDA aims at characterizing network service behavior through their time series, by classifying the different trends that arise on daily basis. It is presented in three parts: (i) data preprocessing, where we ensure the data is suitable for the different methods; (ii) model and methods definition, where we state our model, its assumptions and methods to estimate the parameters to define the functional centroids of every time series classes; and (iii) application of the model, where obtained properties and results are commented.

A. Preprocessing the data

Scale differences and inconsistencies can heavily affect both machine learning and functional data analysis techniques [29]. Thus, the first step of any data analysis pipeline should be rectifying the scale. For that purpose, we perform a normalization using the maximum of the time series so that all values go from 0 to 1.

However, outliers can affect this normalization process, leading to an ineffective use of the aforementioned range. Moreover, network traffic burstiness causes a large span in traffic values. To cope with these issues, we employ companding (compressing and expanding) methods that transform the signals to take advantage of the full range.

Table I contains the precise formulation of the employed companding methods, as defined in [30]. In addition, Figure 1 shows a comparison of the original signal, the scaled signal and the signal after the application of companding processes. Both mentioned companding methods map the interval $[0,1]$ to $[0,1]$ with a logarithmic scale and, in the case of the A-law, with a linear region where the transform is just a scaling. This helps us coping with the effects of outliers and deviations while keeping the details of low-amplitude regions, thus allowing us to properly classify and differentiate the time series.

In the case of Figure 1, we see that a peak in the number of connections at the end of the day (due to periodical backups) causes an underused range when processing data. Although this issue seems shallow, we require that the curves are well spaced, i.e. the L^2 distance,

$$d_{L^2}(f,g)^2 = \int_0^T (f(t) - g(t))^2 dt, \tag{1}$$

or the L^1 distance,

$$d_{L^1}(f,g) = \int_0^T |f(t) - g(t)| dt, \tag{2}$$

between curves of different clusters is high enough, so that our method can correctly detect the different clusters. If the time series are not distant, some clusters that are detectable without the peak will no longer be found due to the normalization, as they will be very close in the normalized $[0,1]$ interval.

TABLE I
COMPANDING METHODS FOR SIGNAL $x \in [0,1]$ IN TERMS OF PARAMETERS A, $L = 1 + \log(A)$ AND μ.

Name	Transform ($F(x)$)	Inverse Transform ($F^{-1}(y)$)
A Law	$\frac{1}{L}\begin{cases} Ax & \text{if } x \le A^{-1} \\ 1+\log(Ax) & \text{otherwise} \end{cases}$	$\frac{1}{A}\begin{cases} y(1+\log(A)) & \text{if } y \le L^{-1} \\ e^{(yL-1)}x & \text{otherwise} \end{cases}$
μ Law	$\frac{\log(1+\mu x)}{\log(1+\mu)}$	$\frac{1}{\mu}\left((1+\mu)^y - 1\right)$

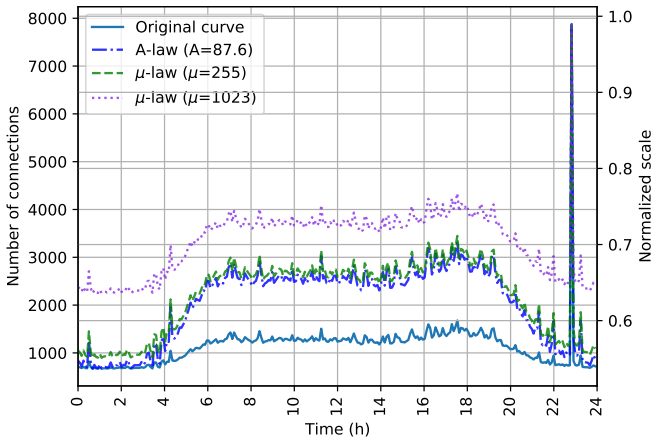


Fig. 1. Example of time series preprocessing for the number of flows. Left axis shows the original scale, whereas right axis shows the scaled range.

B. Defining the model and the method

For the purpose of modeling different time series classes, we define $X_t = \sum_{i=1}^{d} \theta_i X_t^{(i)}$ as the model for our data, where θ_i are some random coefficients that satisfy (i) support$(\theta_i) = \{0,1\}$, i.e. θ_i only takes values in the set $\{0,1\}$, and (ii) $\sum_{i=1}^{d} \theta = 1$, i.e. only one of the θ_i can be activated at the same time. The $X_t^{(i)}$ are stochastic processes defined as

$$X_t^{(i)} = \mu_t^{(i)} + \varepsilon_t^{(i)}, \tag{3}$$

where $\mu_t^{(i)}$ are some deterministic functions and $\varepsilon_t^{(i)}$ are random error terms with $\mathbb{E}[\varepsilon_t^{(i)}] = 0$ and $Var[\varepsilon_t^{(i)}] = (\sigma_t^{(i)})^2$, i.e. $\varepsilon_t^{(i)}$ is heteroscedastic, in contrast to linear models, which require homoscedasticity [31].

Bear in mind that the model aims at defining a concept akin to a mixture of stochastic processes, where we have random coefficients that select which curve the sample comes from. A clear example of this type of behavior can be found, not only in workdays and in non-working days, but also in the first day of the month and special days (e.g. when offers are released). Furthermore, although the model considers a more general concept, our methods will not be able to detect different mixtures if $\mu_t^{(i)}$ are very close, where closeness is defined in terms of the distance that we will be minimizing in our method (normally L^2 or L^1 distances, as defined in (1) and (2), respectively). A limitation of these methods is that they heavily rely on the centroids, so if, for instance, we have two components i and j such that $d(\mu_t^{(i)}, \mu_t^{(j)}) \approx 0$, then only one cluster is detected, no matter how different $\varepsilon_t^{(i)}$ and $\varepsilon_t^{(j)}$ are.

In order to estimate each term, we propose several techniques that aim at the same unsupervised learning problem. The first approach proposes an algorithm to solve the problem directly, whereas the rest use projection to an embedding to solve the problem in a lower dimension space.

1) Direct approach: functional k-means: Since most of the multivariate clustering algorithms work with distances, functional versions can be derived. Here, we describe the functional k-means algorithm [32], an adaption of the original k-means algorithm [33] to the functional setting:

- *Initialization*: The algorithm chooses k curves as initial centroids of the clusters. This can be done by a random selection or a more intelligent random process known as k-means++ [34], whose objective is to select k initial curves that are not very close to each other.
- *Iterative step*: With the centroids of previous iteration (or the initial centroids), we divide our sample into clusters by assigning a cluster to each curve depending on the closest centroid. The closest centroid is determined either by minimizing the L^2 distance (1) or the L^1 distance (2). Once each cluster is computed, the centroid of the cluster can be updated with the deepest function, i.e. by using a depth measurement [35]. These depth measurements generalize the concept of median and quantiles for functional data. Authors in [36], [37] provide precise definitions of the properties of a depth measurement, which is a function $D(f,P)$ that indicates how "deep" the function f is in terms of the distribution of the data, P. In our case, we consider the following half-region depth measure [38]–[40]. Given that P is estimated by the samples $\{f_i\}_{i=1}^{N}$, we define our depth measure
$$D(f,P) = \min(SL_P(f), IL_P(f)), \tag{4}$$
where
$$SL_P(f) = \underset{i=1,\dots,n}{\text{mean}} \frac{1}{T}\int_0^T \mathbb{1}_{\{t\in[0,T];\, f(t)\le f_i(t)\}}(t)dt \tag{5}$$
and
$$IL_P(f) = \underset{i=1,\dots,n}{\text{mean}} \frac{1}{T}\int_0^T \mathbb{1}_{\{t\in[0,T];\, f(t)\ge f_i(t)\}}(t)dt, \tag{6}$$
being $\mathbb{1}_A(t)$ the indicator function of set A. SL and IL measure the proportion of time that f is below the samples or above the samples.
- *Stop criterion*: The algorithm has finished when it reaches an iteration limit or the clusters do not change from one iteration to the next one.

In this case, the centroids are always functions of the sample, so the zero-mean property of the noise is necessary and, in some cases, the centroid is not the visually expected centroid, but just the most similar one in the sample. We will cover this issue and its solution in the evaluation section.

2) Projection-based methods: principal components and autoencoders: Another possible approach is to reduce the dimensionality and to project the data to a finite dimension space, which converts the problem of clustering in a functional space to the same problem in $\mathbb{R}^K$. Several alternatives can be found in the state of the art:

- *Principal Component Analysis (PCA) and Functional Principal Component Analysis (FPCA)*: in this case, we try to find directions that contain the maximum amount of information of the original variable, i.e.

