Fault Prediction for Heterogeneous Networks using Machine Learning : a Survey

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

21-01-2022 / 27-01-2022

CITATION

Murphy, Killian; Lepers, Catherine; lavignotte, antoine (2022): Fault Prediction for Heterogeneous Networks using Machine Learning : a Survey. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.18857759.v1>

DOI

[10.36227/techrxiv.18857759.v1](https://doi.org/10.36227/techrxiv.18857759.v1)

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2021.01

Fault Prediction for Heterogeneous Networks using Machine Learning : a Survey

K. Murphy*, A. Lavignotte[†], and C. Lepers[‡]

SAMOVAR laboratory, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau

Email: *killian.murphy@telecom-sudparis.eu, [†]antoine.lavignotte@telecom-sudparis.eu,

[‡]catherine.lepers@telecom-sudparis.eu

This work was financed in part by SPIE ICS. It was also financed by the Association Nationale de la Recherche et de la Technologie through grant n°2020/1281. We wish to thank SPIE ICS for sharing their knowledge on network maintenance.

ABSTRACT Network Fault Prediction, a field three decade old, has seen a surge in scientific interest in the recent years. The ability to predict network equipment failure is increasingly identified as an effective tool to increase network reliability. This predictive capability can then be used, to either mitigate incoming network failures, or to enact preventive maintenance on incoming failures, and could enable the emergence of zero-failure networks and allow safety-critical applications to run over larger, higher complexity heterogeneous networks. After defining the key terms and describing the core concepts of Network Fault Prediction, this article shares performance parameters of a specific network and systems integrator's current operations in network maintenance, and provides a description of the Machine Learning methods used for Network Fault Prediction, followed by a survey of recent research applications in the field. Finally, this article lays out perspectives for future research.

INDEX TERMS Network fault management, network fault prediction, network reliability, network failure prediction, network maintenance, heterogeneous networks, Machine Learning, networks, survey, state of the art.

NOMENCLATURE

ARMS	Availability, Reliability, Maintainability and Survivability.
BN	Bayesian Networks.
DC	Data Center.
DC	Directly Connected.
DL	Deep Learning.
DT	Decision Tree.
FN	False Negatives.
FP	False Positives.
FCC	Federal Communications Commission.
FPR	False Positive Rate.
GAN	Generative Adversarial Networks.
HLLE	Hessian Locally Linear Embeddings.
HSDN	Hybrid Software Defined Networks.
IP TV	Internet Protocol Television.
ISPs	Internet Service Providers.
kNN	k-Nearest Neighbor.
KPIs	Key Performance Indicators.

LASSO	Least Absolute Shrinkage and Selection Operator.
LSTM	Long Short Term Memory.
LVQ-LM	Learning Vector Quantization Learning Machine.
ML	Machine Learning.
MTBF	Mean Time Between Failures.
MTTI	Maximum Time To Intervene.
MTTR	Maximum Time To Restore.
MLP	Multi Layer Perceptrons.
NFP	Network Fault Prediction.
NSI	Network and Systems integrator.
NN	Neural Network.
OSPF	Open Shortest Path First
PHM	Prognosis and Health Management.
QoE	Quality of Experience.
QoS	Quality of Service.
REP-Tree	Reduced Error Pruning-Tree.
RF	Random Forest.

RNN	Recurrent Neural Networks.
RTTF	Remaining Time To Failure.
SLA	Service Level Agreement.
SDN	Software Defined Networks.
SVM	Support Vector Machines.
TN	True Negatives.
TP	True Positives.
WDM	Wavelength Division Multiplexing.

I. INTRODUCTION

NETWORKS grow more complex, as they are constantly integrating new services, and thus increasing the cost of network service degradations. An estimate published in 2018 indicates that small enterprises lose an average of \$55,000 per year of profit due to downtime and another from 2019 estimates a loss of profit due to network downtime of \$39,900 per year per hundred users [1], [2]. Such phenomena as the emergence of Industry 4.0 [3] and cloud computing [4] demonstrate the increasingly important economic role that networks will play in the world. Therefore there is an increasing need and interest in keeping the network running with minimal interruptions [5]–[13]. As a consequence, network management tools must evolve to meet this increasing demand and integrate systems that may detect, diagnose and predict failures [14]–[16].

The field of Network Fault Management is dedicated to the growth of network dependability by detection of network faults, the analysis of the root cause of the fault, the mitigation of failures (automated or not) and the prediction of failures. Most network management systems that are deployed nowadays integrate network supervision and some aspect of automated network fault detection, and diagnosis. However few systems include fault prediction.

Network Fault Prediction (NFP) could greatly improve dependability, or reliability, of networks and systems running on them, by allowing early interventions and mitigation, whether automated or not. This allows, on the one hand, to reduce intervention and network downtime costs, and on the other hand, to provide a better Quality of Experience (QoE) to the network's users and better service overall [17]–[19]. There is also a distinct possibility in some cases to prevent the failure entirely, for example by system rejuvenation for instance as can be done for systems and applications running on network equipment (for some cases of failures) [20].

A study of NFP brings with it all the complexity inherent in the broader category of Fault Prognosis [21], but adds additional complicating factors. An example of one such being the lack of environmental controls. Distribution network equipment is often subjected to external influence (such as humidity, vibration, heat, impact, etc) since distribution network equipment is often not isolated in a technical room. These external factors constitute additional and unpredictable causes for equipment failure. We may also add that networks today are highly heterogeneous (see network heterogeneity in section II-A7), with many different sources of equipment, protocols and services interacting together. This increases

the complexity of NFP and maintenance problems by introducing more parameters to the problem [22]. Manufacturer heterogeneity in particular has been shown to increase the risk of faults [16]. Furthermore, an additional difficulty lies in the fact that new technologies are being introduced into the network, which means that as time progresses new types of faults will appear [23].

As related works, there are several studies that review the state of the art in fault management for networks [22], [24]–[28] but we have found none that specifically tackle NFP. A comprehensive survey of all the applications of Machine Learning for telecommunication networks is presented in [24], and fault management and NFP are presented in a part of the study. Mulvey et al. [28] provide a detailed survey on the use of Machine Learning techniques for cellular network fault management. The different Machine Learning techniques and their advantages for the usage are presented and future research directions are given. Cherrared et al. [25] present the state-of-the-art of fault management techniques in Network Virtualization Environments. The following papers all deal with Software Defined Networks (SDN) [22], [26], [27]. Xie et al. [27] present a comprehensive survey of Machine Learning techniques applied to SDN, and briefly explains fault management. Both Fonseca et al. [22] and Yu et al. [26] describe fault management in SDNs but do not address NFP.

The objective of this article is to provide an overview of the state-of-the-art of the methodology and outcomes of studies in NFP using ML and a description of the ML tools used in the process as there has been none yet published to our knowledge. The main contributions of this article are as follows.

- Definitions and core concepts of NFP.
- The expertise of a network and systems integrator in network maintenance and its application to NFP.
- A systematic review of the Machine Learning methods that were used in the field, with a description of the algorithms.
- A survey of the state of the art of the methodology and results of NFP studies from 1997 to 2021, broken down by prediction type.
- Future research perspectives in the field.