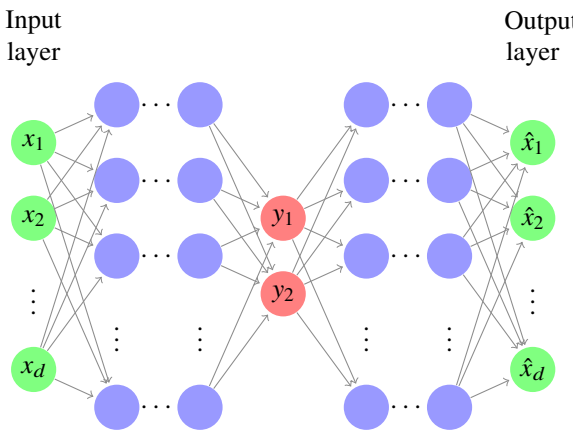


Fig. 2. Architecture of an autoencoder. Green nodes represent the original data X and its compressed version $\hat{X}$, red nodes the embedding output $Y \in \mathbb{R}^2$ and blue nodes intermediate layers.

$$\alpha^{(i)} = \underset{\beta, \langle \beta, \alpha^{(j)} \rangle = \delta_{i,j}}{\arg\max} \left[\mathrm{Var}\left(\langle \mathbb{X}, \beta \rangle \right) \right], \tag{7}$$

where $\delta_{i,j}$ is the Kronecker's delta, $\mathbb{X}$ is the data, $\{\alpha^{(i)}\}_{i=1}^{K}$ are the principal directions (sorted by the value of $\mathrm{Var}\left(\langle \mathbb{X}, \alpha^{(i)} \rangle\right)$) and $\langle \cdot, \cdot \rangle$ is the scalar product. For PCA [41], we have the functions evaluated at some time grid $\{t_k\}_{k=1}^{d}$, so $\mathbb{X}$ is a random vector whose components are $\{X(t_k)\}_{k=1}^{d}$, $\alpha^{(i)} \in \mathbb{R}^d$ and

$$\langle \mathbb{X}, \alpha^{(i)} \rangle = \langle \mathbb{X}, \alpha^{(i)} \rangle_{\mathbb{R}^d} = \sum_{k=1}^{d} X(t_k) \cdot \alpha_k^{(i)}. \tag{8}$$

For FPCA, we have that $\mathbb{X}$ is the stochastic process X, $\alpha^{(i)} \in L^2([0,T])$ and

$$\langle \mathbb{X}, \alpha^{(i)} \rangle = \langle \mathbb{X}, \alpha^{(i)} \rangle_{L^2([0,T])} = \int_0^T X(t) \cdot \alpha^{(i)}(t) dt. \tag{9}$$

In both cases, computation of these $\{\alpha^{(i)}\}_{i=1}^{K}$ can be achieved by extracting the eigenvalues of either the covariance matrix $\Sigma = [\mathrm{Cov}(X(t_i), X(t_j)]_{i,j=1}^{d}$ for PCA or the covariance operator $k(t,s) = \mathrm{Cov}(X(t), X(s))$ for FPCA.
Note that an advantage of FPCA over PCA is that its principal functional component maps directly to the functional structure, whereas PCA may suffer from the curse of dimensionality due to data sparsity [42].
Once principal components are calculated, we need to fix the embedding dimension to some small K (typically, $K = 2, 3$). The process to project any sample $X^{(n)}$ to our embedding space is as simple as defining it as a vector

$$Y^{(n)} = \left[\langle X^{(n)}, \alpha_i \rangle \right]_{i=1}^{K}. \tag{10}$$

Then, $Y^{(n)}$ is usually referred as the coordinates in the principal component basis and the aforementioned operator is just a linear transformation that projects $L^2([0,T])$ to the embedding space $\mathbb{R}^K$ maximizing the amount of information, measured by the explained variance. This last property is equivalent to the minimization problem

$$\min_{\{\alpha_i\}_{i=1}^{K}; \{y_i\}_{i=1}^{K}} \mathbb{E}\left(\left\| X - \sum_{i=1}^{K} y_i \alpha_i \right\|^2 \right), \tag{11}$$

where $\|\cdot\|$ for PCA is the norm-2 of $\mathbb{R}^d$ and for FPCA is the L^2-norm.

- *Autoencoders (AE)*: we train a neural network where input and output are the same with a bottleneck in a middle layer, i.e. a hidden layer with K neurons, where K stands for the dimension of the desired embedding. Figure 2 illustrates this concept. This allows us to encode the original space into a K dimensional one when being trained with loss function the Mean Squared Error (MSE)

$$MSE = \frac{1}{N} \sum_{i=1}^{d} (x_i - \hat{x}_i)^2. \tag{12}$$

Bear in mind that neural networks are usually fed with multivariate data, so there are two possible methods to do it with functional data: (1) ignore the functional nature of the data and, like PCA, use a time grid to build $\mathbb{X} = [X(t_k)]_{k=1}^{d}$ or (2) use a basis such as a Fourier Basis, finite elements, B-splines or even the FPCA basis with K big enough, make it an orthonormal basis (if it was not) and use the coordinates in this basis as input of the neural network.
In first case, the minimization of the MSE leads to the minimization of the norm-2 of $\mathbb{X}$ as an element of $\mathbb{R}^d$, i.e.

$$\hat{X} = \underset{C \in \mathbb{R}^d}{\arg\min}\, \mathbb{E}\left(\sum_{i=1}^{d} (x_i - c_i)^2 \right) = \underset{C \in \mathbb{R}^d}{\arg\min}\, \mathbb{E} \left\| X - C \right\|_{\mathbb{R}^d}^2, \tag{13}$$

whereas in the second case the minimization of the MSE and the basis representation leads to a much stronger property, thanks to a generalization of the Pythagorean Theorem called Parseval's identity:

$$\begin{aligned} \hat{X} &= \underset{f \in L^2([0,T])}{\arg\min}\, \mathbb{E}\left(\int_0^T (X(t) - f(t))^2 dt \right) \\ &= \underset{f \in L^2([0,T])}{\arg\min}\, \mathbb{E} \left\| X - f \right\|_{L^2}^2. \end{aligned} \tag{14}$$

In equation (14), we are assuming two hypotheses: 1) the closure of the span of our basis completes the space (i.e. any element of the space can be represented as a limit of elements built with our basis) and 2) the basis is finite. In ideal conditions, this would mean a contradiction, but in an experimental setup where we have a sampling limit, it is impossible to have an infinite functional basis as well as an infinite coordinate space. Therefore, the norm of our functions can be computed exactly by this sum, given that we are assuming the signals are completely represented using the basis.
As an example, when the embedding layer is of length K, there are no more hidden layers and the activation functions are linear. The first case is equivalent to PCA, whereas the second case is equivalent to FPCA. This can be shown easily since the aforementioned situation leads to an output of the form $\hat{X} = y_1 V_1 + \cdots + y_k V_k$, where $Y = (y_1, \ldots, y_K)$ is X projected in the embedding space and $V_1, \ldots, V_K$ are some vector of $\mathbb{R}^d$ or some functions of $L^2([0,T])$, resulting in the minimization problem given by equations (13) and (14)

$$\min_{\{V_i\}_{i=1}^{K}, \{\beta_i\}_{i=1}^{K}} \mathbb{E}\left(\left\| X - \sum_{i=1}^{K} \beta_i V_i \right\|^2 \right), \tag{15}$$

which is equivalent to PCA and FPCA as defined in (11). This result indicates that not only PCA and FPCA are special cases of an AE, but also that PCA and FPCA can be used to estimate *a priori* the complexity of the AE. This is a helpful result, given the difficulties one can encounter when training neural networks and selecting hyperparameters.

As an advantage of this type of methods, once clusters are computed in the embedding, centroids can also be calculated in the embedding. These centroids may not arise from a statistical argument as before, but they become a reasonable alternative to use when they are mapped to their corresponding function in the original space.

3) Choosing the most suitable K: In both cases, there is a question that remains unanswered: *which K is the most suitable one?* To answer it, we use the silhouette coefficient, which we extend to the functional domain as

$$SC_K = \text{mean}_{1\leq j\leq n} s_K(j), \tag{16}$$

where

$$s_K(j) = \frac{b_K(j) - a_K(j)}{\max\{a_K(j), b_K(j)\}}, \tag{17}$$

$$a_K(j) = \frac{1}{|C_j| - 1} \sum_{i\in C_j; i\neq j} d(i,j) \tag{18}$$

and

$$b_K(j) = \min_{i\neq j} \frac{1}{|C_i|} \sum_{l\in C_i} d(l,j), \tag{19}$$

where C_i stands for the cluster which the curve i^{th} belongs to, and $|C_i|$ the number of curves in the cluster. Note that this applies for the methods of both families, with the only observation that distances can be measured either in the embedding or in the original space for the second group of methods, which may lead to slightly different results. To deal with this, we will ignore the distances in the embedding, since they might not be representative in the original space in the case of AE. Once the silhouette coefficient is computed for each K, we select the K so that SC_K is maximum.

C. Using the model

Before specifying how the model can be used, some remarks about numerical computations must be made. First, it is worth noting that, due to numerical limitations, curves are represented by the evaluation in a grid of points. Therefore, the use of the model or the method is limited to a discretization of the functions, which conveniently simplifies the problem. For the application of the algorithm, we need to estimate the distance between two curves. This can be done by either fitting a basis so our data is just represented by the basis coefficients and basis integrals can be exactly computed or approximated by numerical integration methods such as finite elements.

Once the model is trained, centroids and classified curves are obtained. Centroids provide an estimation of $\mu_t^{(i)}$, which can act as new baseline or even more complex baselines can be obtained from the clusters themselves together with aforementioned methodologies [9]. Usually, such methodologies are based on moving median or moving average [43] of previous data. Deep-FDA provides a systematic way of grouping up similar time series, so baselines can be calculated with previous data that share a trend, instead of getting biased conclusions due to the presence of multiple modes or trends. The final objective of these baselines is to provide critical monitoring services such as alarming or load estimation. On the other hand, cluster proportions can also give an insight of the distribution of the priors that are represented by the $\{\theta_i\}$ random variables of our model. In addition, the classified curves and their distribution could be exploited to obtain information about $\varepsilon_t^{(i)}$ and thus, thresholds for the baselines.

Our proposed models can be used for network management tasks in both proactive and reactive ways depending on the application domain. One of the most interesting use cases of our tool is the capacity planning and resource usage optimization. When it comes to load estimation, it is easy to allocate resources using the centroids, since they provide full day estimation of the load. Using each centroid we can define classes (for instance high, medium and low activity) and we can scale up or down resources proactively for specific services. This allows for elastic resource assignment, which is of paramount importance in multi-tenant environments where resources must be accommodated dynamically in order to maximize their usage while keeping the quality levels high. For example, analyzing the number of connections metric we can scale up and configure service resources (servers, load balancers, etc.) in advance to accommodate more clients in the peak hours, instead of reacting when the system is saturated and clients are already suffering the diminished performance of the service.

Focusing on reactive management, we face the classic problem of alarming services to detect anomalies or abnormal behaviors, such as service outages or traffic increments, which may be related to attacks or misconfigurations. First, we need to define what an anomaly is. Such definition is not universal and may depend on the network conditions and service characteristics. In our use case, we try to generalize defining an anomaly as something unexpected. More formally, an anomaly [44], [45] at time t_a is every X_{t_a} that is far away from the normal or expected value:

$$|X_{t_a} - g(\{X_t : t < t_a\})| > T(\{X_t : t \leq t_a\}), \tag{20}$$

where $g(\{X_t : t < t_a\})$ is the prediction based only on the past, i.e. $g(\{X_t : t < t_a\}) \approx \mathbb{E}\left[X_{t_a} | \{X_t : t < t_a\}\right]$, and T is some threshold that may also depend or not on previous behavior. Normally, T is chosen so that the number of risen alarms is not overwhelming for the network managers, but sensible enough to detect and react to undesired conditions. The regressor g is a function that predicts X_t using the history of the time series. In our proposal, we can directly use the closest $\mu_t^{(c)}$. This is, the function g returns the closest centroid with negligible computation cost, since centroids have been previously calculated.

IV. Evaluation

For the performance evaluation of Deep-FDA, we provide two scenarios: one model-driven simulation where we generate random samples reproducing the model and a second one where we employ three months' worth of real network data.

TABLE II
HYPERPARAMETERS FOR THE EXPERIMENTS

Functional k-means	
Parameter	**Value**
Distance	L^2 distance
PCA	
Number of components	2
FPCA	
Number of components	2
Autoencoder	
Input	Coefficients in the functional basis
Number of hidden layers	3
Size of hidden layers	[5, 2, 5]
Activation functions	ReLu
Loss function	MSE
Optimizer	RMSProp
Learning rate	0.01

Additionally, we show two use cases of the applicability of our approach using real data. For the reproducibility of the results, Table II contains the hyperparameters of the methods, and code of the experiments has been made publicly available[1].

A. Simulations

We reproduce the experiments performed in [35], which Cuevas *et al.* introduce to benchmark functional classification methods, but using different random variables θ_i. Although the context is different (supervised vs unsupervised), we want to test our method under challenging situations where they find substantial differences among classifiers.

To compare the results, different metrics are considered. First, distances to the centroids are calculated, that is, how far are the centroids from the functions $\mu_t^{(1)}$ and $\mu_t^{(2)}$. Second, classification metrics such as the precision or the recall [41] are also computed. Although this is not a supervised learning problem, simulations offer the opportunity to determine which component each sample comes from.

1) Case 1: balanced clusters: We define the mean functions $\mu_t^{(1)} = 30(1-t)t^{1.2}$ and $\mu_t^{(2)} = 30(1-t)^{1.2}t$ and the random noise functions ε_t^1 and ε_t^2 as two independent Gaussian processes with zero mean and covariance $\mathrm{Cov}(X(s), X(t)) = 0.2\exp(-|s-t|/0.3)$. Also, $\theta_1 \sim$ Bernoulli(0.5) and $\theta_2 = 1 - \theta_1$, i.e. both clusters are balanced and the choice is random.

2) Case 2: imbalanced clusters: We proceed as before but $\theta_1 \sim$ Bernoulli(0.85), this is, we assume a situation where clusters are not balanced. Although the authors in [35] do not cover imbalanced experiments, these cases are closer to real data, as we will see in next section.

Figure 3 shows centroids and obtained samples, where the overlap and the small difference between the two centroids are the most remarkable difficulties. As we see in Figure 4, the centroids obtained as the deepest time series of the sample are noisy, and it is arguable that they estimate $\mu_t^{(1)}$ and $\mu_t^{(2)}$. On the other hand, the centroids obtained from the median resemble better the original $\mu_t^{(1)}$ and $\mu_t^{(2)}$.

Table III summarizes the results obtained, in terms of precision, recall, L^1 distance (2) and L^2 distance (1) between real and estimated centroids. We tested different sample sizes,

[1]Available at https://github.com/hpcn-uam/deepfda-experiments

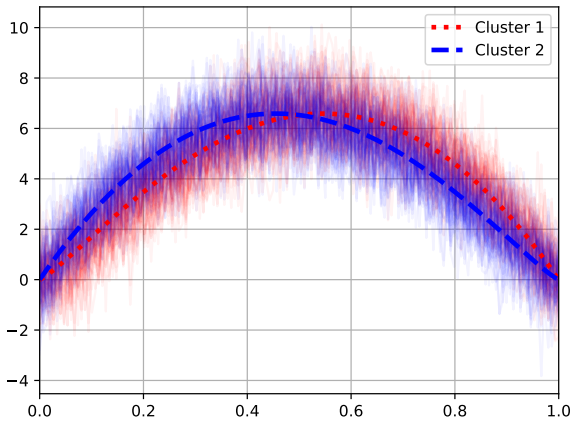


Fig. 3. Clusters and Gaussian processes of case 1.

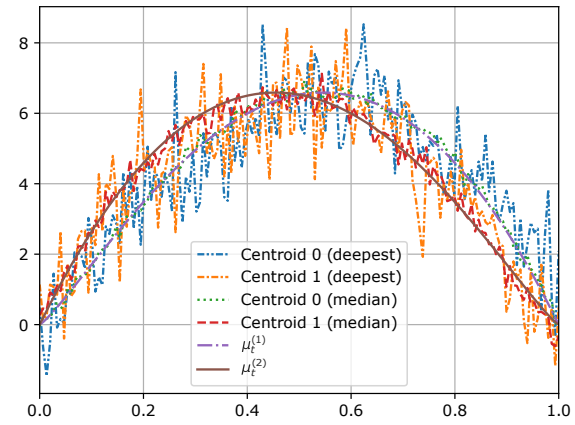


Fig. 4. Centroids obtained by the k-means algorithm for the case 2 (imbalanced clusters) for N=60.

roughly corresponding to periods of two months, three months and one year. Between the k-means approach using just the deepest or adding the median curve, we see that effect in terms of the cluster results are negligible but, given the improvement in the L^1 and L^2 distances, estimates of the centroids $\mu_t^{(1)}$ and $\mu_t^{(2)}$ are better. PCA and FPCA performed similarly in most cases and only a few slightly better scores of PCA over FPCA can be observed. This is something usual given that FPCA usually incorporates a small regularization term to guarantee that the covariance operator is positive definite, a requirement for FPCA. Besides, AE with small sample sizes do not perform well, since neural networks usually need far more samples to converge. This issue with AE can be solved by augmenting the data adding noisy samples. For higher sample sizes, performance is comparable to PCA and FPCA. Bear in mind that AE can be improved given a sufficient sample size, even if we fix the embedding dimension, thanks to the addition of more hidden layers. This means that, even if scores are worse in this case, it is a promising solution to detect imperceptible patterns.