Throughout this article, the core definitions and concepts of NFP used are shared in section II, then an industrial actor's expectations and needs based on its expertise are presented in section III, the Machine Learning methods that were used in the field are described in section IV, in section V the different applications of NFP to date are detailed, and perspectives for NFP are presented in section VI.

II. CORE DEFINITIONS AND CONCEPTS

In this section we provide a definition for the terms and the key concepts described in this paper.

A. DEFINITIONS

1) *Network fault:*

Network fault is defined as a state of a network or network equipment where a deviation exists from normal operational conditions. This state may devolve into a network failure.

2) *Network failure:*

Network failure is a state of a network or network equipment where it fails to perform its purpose as intended. Failures may occur following different time spans. *Complete Failure* may be reached where the failure state prevents the entire equipment or network from performing its task [21].

3) *Network fault management:*

Network fault management describes research and processes specific to dealing with faults and failures in networks. Sub-components of Fault Management are *fault detection*, *root cause analysis/localization*, *fault mitigation* and *fault prediction (NFP)*.

4) *Network Fault Prediction (NFP):*

Network Fault Prediction is the process by which incoming future failures is predicted. This process is based on past and present knowledge of the network state, gained by monitoring the network state.

5) *Network monitoring:*

Network Monitoring is the practice through which a service provider will monitor the operational state of the equipment constituting the network.

6) *Network dependability:*

Network Dependability defines how much a network can be trusted to operate under any circumstance, and how it performs when a failure occurs. It is described along four axis which are Availability, Reliability, Maintainability and Survivability (ARMS) [29].

7) *Network heterogeneity:*

Network Heterogeneity is the presence on the network of different systems that do not follow the same rules. This can be due to the equipment being from different manufacturers, using different technologies, using different protocols. It is accepted that as networks grow more heterogeneous, maintenance and failure prediction becomes more complex [22], an example in optical networks being alien wavelengths appearing due to interoperability issues [30].

8) *Time ahead:*

The time ahead of a prediction is defined here as the minimum amount of time between the moment the prediction was made, and the interval of time where the failure occurs. It can be seen as the 'X' in Fig 2.

B. CONCEPTS

1) NFP

There are usually two alternative ways when trying to predict incoming Network Failures.

We may choose to predict the Remaining Time To Failure (RTTF) of the equipment or network, which is a numerical metric. This is a more classical approach in the broader field of Prognosis and Health Management (PHM), where one would try to predict the Remaining Useful Lifetime of the equipment. In this case the problem becomes a supervised (we usually have the failure data) regression problem. To illustrate the case of the RTTF prediction, we show in Fig 1 a representation of the evolution of a generic prediction of RTTF over time.

However, in the field of networks, predicting health state of the equipment or network during a specific interval of time in the future is more prevalent. This is a categorical value, which makes the problem a supervised classification problem where we usually predict the probability of an unhealthy state with a decision threshold. We illustrate this case with Fig 2. In this case, the predicted value always concerns information at the same approximate distance in time X compared to the present.

In both cases however, different studies have wide disparities in how they set the values of some variables :

- The sources from which the prediction is made. It can be data from a single or several pieces of equipment of the network, a single or several applications running on the network, a subnet or even the whole network.
- The types of faults considered. It could, for example, be hardware failure, application failure, or protocol failures.
- The time ahead gained by the prediction. This time needs to be sufficient for the prediction to be useful (e.g. automated system rejuvenation, or logistics for replacement) while permitting high prediction performance so that decisions can be taken based on the prediction.
- The data type. Different types of data contain different levels of information at different intervals in time, which may severely impact the performance of the prediction system depending on the time gained that is targeted. Common examples of monitoring include SNMP (Simple Network Management Protocol), syslog, and Net-Flow.
- The type of prediction that is made : health state; RTTF; or other.
- The adequate prediction metrics and the expected performance levels associated with them. Some applications may need good recall but could let precision be lower for example (see subsection II-B2).
- The width of the interval in the case of prediction of the health status of a network part. The width will have an impact on the precision of the information gained, but also on prediction performance (the wider the interval, the higher the probability that it will contain a failure).

FIGURE 1: Approximated behaviour of an RTTF prediction. The predicted RTTF will show more uncertainty and variance the further away from complete failure, and (ideally) the closer in time we are to the failure, the closer the predicted RTTF will align with the real RTTF. Then a cutoff threshold can be determined where the predicted RTTF is trustworthy enough compared to the cost of intervention, to make the decision to intervene.

FIGURE 2: Generic failure prediction. A prediction decision is made at time T for the time interval of $[T+X; T+X+Z]$ of duration Z . Decision thresholds, X and Z to be determined, depending on necessary prediction performance for the context.

Once those parameters are set and a prediction decision is taken based on a threshold value, we can try to evaluate the performance of the model.

2) Measuring Prediction Performance

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

TABLE 1: Classification of prediction results.

Usually when considering performance for network fault prediction (in the categorical mode), precision, recall, and

false positive rates are considered. Accuracy or F1-score can also be measured sometimes.

In order to calculate these prediction metrics, we compare the prediction results to the ground truth (actual values) and classify them accordingly following Table 1. We then use the number of elements of each class to calculate the following metrics.

Precision is defined as the proportion of cases where the model predicted the positive class accurately (True Positives), relative to the number of cases where the model predicted a data point as part of the positive class, irrespective of its correctness (True Positives + False Positives).

$$Precision = \frac{TP}{TP + FP}$$

where TP =Number of True Positives and FP =Number of False Positives.

Recall is defined as the proportion of cases where the model predicted the positive class accurately (TP), relative to the total number of elements of the actual positive class (True Positives + False Negatives).

$$Recall = \frac{TP}{TP + FN}$$

where FN =Number of False Negatives.

False Positive Rate (FPR), is defined as the proportion of cases where the model incorrectly predicted a member of the negative class as part of the positive class (False Positives), relative to the number of elements of the negative class (False Positives + True Negatives).

$$FPR = \frac{FP}{FP + TN}$$

where TN =Number of True Negatives

F1-score is a composite metrics based on Precision and Recall. The F1-score can only be high if both Precision and Recall scores are high, as can be seen in the formula below. Therefore it is often used. Another perspective is that maximizing the F1-score of a predictor minimizes both the number of False Positives and False Negatives.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Accuracy represents the proportion of correct predictions (positive or negative) relative to all predictions made. This can be a good metric in cases where the classes are balanced.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

III. EXPECTATIONS OF AN INDUSTRIAL

One of the practical applications of Network Fault Prediction is for Internet Service Providers (ISPs) or Network and Systems Integrators (NSIs) that propose maintenance in operational condition services to predict incoming failures to enhance their failure response performance. Also, ISPs need to maintain their own network infrastructure in operational condition in order to provide their services to their clients and ensure the best QoS.

In this case, the ability to predict the occurrence of network failure could procure a competitive edge in the maintenance of the network, to reduce operating costs and increase client satisfaction. In this section, we describe the necessary performance constraints that would be required of an NFP system, in order to be maximally useful for NSIs to maintain network operation. Technical input that was collected from an industrial actor in the field was used as a reference.

NSIs respond to their clients' network needs and integrate the best solution for them. Depending on the contracts, they are often also in charge of maintaining the clients' infrastructure in operational condition. The Service Level Agreement (SLA) that they sign with each client determines the conditions they must abide by in order to maintain contracted network performance. In case they do not respect these conditions, they expose themselves to penalties that can be very costly in certain cases (for example in critical infrastructure).

The following information is gathered from a survey of professionals within SPIE ICS, an NSI. We present this information with the objective of giving a grounded practical view of NFP, but naturally the values vary depending on the context and the company and clients involved.

Most contracts have different priorities for different network services and/or equipment, usually separated into three levels of priorities P1, P2, P3, from highest to lowest. The SLA stipulates the Maximum Time To Intervene (MTTI), and Maximum Time To Restore (MTTR) for network failures that the integrator must respect, and penalties to be paid when they do not respect the agreed-upon delays. The penalties are usually proportional to the excess time taken in restoring the network and they have a severe impact on the maintenance contract margin. Additionally not respecting the SLA delay has a large impact on customer satisfaction.

These delay specifications depend on the priorities, and usually in SLAs, P1 equipments have an MTTI of around 20 minutes, meaning that the integrator must have someone starting to work on the failure 20 minutes after it is first detected (if the detection of the failures is incumbent on the operator) or after the client first declares it. The MTTR is usually about 4 hours for P1 failures, meaning the integrator must restore the failing service/equipments in less than four hours after detection/declaration of the incident (with the possibility of allowing for distinction between different types of working hours such as day-shift, night-shift, weekends, etc).

Therefore, system integrators have a vested interest in

using fault prediction models, as an effective and reliable prediction would enable them to gain more time to respond by taking a decision for early fault resolution. This would allow the NSI to be more competitive on the SLA terms offered, retain more margin by eliminating cases that exceed contractual MTTR, possibly preventing some types of failure from occurring and reducing maintenance costs. As the solutions improve, they could potentially also offer zero-failure network services with a network architecture that does not have excessive redundancy.

Based on the previous information and insight into the needs and logistics involved in resolving network failures, a few criteria have been derived in order to measure usability of the different proposed models, in the environment of a maintained heterogeneous network. These elements are illustrated in Fig 3.

- T_{ad} : Does the prediction allow enough time to run automatic preemptive diagnostics and mitigation on the networks ?
- T_{md} : Does the prediction allow enough time to run manual preemptive diagnostics on the networks ?
- T_{mm} : Does the prediction allow enough time to enact preemptive mitigative measures on the networks ?
- T_L : Does the prediction allow enough time to deploy necessary logistics to replace the equipment before failure occurs ? Logistics constraints may vary greatly depending on time of day, geographical location of the failure and the NSI actor.

The necessary time to run these diagnostics and actions depends on the state of the network and the weight of the operations that are run. However based on the professional experience of SPIE ICS, the following approximations can be given :

- $T_{ad} \approx 5 \text{ min}$
- $T_{md} \approx 30 \text{ min}$
- $T_{mm} \approx 45 \text{ min}$ (manual diagnosis + 15 minutes)
- $T_L \approx 2 \text{ hours } 30 \text{ min}$ (manual diagnosis + 2 hours)

We will also try to consider the additional information that the prediction brings along with it about what type of failure is about to happen.

Finally, to determine the trustworthiness of the prediction, in order to make business and logistics decisions, we consider the Precision, Recall, and False Positive Rates (FPR) of the predictions (see II-B2).

IV. ML METHODS FOR NETWORK FAULT PREDICTION

In this section we describe the ML methods that were used in the field of NFP up to this point. We have chosen to consider only ML based methods, as from what we have seen, they are the more representative, and they more show promise for evolution and performance in the long run for NFP than other known statistical methods. We have summarised in Table 2 the different ML methods that have been used in the studies that we review.

However, there are several works in Network Fault Prediction among which [18], [36], [61]–[65] which treat the NFP

problem without using ML methods and can show interesting results. One example is the case of Hood et al. [61], where they show that it may be possible to predict network failure several hops away in space and around 10 minutes away in time (further testing and specification needed). A hop is a network distance metric defined by the minimal number of links a message has to cross to reach destination from source. Two Directly Connected (DC) routers are 1 hop away.

Generally Fault predictions tasks in the PHM field are separated into two stages, constructing a failure model and the prediction stage. However, when we use ML methods, both stages can be considered to be joined. Indeed, to construct the failure model you need to train the model by predicting failures, and while the model is in operation it is desirable to continue training the model in case new types of fault arise.

A. LINEAR MODELS

Two tools taken from statistics, **Linear Regression** [33] and **Logistic Regression** [35]. Linear Regression is equivalent to fitting a straight line through the data, and Logistic Regression is equivalent to fitting a straight line, as a decision boundary, to separate two classes of data.

Both of these models are very simple and quick to execute, and the result is readily explainable.

However, they can only separate linearly separable data and usually do not work well when the data is complex.

Learning Vector Quantization (LVQ) Learning Machine [37] is a classification algorithm similar to the k-Nearest Neighbor (kNN) algorithm. kNN saves all the training data as examples and counts the class representation of the k (odd number) examples that are closest in vector-space to the sample in order to predict its class. LVQ Learning Machine operates in a similar way but only measures proximity to a selected few (compared to dataset size) codebook vectors to take the classification decision. It is a very quick, simple, and explainable model, and its capacity increases with the number of codebook vectors selected. However it has a low capacity for non-linear problems.

B. PROJECTION BASED ALGORITHMS

Support Vector Machines (SVM) [40]–[42], [66], also called Vast Margin Separators, are a type of supervised learning model that functions under the following manner. The data is first projected through a kernel function into another space (of possibly different dimension). The kernel function can be chosen to best fit the data, or separate the data, if there is *a priori* knowledge of the structure of the data and of the pertinence of elements to the result that is sought after. Then a hyperplan of the new space is selected so that it separates the data with the widest margin possible.

This type of model works well with very high dimension data. It can also separate both linearly and non-linearly separable data if the adequate kernel function is chosen and kernels can be made explicitly to fit the data, with some knowledge of the data structure, and therefore reach very high performance (requires a high level of expertise). It is

FIGURE 3: Two possible timelines for failure prediction response. In the case of manual preventive mitigation, it is unknown whether the mitigation will have an impact on the emergence of the failure, therefore it may not be necessary to engage logistics.

also less prone to overfitting than most types of models as generalization is built-in to the model (through seeking a large margin).

However, it usually does not converge or breeds poor results if the kernel function does not fit the data, and it is possible that no simple kernel function will fit the data. Because of that, with no *a priori* knowledge of the data structure, after trying the few wide spread kernel functions (linear, RBF, polynomial, sigmoid) it can be advised to try ML methods other than SVM.

Hessian Locally Linear Embeddings (HLLE) [44] is a dimension reduction projection aiming to preserve distance between points locally. This dimension reduction operation aids in reducing complexity of problems, however, in some cases, this step is unnecessary.

Least Absolute Shrinkage and Selection Operator (LASSO) as predictor [45] is a regression algorithm. This model performs variable selection (based on covariance) and regularization, before performing prediction. This model provides explainability by dimension reduction, and greater prediction performance than simple regression. However it does not perform well in very high dimension environments.