In terms of performance, all methods but the AE can be trained with 100 curves (more than 3 months of data) in less than a second in commodity hardware (i7-1068NG7, 32 GB RAM). Training artificial neural networks, such as AE, using common libraries like Keras [46] can vary from seconds to several minutes. This system is supposed to re-compute

TABLE III
RESULTS OF THE EXPERIMENTS WITH SYNTHETIC DATA FOR DIFFERENT CONFIGURATIONS OF THE PARAMETERS

	Case 1: balanced clusters				Case 2: imbalanced clusters			
Experiment	**Precision**	**Recall**	**L^1 distances**	**L^2 distances**	**Precision**	**Recall**	**L^1 distances**	**L^2 distances**
F. k-means N=60	0.93103	0.93103	0.80747, 0.85301	1.00959, 1.06130	0.87500	0.58824	0.80747, 0.75365	1.00959, 0.94715
with median	0.93548	1.00000	0.17028, 0.20957	0.21123, 0.26858	1.00000	0.74194	0.26790, 0.55455	0.32386, 0.64678
F. k-means N=120	0.98182	0.76056	0.83224, 0.74568	1.02664, 0.95606	0.95000	1.00000	0.78501, 0.74499	0.99911, 0.93045
with median	0.92727	1.00000	0.11272, 0.16516	0.14534, 0.20065	0.47000	0.90385	0.24270, 0.65084	0.29543, 0.74889
F. k-means N=360	0.85185	0.98773	0.78501, 0.76361	0.99911, 0.98696	0.98305	0.80556	0.78819, 0.76475	0.96087, 0.91427
with median	0.95238	0.97297	0.07627, 0.07682	0.09528, 0.09744	0.98305	0.87879	0.06515, 0.13854	0.08136, 0.17489
PCA N=60	0.96774	1.00000	0.13131, 0.16403	0.16821, 0.21379	1.00000	0.75000	0.25478, 0.56055	0.29804, 0.65403
PCA N=120	0.96923	0.92647	0.09033, 0.14816	0.11298, 0.17953	1.00000	0.95238	0.07703, 0.18788	0.09816, 0.23166
PCA N=360	0.95238	0.97826	0.05821, 0.06100	0.07280, 0.07664	0.96678	0.99658	0.05155, 0.09762	0.06422, 0.12366
FPCA N=60	0.96774	1.00000	0.15670, 0.23892	0.19767, 0.29530	0.53846	1.00000	0.23362, 0.53867	0.27395, 0.61573
FPCA N=120	0.96923	0.92647	0.11445, 0.15417	0.14161, 0.18603	1.00000	0.95238	0.07760, 0.27962	0.10012, 0.33064
FPCA N=360	0.95238	0.97826	0.06451, 0.05751	0.07843, 0.07613	0.96678	0.99658	0.08739, 0.12541	0.10425, 0.15362
AE N=60	0.63636	0.61765	0.42593, 0.38448	0.49871, 0.44875	0.50980	0.92857	0.27384, 0.64494	0.33641, 0.74088
AE N=120	0.93846	0.96825	0.16077, 0.23679	0.20234, 0.28080	0.56075	0.90909	0.24730, 0.68374	0.29386, 0.77499
AE N=360	0.94578	0.96319	0.09355, 0.12453	0.11849, 0.15043	0.96429	0.80597	0.15626, 0.24205	0.19043, 0.28721

clusters and centroids at nights, when monitoring systems can spare resources for this task. Nevertheless, since each network segment has its own clustering, this can be done in parallel using computing resources in the cloud or a cluster.

B. Real-world data

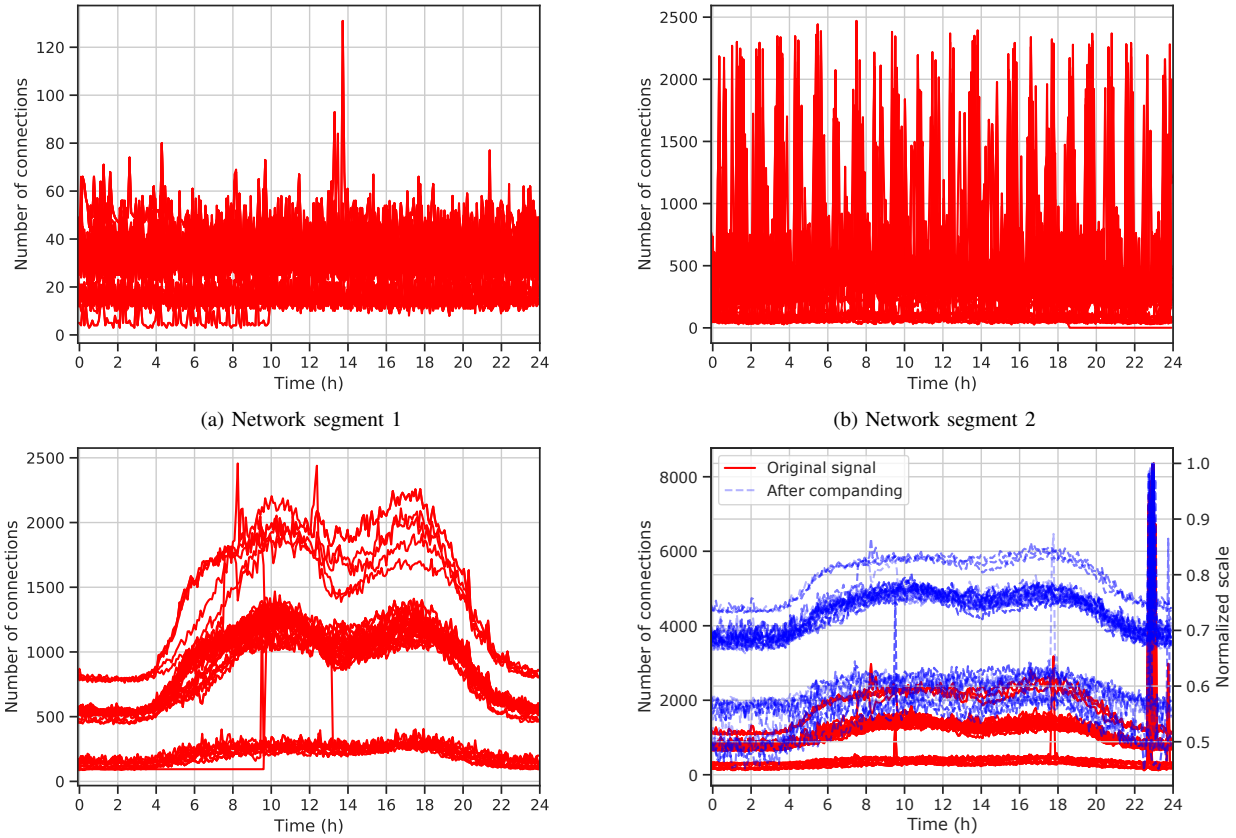
For this second scenario, we use a dataset[2] that comprises four months' worth of daily time series of the number of concurrent flows in several network segments of a multi-national company in Spain. The aggregation per network segments is performed so that services of similar nature are merged together. Although manual labeling of the data is a possibility, we approach this problem as a non-supervised one to ensure the scalability of the approach to network monitoring. Consequently, the metrics computed before are not available and we have to focus on metrics that do not rely on the ground truth. Some examples are:

1) *Functional silhouette coefficient (SC)*: using the metric defined in equation (16), we can compute not only a metric to choose the most appropriate number of clusters but also a metric to compare the result of different procedures. In this case, we need to compare all of them in the original space and not in the embedding space, since embeddings are not always available as in k-means or they do not necessarily represent the distance in the original space.
2) *Functional Davies-Bouldin Index (DBI)*: in this case, DBI [47] is based on metric properties that we also extend for the functional case. Calling the scatter of the cluster C_i
$$S_i = \underset{x \in C_i}{\text{mean}}\, d(x, \hat{\mu}_i) \tag{21}$$
and the separation of the clusters
$$M_{i,j} = d(\hat{\mu}_i, \hat{\mu}_j), \tag{22}$$
where $\hat{\mu}_i$ is the centroid of cluster C_i, we can define a ratio between the two quantities above
$$R_{i,j} = \frac{S_i + S_j}{M_{i,j}}, \tag{23}$$
which preserves properties such as being positive, being symmetric and improving in the following situations: if two clusters are equally compact ($S_j = S_k$) but one is farther away from C_i than the other ($M_{i,j} \leq M_{i,k}$), then $R_{i,j} > R_{i,k}$ and the dual one, if two clusters are equally far from C_i ($M_{i,j} = M_{i,k}$), but one is more compact than the other ($S_j \leq S_k$), then $R_{i,j} < R_{i,k}$. To measure the performance of cluster C_i, the worst $R_{i,j}$ is chosen, i.e. we define
$$D_i = \max_{i \neq j} R_{i,j} \tag{24}$$
and the functional DBI as
$$DBI = \underset{i=\{1,\ldots,K\}}{\text{mean}}\, D_i. \tag{25}$$
In contrast to silhouette coefficient, this metric relies heavily on the centroids to ensure the obtained clusters are tight and separate one from each other. Besides, it is computationally less expensive to compute since it only iterates once over the whole set of time series. Nevertheless, the more clusters we have, the better this metric scores, so it is only comparable if we fix the same number of clusters for all the methods.

From the network under study, we have selected four network segments that provide different services, which represent some of the most interesting cases that arise when facing network monitoring in real-world environments. Each time series is a full day of the metric aggregated in intervals of five minutes, and days are not necessarily consecutive.

1) *Network segment 1, survivable branch appliance*: It contains several devices responsible of the uninterrupted Voice over IP (VoIP) communications of different branches of the company. Figure 5a shows the time series of the number of connections among the day. Several trends can already be guessed but, in fact, it seems to have a stable behavior between 15 to 75 connections.
2) *Network segment 2, regional DHCP and DNS servers*: It is composed of several servers that act as DHCP or DNS of the company for a whole geographical region. Figure 5b displays the time series of the number of connections among the day. Although it seems to have

[2]Available at https://github.com/hpcn-uam/deepfda-experiments/tree/master/dataset



(a) Network segment 1

(b) Network segment 2

(c) Network segment 3

(d) Network segment 4

Fig. 5. Network segments daily time series for three months.

a very bursty behavior, it is periodic and has to do with the synchronization with the rest of regional servers.

3) *Network segment 3, payment gateway 1*: It is composed of only one server that acts as gateway for the payment mechanism, providing not only connectivity to bank companies but also security mechanisms to protect the data of the transactions. Figure 5c depicts the time series of the number of connections among the day. Clearly, it exposes three different trends that are stronger in the peak hours of activity of the day.
4) *Network segment 4, payment gateway 2*: Like the previous segment, it is composed of only one server that acts as gateway for the payment mechanism. In Figure 5d, the time series of the number of connections among the day can be observed. Although it is related to a payment system as before, we see a rather different case. Some similarities are still present such as the three trends and the peak hours of the morning and the afternoon, but the peak activity at the last hour of the day is the most remarkable event that makes it difficult to differentiate the three existing trends. Because of this, we employ the aforementioned companding procedure using the μ-law with $\mu = 1023$, which makes the apparent clusters more spaced apart.

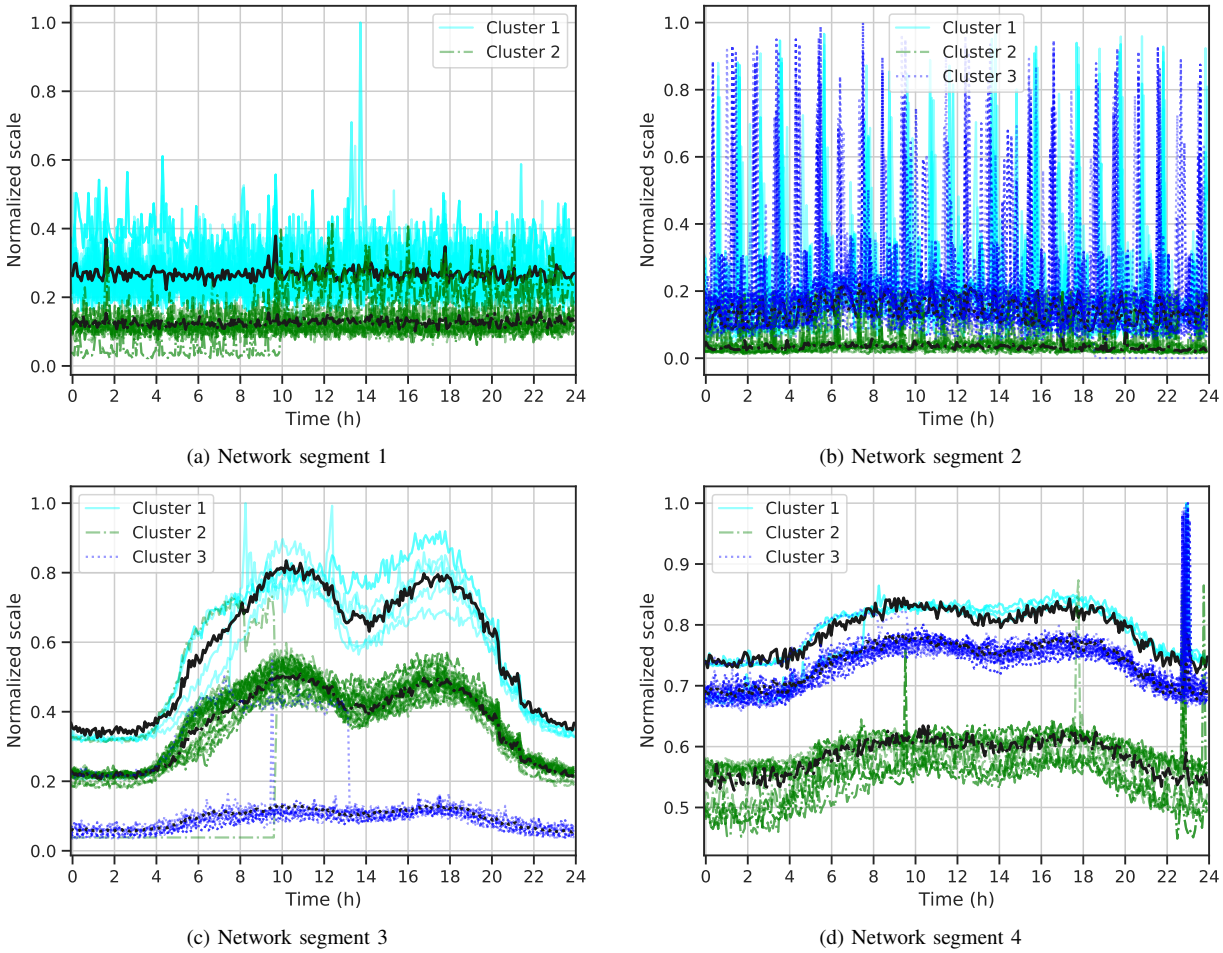
For all the network segments described above, we have fitted the methods described in previous sections. Results in terms of **SC** and **DBI** are shown in Table IV. As a reminder, **SC** does not take into account centroids, so it is common to have ties if two methods formed the same clusters. Thus, we must look at the **DBI** to distinguish which algorithm scores better. Additionally, **DBI** is always better if the number of clusters is high, so we must fix the number of clusters, which is done by voting among the decision given by **SC** for all the classifiers. K-means algorithm is usually better than projection-based algorithms, which is reasonable since projections reduce the amount of information, whereas direct approach can work with the whole signal. Difficult cases as network segments 2 and 4 are particularly hard for PCA or AE, which can be due to the loss of information of the projection and the small number of samples for training. It is remarkable that in case of network segment 4, FPCA scores better than PCA, given that it is able to differentiate better the two clusters that were very close, as we will cover next.

For each network segment, best results are shown in Figure 6. For the case of network segment 1, it can be observed in Figure 6a that two clusters were obtained. These clusters are clearly *low load days* (holidays, weekends, days with less people working) and *workdays*. In this case, we observed that both patterns are stationary and the use is homogeneous along the day. In addition, centroids visualization allows us to see the central behavior separated from the random component. In this example, we see that the clusters help us not only to detect different trends but to see that the deviation of the noise also depends on the cluster, e.g. low load days are more regular than workdays, where we see strong deviations, even doubling the number of connections in short periods. For low-load days, time series can also jump to levels similar to

TABLE IV
RESULTS OF THE CLUSTERING METHODS FOR THE REAL-WORLD DATASET FOR SEVERAL SIZES OF DATASET

	Network Segment 1		Network Segment 2		Network Segment 3		Network Segment 4	
Methods	**SC**	**DBI**	**SC**	**DBI**	**SC**	**DBI**	**SC**	**DBI**
F. k-means (1 month)	0.61714	0.47776	0.42202	0.80097	0.89050	0.11431	0.82868	0.20817
with median	0.61714	0.41333	0.49735	0.76369	0.85614	0.17242	0.82868	0.22088
F. k-means (2 months)	0.59032	0.65805	0.32844	1.05486	0.84572	0.20396	0.83508	0.25474
with median	0.59032	0.57102	0.37922	0.87207	0.84572	0.18104	0.83508	0.25324
F. k-means (3 months)	0.57107	0.66552	0.29950	1.22141	0.86293	0.21974	0.78822	0.30459
with median	0.57107	0.59909	0.34325	1.24314	0.86293	0.21802	0.77675	0.29689
PCA (1 month)	0.61714	0.54452	0.49735	1.00426	0.89050	0.21448	0.85415	3.76542
PCA (2 months)	0.59032	0.59857	0.37978	1.31938	0.84572	0.22227	0.83508	3.89442
PCA (3 months)	0.57107	0.61461	0.36283	1.54504	0.86293	0.22415	0.81444	3.94855
FPCA (1 month)	0.61714	0.56901	0.45510	1.72075	0.89050	0.23799	0.85415	0.26204
FPCA (2 months)	0.59032	0.61359	0.37978	2.03991	0.84572	0.28235	0.83508	0.29437
FPCA (3 months)	0.57107	0.59858	0.34250	1.82842	0.86293	0.26692	0.81444	0.32805
AE (1 month)	0.61714	0.81439	0.47984	4.58737	0.89050	1.85030	0.77582	6.93602
AE (2 months)	0.59032	0.58760	0.34726	2.30413	0.84487	0.30296	0.75568	2.62730
AE (3 months)	0.55785	0.60325	0.33632	2.23477	0.86293	0.23725	0.74551	5.33807

* SC: More is better. DBI: Less is better.



(a) Network segment 1

(b) Network segment 2

(c) Network segment 3

(d) Network segment 4

Fig. 6. Clusters for each network segment. Best centroids are represented as wider black lines with the same line style.

the cluster 1 from 10 to 22 h. Other state-of-the-art systems would incorrectly consider that, when time series of cluster 2 intersects the centroid of cluster 1, they are just one cluster. As we consider the global behavior instead of the local one, we are able to detect properly that there are two global types and, even if they intersect at some time interval, they have different characteristics, as for instance, the dispersion level, in contrast to other state-of-the-art alternatives [24].

On the other hand, the case of network segment 2 is completely different, as it can be seen in Figure 6b. We see three different clusters: the *low load one* with almost no peaks, and two *high load* ones with peaks, where the only difference between them is a small shift in time. Since these servers depend both on human behavior (DNS queries, DHCP renovations) and automatic behavior (synchronization among regional branches of the company), the series are

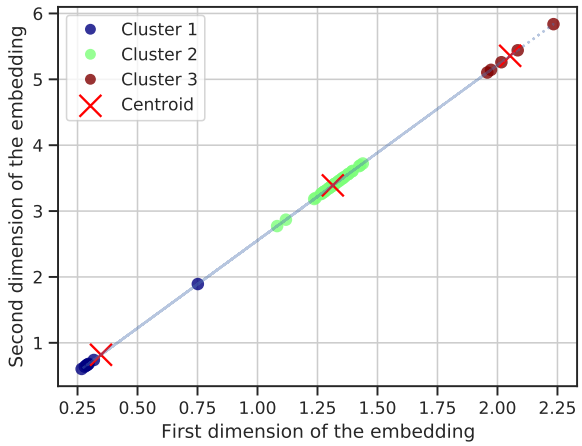


Fig. 7. Embedding space for network segment 3 obtained using an Autoencoder. Linear relation is marked as a dotted line.

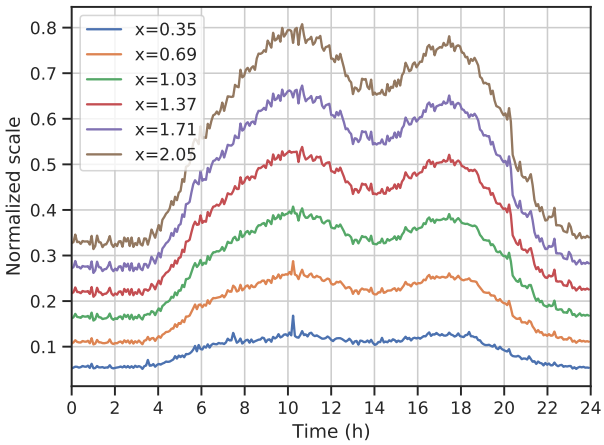


Fig. 8. Manifold representation for network segment 3 for some values of the parameter.

not stationary, but they are periodical. This shows both an advantage and a disadvantage: on the one hand, human operators would not have been able to deal with the periodicity issue and they would only consider two clusters instead of the three detected by the system. On the other hand, any trend that is not periodic every 24 hours or a multiple of 24 hours can lead to multiple clusters. Nevertheless, clusters are still usable for daily baselines and this issue will only affect the estimation errors, since cluster sizes are smaller. In other cases, frequent and significant events that happen at random times can affect negatively baseline models. It can be solved with more complex probabilistic approaches, supporting extreme events and studying their distribution [48]. The conclusion of this example is that, even in cases where bursts may affect the metric, clustering methods are able to detect trends based on the functional structure of the data.

Network segment 3 exhibits a rather different nature, as it is shown in Figure 6c. There are three clusters: one of *low load days*, where we cannot even observe a trend and almost assume it is stationary, *regular load days*, where the daily trend is remarkable, and *high load days*. For high load days, we could barely see four examples in two months, which means that our methods were able to detect even low probability tendencies that are significantly apart from the most common ones. This case proves the usefulness of this approach. One of the motivations was to help in the definition of baselines by grouping similar trends. In this case, operators were unable to estimate when a day belonged to cluster 1, since it has some random and user-dependent behavior. Our system provides a baseline for these days, preventing false alarms and improving monitoring capabilities.

Network segment 4 shows a similar case but with significant differences, as it is depicted in Figure 6d. Although some similarities are clear, such as the shape, it is noticeable that *high load* and *regular load* clusters are much closer than before, as well as the periodical bursts that happen at the end of the day, but only in two of the three clusters. This shows that companding and preprocessing methods help us to separate the clusters, improving the results and that such techniques can detect periodical phenomena that affect the time series every single day.

As we have shown in the previous two cases, similar services may behave very different not only in terms of magnitude but also in terms of the shape. In fact, this predominant shape of network segment 3 is detectable also in the embeddings. Figure 7 is an example where the embedding of an AE provides a stronger representation. As we see, data has restrictions and the dimension of the data is only one. This means that this type of representation does not only help to find clusters but also to find the real dimension of the data. In this case, this means that, if the decoder part of the AE is called $\phi(x, y)$ and the line equations is $y = mx + b$, then we have that

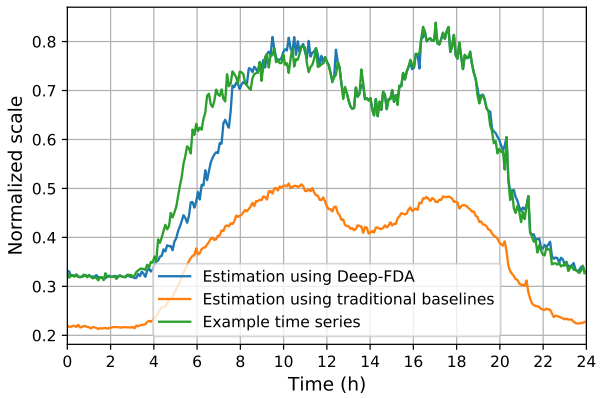
$$\Phi(x) = \phi(x, mx + b) \tag{26}$$

is a parametrization of a manifold of dimension 1 in $L^2([0,T])$, which means that our functional data, which presumably had infinite dimension, indeed it has only dimension 1 and it can be represented by just the first component obtained in the embedding of our AE. Figure 8 contains the result of $\Phi(x)$ for several values of x. It is easy to see that this parametrization is more than just a scaling or a shift, since higher values of the parameter makes the trajectories more variable. For network management, this is extremely useful since it provides a structure of the time series that should be observed, meaning that deviations from this structure should be reported as incidences to network operators.