C. DECISION TREE-BASED ALGORITHMS

All of the following methods are based on Decision Trees.

Decision Tree (DT) algorithm [48] is a succession of (if, else) statements, feature per feature arranged in the form of a tree. Each time, the feature is selected by choosing the one allowing maximum entropy gain by a simple threshold separation of the data along this feature. The selected feature and the threshold separation are included in the DT before repeating the step to continue forming the DT. This method yields very explainable results. Each DT has contained within the information of each split along with its entropy levels and the quantity of data of each class remaining. This information can be used to generate graphical aids. However, as the split made is dependent on the data used, DTs usually have high variance in decisions depending on data and so a poor generalization capacity. Also when the data is complex, DTs

tend to have poor performance. To counter these weaknesses and improve performance, other algorithms have been built by aggregating DTs in certain manners.

Random Forest (RF) [47] is an ensemble method on Decision Trees trained by bagging, or Bootstrap Aggregating. Each DT is trained on a different subset of the data, and the final prediction for each input sample is the average or most voted decision out of all DTs trained. In the RF algorithm, the different DTs are also trained on different subsamples of the features so that each DT is more independent. This model has the advantage of working well without much pretreatment and is moderately explainable, but it does not work well on non-linearly separable data, and setting the forest and tree sizes requires a lot of tweaking.

AdaBoost [50] is another ensemble method on Decision Trees trained by boosting and adapting voting rights to amplify the importance of stronger weak learners. Boosting is the process in which you give more importance to training samples where the previous model was wrong so that the new weak learners that you train are better trained for these types of samples. It is known to increase performance when having many weak classifiers (DTs), however the algorithm needs these weak classifiers to have less than 50% error and be as independent as possible.

XGBoost [51], another ensemble method for Decision Trees trained by bagging, boosting, adapting voting rights. There is the added step of pruning to remove the weaker branches DTs and weaker weak learners (DTs). It is known to have great performance, very high adaptability, but needs correct calibration of hyperparameters which can be a lengthy process.

Reduced Error Pruning-Tree (REP-Tree) [52] is a pruning method for DTs. It learns to split tree leaves by reducing prediction error step by step on a subsample of the training set. Then it applies a pruning mechanism where another subsample of the training set is used to evaluate the splits and the branches. Poorly generalisable branches and splits are removed. It is used to reduce the high variance problem of DTs.

Model type	Model name	Type	Examples in NFP	Litt.	Description	Advantages	Disadvantages
Linear Models	Linear Regression	Regression	[31], [32]	[33]	Fit a line through the data	Simple, quick	Linear, doesn't fit complex data well
	Logistic Regression	Classification	[34]	[35]	Place a decision threshold based on a fitted line through the data	Simple, Quick	Linear, doesn't fit complex data well
	LVQ Learning Machine	Classification	[36]	[37]	Similar to k-Nearest Neighbor algorithm, measures proximity to selected few (compared to dataset size) codebook vectors to take classification decision	Less complexity than k-NN, simple, explainable	Low capacity for non-linear problems
Projection based algorithms	SVM	Classification/Regression	[31], [32], [38], [34], [39]	[40]–[42]	Project data in other space through kernel function then choose largest margin separator threshold	Quick, can fit both linear and non-linear	Doesn't converge if kernel isn't good enough
	Hessian Locally Linear Embeddings (HLLE)	Dimension reduction	[43]	[44]	Dimension reduction projection aiming to preserve distance between points locally	Dimension reduction aids in reducing complexity of problems	Sometimes not useful
	LASSO as predictor	Regression	[31]	[45]	Model that performs variable selection (based on covariance) and regularization	Explainability by dimension reduction, greater performance than simple regression	Doesn't perform in very high dimension environments
Tree- based	Random Forest	Classification/Regression	[39], [46]	[47]	Collection of Decision Trees trained by bagging on different subsamples of the features	Moderately Explainable, works well without much pretreatment	Doesn't work well on non-linearly separable data, setting forest and tree size
	Decision Tree	Classification/Regression	[39]	[48]	Collection of (if, else) on a feature statements arranged in the form of a tree, feature decided by maximum information gain by split	Very explainable, entropy information included	High variance in decisions depending on data
	AdaBoost	Classification/Regression	[49]	[50]	Collection of Decision Trees trained by bagging, and boosting and adapting voting rights to amplify important of stronger weak learners	Known to increase performance of many weak classifiers (DTs)	Need weak classifiers to have error < 50%
	XGBoost	Classification/Regression	[39]	[51]	Collection of Decision Trees trained by bagging, boosting, adapting voting rights, and pruning to remove the weaker branches and weaker weak learners	Known to have great performance, very high adaptability	Needs correct calibration of hyperparameters
	Reduced Error Pruning- Tree	Classification/Regression	[31]	[52]	Method for pruning DTs by using other subset of dataset to take the pruning decision	Reduces dataset subset variance influence on pruning	
	MSP	Regression	[31]	[53]	Use of DTs as a regressor, learns to split tree leaves by reducing error of regression step by step	Explainability, equations included, easy to implement	Low capacity for non-linear problems
NN	Multi-Layer Perceptron	Classification/Regression	[29], [32], [36], [39], [54]	[55]	Collection of artificial neurons, with one input layer, one output layer, and any low number (otherwise Deep Learning) of hidden layers	Can achieve great performance, network problem capacity increases with size	Poor explainability, quite susceptible to the bias/variance problem
	Auto Encoder	Dimension reduction	[32]	[56]	Deep Learning NN architecture in the shape of an hourglass. Designed for dimension reduction while conserving information.	Has been shown to generate great dimension reduced data representations in many examples	Resource consuming
	Recurrent NN (LSTM)	Classification/Regression	[57]	[58]	NN model designed to take sequences of input and propagating the context through the treatment of the sequence.	Improvement of classical RNNs (vanishing gradient problem), usually good performance compared to simple feedforward techniques	Resource consuming
	Generative Adversarial Networks (GAN)	Classification/Regression	[59]	[60]	Double network used to generate additional coherent data. One network is trained to create false examples resembling real ones. Other is trained to discriminate	Great for coherent data generation	Resource consuming

TABLE 2: Description of ML methods used in the field of NFP

MSP [53] uses DTs as a regressor. This algorithm learns to split tree leaves by reducing regression error step by step on the training set. A step of pruning is set after that to remove branches that contribute most to regression error. This algorithm has high explainability, as the equations are included, and it is easy to implement. However it has low capacity for non-linear problems.

D. NEURAL NETWORKS (NN)

Neural Networks [67] using Backpropagation are the most famous branch of ML currently. Similar to DT-based algorithms, they are based on a nuclear function, the artificial neuron, that is arranged into a structure. The function of the neuron is simply a weighted sum of all inputs (a bias input is added) passed through an activation function. There are several commonly used activation functions, and when the output of the activation function passes a threshold, the neuron is excited, otherwise inhibited.

In **Multi Layer Perceptrons (MLP)** [55], neurons are stacked in several hidden layers between an input layer and an output layer. This type of model can achieve great performance, as the networks problem capacity increases with the number and size of hidden layers. It is also highly versatile in that shapes can be completely customized. However as a consequence, it has very poor explainability, though there have been works in this field recently, and it is quite susceptible to the bias-variance problem.