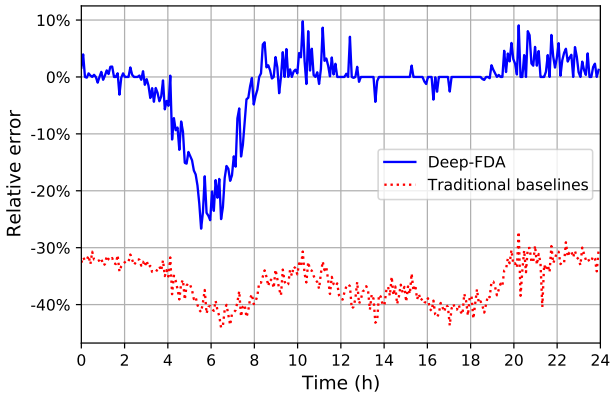
C. Use Cases

This subsection shows, without loss of generality, the feasibility of using Deep-FDA on real network monitoring tasks such as capacity planning and anomaly detection for alarming by means of simple examples.

As a specific use case, baselines can be used for capacity planning. First, we only consider the baseline that acts as an estimation of the time series. Figure 9a depicts in green a time series of one day not included in the training dataset from segment 3 and baselines obtained using both Deep-FDA and traditional baselines using moving median (blue and orange,



(a) Time series and baseline estimates



(b) Relative error of the estimates

Fig. 9. Example of the improvements on capacity planning using Deep-FDA using data from high-activity cluster of segment 3.

respectively). To compare the results, Figure 9b shows the relative error in both cases.

In the depicted example, it is critical to detect different trends, as resulting baselines may be distorted due to specific high-load events, such as occasional backups or equipment misconfigurations. As we can see, at busy hours (10 AM and 5 PM), traditional baselines tend to under-estimate the expected number of connections (between 30% and 40% percent of relative error). If we use such a baseline for in-advance dynamic resource provisioning, chances are that users may suffer performance issues due to a lack of resources. In contrast, the Deep-FDA baseline significantly reduces the provisioning error.

On the other hand, Figure 10 shows another time series for the number of connections for segment 3. In this case, we use the current past data to predict a point in the future. Using such data, we estimate a threshold of 10%, that is, an alarm is triggered if the relative error between the estimated value and the current value is greater than 10%. Note that defining which threshold to use is out of the scope of this work and in some cases it can only be possible with human interaction and feedback data.

In this second example, the threshold using global baselines is not precise enough due to the presence of the different trends, whereas the threshold using Deep-FDA takes only the curves that belong to the cluster of the current time series.

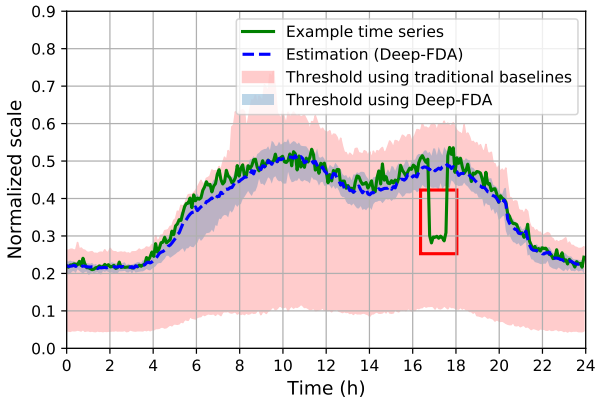


Fig. 10. Example time series and its prediction using Deep-FDA. Intervals using percentiles 10 and 90 are also presented with and without Deep-FDA. Red rectangle exposes an alarm triggered by Deep-FDA

Consequently, it is much more precise and narrow. As an example, an alarm detected by our system and not by other state-of-the-art systems is highlighted in the figure.

V. Discussion

The evaluation of Deep-FDA has illustrated the viability —through simulations and performance assessment with real-world data— and applicability —through two specific use cases for network and service management— of FDA and neural network techniques for network and service characterization based on time series classification. Some of the most significant ideas and lessons learned that contribute to the current state of the art are:

1) **Preprocessing is a key element to improve the analysis capabilities**: network segment 4 was an important use case of advanced preprocessing techniques to boost the performance of all the techniques. It is worth noting that simple outlier techniques are not always suitable, because we do not want to get rid of this data, but to see also the probability of this type of events. In this case, preprocessing techniques such as companding lead to a clearer time series without losing the ability to detect extreme events that are part of the normal behavior of the network.
2) **Mixture functional model provides a global vision of the daily behavior**: as said in previous section, the importance of the FDA approach is not really working in infinite-dimension spaces such as $L^2([0,T])$ but on using the structure of the space. Other alternatives just consider the local behavior of the phenomenon, whereas our model provides a global insight so, even if curves overlap as shown in simulations and network segment 1, it can properly detect the correct amount of clusters, the centroid and dispersion.
3) **Clustering metrics for the functional setting are useful for training and model selection**: as shown in the comparison between different methods in Table III and Table IV, proposed functional clustering metrics such as functional **SC** or functional **DBI** exposed that there is not any rule of thumb to choose one algorithm

over the rest. Each approach obtains different results and has different properties: k-means is the simplest direct approach, which usually performs well but cannot be further improved; PCA and FPCA are linear projection methods and thus, simple yet powerful representations that depend on the number of components and the nature of the phenomenon; and AE is the most powerful technique that is able to learn even non-linear behaviors, but with several parameters to consider to obtain the best performance without overfitting.

4) **FDA and neural networks can be combined**: although FDA and neural networks seem completely different approaches for time series and signal analysis, they are related, as shown in equation (15), and they can be employed together to improve the weak points of both techniques. In this case, the result of the performance of PCA or FPCA can help to dimension properly the neural network, since the simplest autoencoder is equivalent to PCA and FPCA, with the great advantage that these methods do not rely on training and convergence. Moreover, we have presented in this work a novel approach where an AE is fed with functional data, resulting in a *functional autoencoder*, namely Deep-FDA.

These contributions have proven to be useful in the presented use cases, with direct application to network management tasks such as capacity planning and anomaly detection.

VI. Conclusion

This paper has presented Deep-FDA, a novel approach that combines FDA and AE to characterize network services based on their traffic time series, mixing the potential and suitability of statistics and neural networks for data analysis to identify problems and drive management decisions. The results have shown that Deep-FDA can be a valuable technique for a better understanding of network and service management data, with several advantages over previous available methods. Specifically, we have shown two use cases, namely capacity planning and alarming, where Deep-FDA outperforms state-of-the-art approaches. In conclusion, the application of our method will help to have a deeper understanding of the network behavior, which leads to make better decision with available operational and service data. This work has grounded the basis for future works, where, following a similar approach, methodologies can be extended to multivariate time series whenever network managers consider that they should be analyzed together. Moreover, further work can focus on the limitations of baseline models to detect frequent extreme events that happen at random times.

References

[1] D. Helsper, J.-F. Huard, D. Homoki, A. Rasmussen, and R. Jannarone, "US7280988B2: Method and system for analyzing and predicting the performance of computer network using time series measurements," p. 27, 2007. [Online]. Available: https://patents.google.com/patent/US7280988B2/en

[2] Y. J. Lin, "US8606913B2: Method for adaptively building a baseline behavior model," p. 16, 2013. [Online]. Available: https://patents.google.com/patent/US8606913B2/en

[3] J. D. Brutlag, "Aberrant Behavior Detection in Time Series for Network Monitoring," in *Proceedings of the 14th USENIX Conference on System Administration*, ser. LISA '00. USA: USENIX Association, 2000, p. 139–146.

[4] A. Clemm, M. F. Zhani, and R. Boutaba, "Network management 2030: Operations and control of network 2030 services," *Journal of Network and Systems Management*, vol. 28, p. 721–750, 2020.

[5] D. Carrera, G. Casale, T. Inoue, H. Lutfiyya, J. Wang, and N. Zincir-Heywood, "Guest editorial: Special issue on novel techniques in big data analytics for management," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 797–799, 2019.

[6] R. De O Schmidt, R. Sadre, N. Melnikov, J. Schönwälder, and A. Pras, "Linking network usage patterns to traffic gaussianity fit," in *Networking Conference, 2014 IFIP*, June 2014, pp. 1–9.

[7] F. Simmross-Wattenberg, J. Asensio-Pérez, P. Casaseca-de-la Higuera, M. Martín-Fernández, I. Dimitriadis, and C. Alberola-López, "Anomaly detection in network traffic based on statistical inference and alpha-stable modeling," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 4, pp. 494–509, July 2011.

[8] M. Alasmar, G. Parisis, R. Clegg, and N. Zakhleniu, "On the Distribution of Traffic Volumes in the Internet and its Implications," in *Proceedings - IEEE INFOCOM*, vol. 2019-April. Institute of Electrical and Electronics Engineers Inc., 4 2019, pp. 955–963.

[9] C. Vega, J. Aracil, and E. Magana, "KISS Methodologies for Network Management and Anomaly Detection," in *2018 26th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2018*. Institute of Electrical and Electronics Engineers Inc., 11 2018, pp. 181–186.

[10] A. Cuevas, "A partial overview of the theory of statistics with functional data," *Journal of Statistical Planning and Inference*, vol. 147, no. 0, pp. 1 – 23, 2014.

[11] J. Ramsay and B. Silverman, *Functional Data Analysis. 1997*. Springer, New York, 1997.

[12] F. Ferraty and P. Vieu, *Nonparametric functional data analysis: theory and practice*. Springer, 2006.

[13] P. Hall and M. Hosseini-Nasab, "On properties of functional principal components analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 109–126, 2 2006.

[14] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.

[15] D. Muelas, J. E. López de Vergara, J. R. Berrendero, J. Ramos, and J. Aracil, "Facing Network Management Challenges with Functional Data Analysis: Techniques & Opportunities," *Mobile Networks and Applications*, vol. 22, no. 6, pp. 1124–1136, 12 2017.

[16] Y. Ben Slimen, S. Allio, and J. Jacques, "Anomaly prevision in radio access networks using functional data analysis," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.

[17] Y. Ben Slimen, S. Allio, and J. Jacques, "Model-based co-clustering for functional data," *Neurocomputing*, vol. 291, pp. 97–108, 5 2018.

[18] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, 12 2018.

[19] E. Jalalpour, M. Ghaznavi, R. Boutaba, and T. Ahmed, "Tmas: A traffic monitoring analytics system leveraging machine learning," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 408–414.

[20] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection," in *2019 IEEE Conference on Communications and Network Security, CNS 2019*. Institute of Electrical and Electronics Engineers Inc., 3 2019, pp. 91–99.

[21] M. Kiran, C. Wang, G. Papadimitriou, A. Mandal, and E. Deelman, "Detecting anomalous packets in network transfers: investigations using PCA, autoencoder and isolation forest in TCP," *Machine Learning*, pp. 1–17, 3 2020.

[22] Z. Chen, J. Wen, and Y. Geng, "Predicting future traffic using Hidden Markov Models," in *Proceedings - International Conference on Network Protocols, ICNP*, vol. 2016-December. IEEE Computer Society, 12 2016.

[23] M. Mouchet, S. Vaton, T. Chonavel, E. Aben, and J. D. Hertog, "Large-Scale Characterization and Segmentation of Internet Path Delays With Infinite HMMs," *IEEE Access*, vol. 8, pp. 16 771–16 784, 2020.

[24] F. S. Samani and R. Stadler, "Predicting Distributions of Service Metrics using Neural Networks," in *14th International Conference on Network and Service Management, CNSM 2018 and Workshops, 1st International Workshop on High-Precision Networks Operations and Control, HiPNet 2018 and 1st Workshop on Segment Routing and Service Function Chaining, SR+SFC 2*, 11 2018, pp. 45–53.
[25] F. Rossi, N. Delannay, B. Conan-Guez, and M. Verleysen, "Representation of functional data in neural networks," *Neurocomputing*, vol. 64, pp. 183 – 210, 2005, trends in Neurocomputing: 12th European Symposium on Artificial Neural Networks 2004.
[26] F. Rossi and B. Conan-Guez, "Functional multi-layer perceptron: a non-linear tool for functional data analysis," *Neural Networks*, vol. 18, no. 1, pp. 45 – 60, 2005.
[27] D. S. Broomhead and D. Lowe, *Radial basis functions, multi-variable functional interpolation and adaptive networks*. Malvern, Worcs.: Royals Signals & Radar Establishment, 1988.
[28] J. S. Baras, M. Ball, S. Gupta, P. Viswanathan, and P. Shah, "Automated network fault management," in *Proceedings - IEEE Military Communications Conference MILCOM*, vol. 3. IEEE, 1997, pp. 1244–1250.
[29] P. Ferreira, D. C. Le, and N. Zincir-Heywood, "Exploring Feature Normalization and Temporal Information for Machine Learning Based Insider Threat Detection," in *15th International Conference on Network and Service Management, CNSM 2019*. Institute of Electrical and Electronics Engineers (IEEE), 10 2019.
[30] International Telecommunication Union, "Pulse Code Modulation (PCM) of Voice Frequencies," *Recommendation G. 711*, nov 1988.
[31] R. R. Wilcox, "Least Squares Regression and Pearson's Correlation," in *Applying Contemporary Statistical Techniques*. Elsevier, 1 2003, pp. 173–206. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780127515410500274
[32] M. Febrero-Bande and M. Oviedo de la Fuente, "Statistical Computing in Functional Data Analysis: The R Package fda.usc," *Journal of Statistical Software*, vol. 51, no. 4, pp. 1–28, 2012.
[33] J. A. Hartigan and M. A. Wong, "Algorithm Applied Statistics 136: A K-Means Clustering Algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.
[34] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.
[35] A. Cuevas, M. Febrero, and R. Fraiman, "Robust estimation and classification for functional data via projection-based depth notions," *Computational Statistics*, vol. 22, no. 3, pp. 481–496, 9 2007.
[36] I. Gijbels and S. Nagy, "On a general definition of depth for functional data," *Statistical Science*, vol. 32, no. 4, pp. 630–639, 11 2017.
[37] A. Nieto-Reyes and H. Battey, "A topologically valid definition of depth for functional data," *Statistical Science*, vol. 31, no. 1, pp. 61–79, 2016.
[38] Y. Zuo and R. Serfling, "General notions of statistical depth function," *Annals of statistics*, vol. 28, no. 2, pp. 461–482, 2000.
[39] S. López-Pintado and J. Romo, "On the concept of depth for functional data," *Journal of the American Statistical Association*, vol. 104, no. 486, pp. 718–734, 2009.
[40] S. López-Pintado and J. Romo, "A half-region depth for functional data," *Comput. Stat. Data Anal.*, vol. 55, no. 4, pp. 1679–1695, Apr. 2011.
[41] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.
[42] H. L. Shang, "A survey of functional principal component analysis," *AStA Advances in Statistical Analysis*, vol. 98, no. 2, pp. 121–142, 4 2014.
[48] D. Perdices, D. Muelas, I. Prieto, L. de Pedro, and J. E. López de Vergara, "On the modeling of multi-point RTT passive measurements for network delay monitoring," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1157–1169, 9 2019.
[43] P. Rousseeuw and A. Leroy, "Related Statistical Techniques," in *Robust Regression and Outlier Detection*. John Wiley & Sons, Ltd, 1987, pp. 248–291.
[44] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Comput. Surv.*, vol. 41, no. 3, 7 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882
[45] C. M. Salgado, C. Azevedo, H. Proença, and S. M. Vieira, "Noise versus outliers," in *Secondary Analysis of Electronic Health Records*. Springer International Publishing, 1 2016, pp. 163–183. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-43742-2_14
[46] F. Chollet, "Building Autoencoders in Keras," *The Keras Blog*, 2016.
[47] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

Daniel Perdices (daniel.perdices@uam.es) is researcher and teaching assistant at Universidad Autónoma de Madrid (Spain) . He received the B.Sc. (Hons) degrees in Mathematics and in Computer Science (2018), the M.Sc. in Mathematics (2019) and the M.Sc. in Information and Communications Technologies (2020) and currently is a Ph.D. student, all at Universidad Autónoma de Madrid (Spain). He researches on statistics, mathematical modeling, machine learning, network traffic analysis and SDN.

Jorge E. López de Vergara (jorge.lopez_vergara@uam.es) (S'02-M'04-SM'19) is associate professor at Universidad Autónoma de Madrid (Spain), and founding partner of Naudit HPCN, a company devoted to high performance traffic monitoring and analysis. He received his M.Sc. and Ph.D. degrees in Telecommunication Engineering from Universidad Politécnica de Madrid (Spain) in 1998 and 2003, respectively. He researches on network and service management and monitoring, having co-authored more than 100 scientific papers on this topic.

Javier Ramos (javier.ramos@uam.es) is associate professor at Universidad Autónoma de Madrid (Spain). He received the M.Sc. degree in computer science and the Ph.D. degree in computer science and telecommunications from the Universidad Autónoma de Madrid, Spain, in 2008 and 2013, respectively. His research interests are in the analysis of network traffic, quality of service, software defined networks and network function virtualization.