Auto Encoders [68] are a class of Deep Learning (DL) designed for dimension reduction. Its architecture is in the shape of an hourglass and the expected output for training is the same data as the input. Therefore the model learns how to generate a decent dimension reduction from the input to the middle hidden layer, and learns to reconstruct with the lowest error possible the input from the middle hidden layer. It has been shown that using the dimension reduced/encoded inputs for other problems can increase performance. However, as

it belongs to the class of DL, it is known to be resource consuming.

Recurrent Neural Networks (RNN), and in this particular case, **Long Short Term Memory (LSTM)** [58] are a type of NNs. In RNN, the model is fed a sequence of inputs (can be thought of as another dimension to the data, such as time) and it is engineered to use the information from previous samples of the same sequence in addition to current input to generate an output. LSTM are considered an improvement of classical RNNs where the vanishing gradient problem is addressed (though not completely solved) by incorporating an additive operation into the model. This type of model usually has good performance compared to simple feedforward techniques and has the capacity to tackle issues hard for simple feedforward techniques. However it is also very resource consuming.

Generative Adversarial Networks (GAN) [60] are a type of architecture used for coherent data generation (fake generator) and for false data detection. It is done by employing two NNs. One network is trained to create false examples resembling real ones, and the other is trained to discriminate between real examples and examples generated by the other model. GANs have a lot of potential uses and could generate very useful coherent data but the training process is quite resource consuming.

V. APPLICATIONS OF NFP TO NETWORKS

In this section we describe the different studies that have been realized in NFP, classified by the application of the prediction. We have regrouped in Table 3 these different studies per application.

A. EQUIPMENT HEALTH

An equipment health prediction is the prediction of the health state of the equipment in time X , for duration Z , or for the interval :

$$[t + X; t + X + Z]$$

This falls into the category of classification problems. For reference, in all the papers mentioned, healthy is negative and faulty is positive.

Lu et al. [43] work on the prediction of network failures in a controlled Local Area Network (LAN) environment. They recognize that online (meaning *in real-time*) failure prediction has the drawback that it needs a lot of specialized knowledge in order to extract and adequately format features necessary for prediction.

In order to address this issue, they propose an autonomous system for feature selection. They aggregate six sources of data and use Hessian Locally Linear Embeddings (HLLE), then Supervised HLLE (SHLLE) - that uses class information to better separate data in the embedding space - to select the appropriate features for optimal prediction performance.

They use their data to predict Network, CPU, and Memory failures coming from faults they inject into the network. We may note that injecting the failure might not reproduce the

reality of the network state in a naturally occurring fault, and so the network state might be missing or showing elements that make the prediction easier or harder.

They then predict the occurrence of the failure before it happens. They observe that injecting faults typically results in a failure occurring in about 110 seconds.

Although they do not show all the results they have, using $K = 20$ parameter for SHLLE, they achieve 72.3% Precision and 70.1% Recall on their test data (with a false positive rate of 11.6%). Prediction performance varies depending on the type of failure, as the study declares more than 60% recall on CPU and memory related failures, and around 70% recall for network failures. Coupled with the average of around 110 seconds in RTTF from injection of the faults, those results show that it is possible to implement automatic response mechanisms to the predicted failures. However in the case of the need of equipment replacement or manual intervention, it seems that these predictions would not be actionable, as they do not grant enough time for action to be taken before failure.

They conclude that fault prediction may, in turn, replace root cause analysis and root cause detection. We may add that predicting the root cause in addition to the present predictions may be a key in implementing early response, especially when human intervention is required, such as for when equipment replacement is inevitable.

Wang et al. [38] work on equipment fault prediction in an optical Wavelength Division Multiplexing (WDM) mesh network. They identify that while passive protection for optical networks exists, when a network fault occurs, data is still lost while waiting for the network protection and the recovery mechanisms of the network to kick in. Therefore, using prediction methods, we could enable proactive protection methods, not losing any data at the time of failure.

They propose a model based on five variables of the physical layer of the equipment, and then create three features based on each variable, namely, the minimum, average and maximum values of the day. Every day, the values of these features for the next day are predicted using Double Exponential Smoothing (DES). The resulting predicted vector is then fed into a Support Vector Machine (SVM) classifier [66] which predicts whether, the next day, the equipment will be unavailable for more than 40,000 seconds (11 hours) or not. The paper shows very promising results in these conditions, being able to boast a 91% prediction accuracy on a 24 hour advance time prediction, however the definition of failure as 11 hours of unusable time is quite a large one. The prediction metric given is accuracy, and the data is class balanced, so the accuracy should be somewhat representative of both classes (as visible in section II-B2 with the formula for accuracy, in cases where class imbalance is present, accuracy is not a good performance metric). Additionally, the information provided is relative to each piece of equipment and with the 24h time frame, the predictions are very easily actionable. However, while the low definition of the failure hypothesis and the 24 hour window probably make the problem much easier to solve, it limits the utility of this method for larger and more

Applications	Description	NFP examples
Equipment Health	Predict failure in network equipment in the future	[38], [39], [46], [54], [57]
RTTF	Predict Remaining Time to Failure on network equipment	[31]
Network Health	Predict Network Health (global state)	[29], [32], [36]
Link Failure	Predict whether link between two network equipments will go down	[34]
Alarm Prediction	Predict whether or not certain network states will lead to an alarm being raised (by an automated system or not)	[49], [59]

TABLE 3: Description of the different applications of NFP to networks

heterogeneous networks as further diagnostics need to be run to find and solve the root cause of the failure.

This paper also showcases the use of an alternative method of predicting future Key Performance Indicators (KPIs), and then detecting the fault in a second step. This method could be used to replicate some existing ML-based Fault Detection methods into a prediction environment with minimal adaptation work.

C. Zhang et al. [57] also work on an optical network and consider the same five physical variables as Wang et al. [38]. They propose the method of using an LSTM model to replace the two-step method of DES-SVM into one step.

They also consider 40,000 seconds unavailable time per day for an equipment being a failure. This paper boasts 93% prediction accuracy (though class balance data is not shared), however the results shown are incomplete.

These results would tend to show that there is potential for using more complex, RNN-type methods in NFP as propagating the time sequence context could provide better results than traditional single-vector inputs.

S. Zhang et al. [46] work on a set of about 10,000 switches of 3 different models in a Data Center (DC). They point out that in the case of DCs, where there can be several thousand switches online at the same time, the ability to predict switch hardware failure is paramount to ensure desirable QoS levels. Indeed, switch failures are responsible for 74% of downtime in modern DCs. Therefore ensuring an uneventful and preventive replacement of failing switches may prove to be a very effective way to ensure DC QoS.

In a given DC, there is usually only one or two different models of switches installed, and the network is very homogeneous. Thus they propose a model, *PreFix*, aimed to be trained specifically for each model used in DCs. *PreFix* uses batches of syslog messages in a switch, mapped onto generic templates. In this case, the duration covered by the batch is 30 minutes. They then extract four features from the batch: frequency, surge, seasonality and the sequence. The model then predicts whether or not the batch of syslog messages represents what the authors call a switch failure *omen*. A switch failure omen being a sequence of messages that appear less than a determined amount of time (in their case 24 hours) before a switch failure occurs in the equipment.

This model was trained on three switch models, on a database composed of data from around 10,000 switches over a period of 3 years, where switch failures were manually

classified by network operators. The results obtained are an average of 60% recall on the evaluation, with a False Positive Rate (FPR) of around 1.2 per 10,000 switches, per day. This FPR is deemed acceptable by the audited DC. In practice, this would mean that, for the DC, the cost of replacing misdiagnosed switches added to the cost of replacing the failing switches that were accurately identified, is inferior to the total cost of downtime that would be caused by the failing switches that were accurately identified by the model. An idea could be to run additional diagnostics on the predicted positive class equipment to see if this cost to reward ratio could be improved *a posteriori* of the first prediction.

In any case, this method seems to be effective in the case of very homogeneous networks, and with the wide time interval predicted, so it should provide enough time for logistics, a clean transition and replacement of the failing switches. The low FPR also contributes to the value of this model as it would help incorporate the model into the decision-making with minimal error cost. However, the dataset is heavily imbalanced and there seems to be around one failure per day in the dataset, which would mean there are more false positives than true positives predicted. As stated above, further verification could improve the cost to reward ratio of this model. This model also provides specific fault type information as the sequences of logs may be used to give a prognosis on the type or reason of the incoming failure. This model therefore seems adequate to base decisions upon. In the case of Heterogeneous Networks however, the method needs some adaptation to generalize it, as it is very specific to the equipment's design.

The authors plan to tune *PreFix* to more switch models, and try to tackle other types of switch failure. The trained models are available [here](#).

Boldt et al. [39] work on a set of 4G cellular base stations and the minimisation of the incident response time to restore the service provided by the base stations. They determine that given the increasing importance of networks, the capacity for cellular network service providers to anticipate service interruptions is paramount to ensuring maximal QoS for their customers.

A point to be noted is that although the authors describe their work as alarm prediction, they describe what they predict as *severe alarms* which they define as radio communications in one of the sectors of the cell is interrupted. This is for all intents and purposes a complete network failure for

that antenna in our case.

The authors seek to determine which ML models have the best performance with minimal optimization, and what is the best *time ahead* to use considering their data and chosen model. To this end, they setup two experiments : one to select a best-performing model based on the 1 hour ahead data based on prediction performance; and the second to evaluate the evolution of performance depending on *time ahead* selected (from 10min to 48h).

The data they use consists of classic monitoring data for 4G cells, with 231 features. They balance the no-failure/failure classes by a method of sampling, and normalize the features.

In the first experiment, DT, SVM, MLP, RF, and XGBoost models are evaluated using 10-fold cross-validation (a method where the model is evaluated 10 times along different splits of the training data in different training and validation sets) and performance is compared. They conclude that RF has performed best along the Area Under Curve (AUC) metric with $\sim 81\%$ Precision and $\sim 62\%$ Recall. MLP and XGBoost also perform similarly. RF is therefore chosen for the second experiment. The authors also evaluate the statistical relevance of these tests in comparing the different models.

In the second experiment, the RF model is evaluated with 10min, 20min, 30min, 1h, 2h, 3h, 6h, 12h, 24h, 48h *ahead time*. This experiment shows high performance in F1-score, between 10min and 3h ahead time, with a sharp drop between 3h and 6h. Perhaps this gap could be investigated in more detail in the future to determine an optimal *time ahead* to prediction performance ratio. The ELI5 module, a python module dedicated to explaining the predictions of classifiers, is also used to investigate the feature importance in decisions and identify several features heavily linked to the prediction.

They conclude that the necessary *time ahead* and precision metrics are linked to the decision making process that takes place after the prediction for maintenance of the model, so there is a need to investigate the decision boundary and the different performance behaviours of the models along these dimensions. Although they have one model that works well for one type of fault, they will experiment on more types, and train their models with more hyper-parameter tuning.

With regards to prediction performance discussed in the study, in the case of the selected ML model, that is to say RF, the levels shown for 1h and 3h *ahead time* for example seem satisfactory while still being actionable. Respectively, the precision and recall results given are $\sim 81\%$ and $\sim 62\%$ for 1h, and $\sim 77\%$ and $\sim 56\%$ for 3h. What remains to be determined, however, in order to determine the best *time ahead* to prediction performance for the model, is the cost of false positives in the case of a cellular network and what corrective actions (with respect to their cost) can be taken to reduce the impact of failures before they occur.

This study shows that there is a relevant need in investigating the different prediction performance levels at different *time ahead* values enabled by the data and the chosen mod-

els. Future works would be more complete if they were to integrate such a study.

Rafique et al. [69] publish a tutorial to apply ML-based techniques for networks in general. In this work, they present the idea of a two-step fault detection method. The first phase is a low computation cost model using the minimal number of necessary features to have an overview of the health status of the equipment. This model determines whether there seems to be anomalous behavior within the network or the equipment. If such behavior is detected, the first model triggers a second, more in-depth, investigation. The second phase being a much more precise diagnosis, which uses more network and computation resources. Such an idea seems reasonable to apply to fault prediction, as this could reduce the network and computation costs of distributed fault prediction models, while aiding to reduce error in predictions and decision making.

Velasco et al. indicate the previously cited method was tested for predictive maintenance in [54]. They indicate that ML techniques seem adequate to introduce dynamic adaptation of the network to conditions, however they do not share any results.

B. RTTF

Predicting the RTTF is a regression problem. As was described in section II-B1, uncertainty should reduce as the incoming failure approaches.

Pellegrini et al. [31] work on the effect of server applications failures on the network equipment (the server). They point out that the slow build-up of random errors in applications may account for a large part of anomalies on servers and addressing this in advance may be an opportunity to reduce equipment failure occurrence (and therefore network failure). Additionally, an effective technique for dealing with these errors is to force application or system rejuvenation, which consists in returning the application or system to a clean state. This is often achieved by restarting the system.

They therefore propose Framework for building Failure Prediction Models (F²PM) [31], that is a framework to generate application-caused failures on the network and use them to train a model to predict the exact Remaining Time To Failure (RTTF) for applications, due to memory leakage and non terminated thread build-ups. The paper uses data point aggregation and local derivation as their feature selection and pre-treatment. This is in order to allow a history of relevant information to be used without overloading the model with unnecessarily high resolution data.

The paper presents experiments using the data the framework generates on six types of models. They obtain very precise predictions of failure in the 0 to 10 minutes RTTF interval, using MSP or REP-Tree models. However, in the models, the **predicted RTTF** seems to plateau and fall back down to around 10 minutes when the **real RTTF** starts to exceed 10 minutes. This is an issue as it could lead to a high false positive rate (defined in section II-B2) when selecting a predicted RTTF threshold to raise an alert. Maybe this

'plateau and fall-back' issue can be explained by the design of the database where the failure is designed to always occur relatively quickly, and these issues could maybe be solved by widening the range of training examples and introducing segments of data without a failure occurring at the end. However, the results presented seem to show no such problem in the 0 to 8 minute range. Therefore, maybe an acceptable threshold could be found. And, in practice, a 8 minute reprieve or less (when the model seems to still be accurate) is largely sufficient to save the application's ongoing work and restart it from a clean state.

Perhaps this method can be replicated to more causes of network faults, but it is important to note that these application-related errors stem from ever-growing stacks of small errors that do not release the computation and memory resources. Whereas non application-related faults may also depend on the user load and network congestion, which vary greatly with time, and the state of other equipment. This might have a significant impact on RTTF prediction accuracy if this method is used for other types of networks. To conclude, this paper proposes a good method to predict RTTF with good quality of prediction for RTTFs below 10 minutes for application related faults.

C. NETWORK HEALTH

Some studies have focused on network health as a whole, instead of individual equipment. Depending on the size of the network, this approach can help reduce the class imbalance issue as there will be more frequent failures when considering more equipment. Modeling network health as a whole can also help plan maintenance teams' work.

Snow et al. [29] tackle the dependability of 2G cellular networks infrastructure. The Federal Communications Commission (FCC) changed its rules on the declaration of outages. Since then, and at least during the time that study was conducted, mobile operators must declare network failures that exceed 30 minutes of time, and result in at least 15,000 lost subscriber hours of service. This makes it of capital importance for wireless carriers to improve their network dependability, in order to defend economic priorities.

The authors assess that the optimal strategy with regards to improving network dependability has not been determined. They therefore study the evolution of network dependability under the new FCC rule, according to two dimensions of network dependability : reliability and maintainability. They also establish a model to attempt to predict the number of occurrences of network failures across time. They determine that a good strategy is to maintain reliability of the network (constant MTBF), while increasing maintainability (decrease Mean Time to Restore - MTR).

This seems logical, as the MTR network operations influences on the two points of the declarable outages rule of the FCC. Indeed, when the MTR is lower, there is both a higher chance that the failure will last less than 30 minutes, and that there will be less than 15,000 lost subscriber hours. This point is interesting, since, as seen in section III, other industry

actors are affected by similar criteria. NSIs and ISPs have to restore network operation within differing timeframes, and pay differing penalties when they cannot, depending on the criticality of the failures (which usually depends on the number of users dependant on the operation of the equipment).

Kumar et al. [32] work on cellular network data. They recognize that with the growing complexity and importance of cellular networks, mobile network faults become more common as their causes multiply, ranging from power outages, software or hardware malfunction, and constructor incompatibility, to misconfiguration of networks and cell density. The authors contend that the reactive maintenance of such networks is no longer appropriate given the possibility for most mobile network providers to introduce predictive maintenance (large capacity for data collection and emergence of predictive techniques).

Kumar et al. realize a study to test different models for predicting network faults. The experiments are conducted on a month's worth of mobile operator data. They aim to predict when the network will require maintenance to ensure the best QoS for their clients, using several different models to determine the best of them. The methods tested were : linear regression [70]; exponential regression [71]; linear SVM regression [72]; Gaussian (RBF) SVM regression [72]; different simple Neural Network [67] models (with around 20 neurons); and an auto-encoder [68]. A Continuous Time Markov Chain Analysis [73] was also used to analyze the data. The prediction made is whether a new fault will occur in the next 4 hours in the whole cellular network. However, the formulation of the problem makes it so that the network is considered unhealthy 95% of the time, while only one or a few of its nodes are failing. So there may be problems with always considering the network as a whole (when it reaches a certain size), as there would be difficulty in finding the faulty node using such a model. Perhaps a distributed view, while more costly in calculation resources, might be more capable of adaptation to the different conditions present in the network. Redefining the bounds of the network considered could also allow for better class balance in the data. Further analysis is necessary before it is possible to use the prediction for solving incidents.

We may note that after testing the prediction with several models, the auto-encoder model seems to perform better than the other models that were tested.

It seems that large cellular networks are failing most of the time, but the failure is localized to few or a singular equipment. Therefore there is an additional need in this case to be able to locate where the failure will occur in the future, in order to launch maintenance operations in advance. For reference, in the case of a large scale cellular infrastructure, in 30% of cases, the healthy network will experience the next failure in less than an hour, and in 87% of cases, it will experience failure in less than six hours, according to the data of the study.

D. LINK FAILURE

Most studies focus on equipment failures since when the equipment has a problem and is repaired or replaced, the network problem is also solved. However, there is also the possibility to reconfigure routes and avoid failing links between two network nodes, as some routing protocols rely on link states. Therefore there is a possibility of mitigating the consequences of link failures by rerouting in the case of predicted link failures.

Ibrar et al. [34] work on a Hybrid SDN (HSDN) architecture composed of legacy (non SDN) and SDN activated switches. They identify the need in HSDN networks for legacy switches link failure information to be communicated quicker to the Controller. Indeed when the legacy switches broadcast their link states, through the Open Shortest Path First (OSPF) routing protocol for example, this information must first reach SDN switches before the SDN switches themselves relay the information to the controller. This two-step process introduces delays and issues in the reconfiguration of the network to adapt to failures and causes performance issues.

Therefore the authors propose to predict link failures in the HSDN so that information reaches the controller quicker and the configurations can be changed in time for minimal interruption (while maintaining network coherence), as it is one of the functions of the SDN controller.

A simulation is built to accumulate data with 38 SDN compatible switches linked by 519 links into a single network. Varying percentages of switches are chosen randomly to function in legacy mode (not part of the SDN). Two models are then trained on the accrued data, a Logistic Regression and a SVM. In this study, the failure is considered predicted successfully if it is mitigated before it happens (there is no time in advance of the failure considered). In this case, mitigation means that the routes were recalculated incorporating the failure prediction information and the new routes were propagated before the occurrence of total failure.

The Logistic Regression model achieves the higher performances with 74% precision and 81% recall, and the SVM with 69% precision and 71% recall. These results seem to show that the models could very well be integrated into SDN controllers to increase prediction performance. However it is important to note that these results seem to vary depending on the proportion of SDN switches in the network and the rate of failure of the links. Further experimentation would be needed to determine the performance to be expected according to the number and proportion of equipment and rates of failure in order to determine the effectiveness of the method. Also the additional CPU and memory cost to the SDN controller should be taken into account as it might become an issue in larger networks.

This study shows an alternative to equipment failure prediction as redirecting network flow in prevision for failure could allow for the network to always be running, and the replacement of equipment could still happen after the fact without QoS and QoE impacts. However this method relies

on the implementation of SDN in the network and SDN is still far from widespread today. Perhaps in the future this could be a direction to solve the NFP problem.

E. ALARM PREDICTION

While similar to equipment health prediction, alarm prediction consists of predicting whether an alert will be raised rather than complete failure will be identified. An alert can be raised if certain monitoring values rise above a certain threshold, or if a customer complaint is registered for the equipment associated with the service.

Qian et al. [49] work on predicting incoming alarms in an Internet Protocol Television (IP TV) network. They spot that for IP TV networks, faults are primarily detected and reported by the end user, which introduces a delay in the troubleshooting and repair, and has a negative impact on customer satisfaction.

They therefore try to automate fault detection in IP TV, as that would reduce delays in detection and enable the service provider to restore quality of service before the client experiences significant trouble.

In order for the detection to be based on accurate data, they collect Quality of Service (QoS) data from the next-hop routers from the end users. They then select features among those KPIs by deleting those with low variance, those with low (linear or non-linear) correlation to the Quality of Experience (QoE) of end users. The Random Forest algorithm was used to estimate non-linear correlation of the features with the label.

The data is then fed into an AdaBoost model using its original error function and another AdaBoost model using F1 score as its error function in parallel [50].

They achieve 100% recall, with a 10% false positive rate on their test data with their improved AdaBoost model. However, the prediction that is made is a prediction of whether a customer is willing or not to make a complaint. This definition of failure is subject to variance depending on the customer, and whether a certain QoS level will lead him to take action.

The results show that it is possible to evaluate customer and network users QoE in the demanding setting of IP TV (high jitter and throughput constraints). Such a system may be adapted to predict network faults and their criticality for the client or the end user.

Zhuang et al. [59] work on a large scale network (presumed to be IP). They recognize that network monitoring data is quite dirty, as there is often empty data, due to congestion or equipment failures, and data is not normalized, as different manufacturers give different performance measures, etc. There is therefore a necessary and hefty process of cleaning the data or else NFP performance will not be satisfactory, and even cleaning the data does not yield better results as there is often not enough data left to adequately train models. As such, the authors try to build upon other works on generation of coherent data based on pre-processed real data.

After cleaning and pre-processing their data based on 50,000 alarms and 10,000,000 performance samples, they are left with 58 alarm sequences of data. They propose to generate more data with a GAN model, and train the model accordingly until it reaches Nash equilibrium. They then use the model to generate 6000 samples, 4,500 for training, and 1,500 for testing. Performance of an MLP trained using this data is then compared to performance of a model trained on just the clean data, another trained on augmented data using another method.

They boast a 99.9% accuracy and a 2.2% increase in performance compared to the other augmentation method. However, a common problem with data augmentation is the difficulty in determining the overlap between training and testing data, and it is therefore difficult to determine the part of increased performance that is due to an eventual overlap. A test of the augmented-data trained model using non-overlapping original data as a test set could be used to evaluate this effect.

Perhaps this method could be more widely tested in a benchmark on several ML methods of the previously mentioned studies in order to verify that these results generalize, but this method seems promising to ensure quality of data.

VI. PERSPECTIVES

Fault prediction has historically been more about predicting whether or not a network or a particular equipment would still be in a healthy state (functioning within performance expectations) in a future period of time. In some cases, it has also been about trying to determine the remaining time before the next failure on the equipment or the network, or link state failure, or predicting alarms raised. These predictions would then allow the actor in charge of maintenance to engage in proactive action and so reduce the MTTR.

However, in order to fully neutralize the impact of a component failure, a system that would, at a minimum, allow zero-loss and zero-delay handover is needed. This system could additionally trigger the intervention (if needed) enough time before the failure occurs so that the operations happen before users experience faulty service. In order to be applied in real networks, such a model would need to give an accurate failure prediction, locate the faulty equipment, and be able to either implement a response automatically, or advise the network administrators on policies to mitigate the predicted failure. In addition, an estimate of the criticality of the failure would also be desirable, in order to allow network administrators to budget and make decisions accordingly. Therefore, in order to enable zero-loss and zero-delay failure handover networks to emerge, NFP would need to become an extension of fault detection, root cause localization and fault mitigation in the future.

We have already seen a system that goes in that direction in Ibrar et al. [34], where the authors use an SDN controller to reroute traffic to avoid failing links. However if the objective is to completely neutralize the impact of failures, the system could be generalized to other types of networks (not only

HSDN and not only switches), and alarms could be raised to ensure maintenance for faulty nodes.

Several promising areas for future study have been identified in the following paragraphs.

Network supervision data is usually noisy, presents missing data points and is not normalized. Several studies have tried to establish a method for treating this issue [43], [49], [59]. However their methods have not been compared yet on a single dataset and perhaps better performing methods can be found.

In the case where we have several concurrent elements to form one prediction - for example failure prediction occurrence, localization of the failure, proactive mitigation and criticality - each predicted element needs to pertain to the specific fault instance. In the case of a large network for example, there could be several failures occurring in different places in the network in a short time frame. It would be highly undesirable for the different elements to concern unrelated failures. In order to ensure coherence of the global prediction, perhaps they could be realized in a sequential manner, using the output of all the previous predictions as input for the next one (for example using the predicted information of failure occurrence and fault localization as part of the input for predicting criticality of the failure). Another option could be to design the ML architecture using an intermediary common feature extraction system as input for all the different models. A third option could be to remove the need for certain predictions through other means. The experiment in Hood et al. [61] may indicate the possibility that we could use a limited number of prediction nodes, scattered in the network, in order to locate incoming failures by some sort of triangulation. This would allow us to solve the equipment localization problem without designing a separate model to predict the future failing equipment.

Another issue would be to determine the calculation costs for each method, that could be used for comparison and optimization purposes for choosing equipment and determining running costs. In the case where the NFP model is implemented in network equipment such as is the case in SDN, it would be useful to ensure that the additional processing overhead necessary to run the predictions does not cause further network failures.

One additional problem identified and partially treated in Boldt et al. [39] is that we have no benchmark to compare the different methods yet. This is mostly due to a lack of a commonly shared database for failure prediction and the wide disparity of types of networks that were studied (cellular, IP, DC, IPTV, optical, etc). There is therefore a need to introduce one or several public datasets and establish a performance benchmark according to several common prediction metrics.

Lastly, for a heterogeneous network NFP model to be implemented, a database large enough to cover most cases of faults and failures would be needed, and as such, should span multiple different networks. However, although there are databases for Fault Detection [74]–[76], and [this website](#) has been proposed for the sharing of databases for cognitive

network management in [77], to the best of our knowledge there are no databases available freely to researchers for Fault Prediction [23]. A freely available benchmark dataset of a heterogeneous network where all these methods could be used would be a first step towards bringing all these methods together.

VII. CONCLUSIONS

An overview of NFP, and a survey of the different studies of NFP using ML, along with a presentation of the ML methods used, were outlined in this work. The expectations of an actor of the field and future perspectives for research were also laid out.

It has been shown that in NFP, ML is a promising technology that remains in the early stages of development. Certain ML methods, such as transformers (attention-based models) and RNN, are receiving a lot of attention in other fields recently for their potential to deal with sequential input, and are therefore very promising for future NFP research.

Efforts have often been centered on homogeneous networks, dedicated to a single technology and manufacturer. However, companies now move towards a single network to transit all their multi-technology data, therefore making their networks ever more heterogeneous, and it is crucial to improve network dependability in order to reduce losses, and optimize performance. As a consequence, as networks are becoming ever more heterogeneous and critical, systems that are able to maintain efficiency while in such an environment are becoming increasingly desirable, and further work is required in this area.

Training effective models for heterogeneous networks would also need a set of common features, shared across the different types of equipment involved. To this end, testing the variation of prediction performance between using a narrower set of features, common to all types of equipment, and using a wider set of specialized features that are not common to all types of equipment seems worthy of further investigation.

Overall the different studies surveyed in the field of NFP treat a very wide selection of mostly-homogeneous network types, which makes the results difficult to compare to each other. Each of the applications considered tends to have a different set of characteristics and so the authors set up each problem differently.

As a consequence of specific networks having different crucial constraints, the prediction performance metrics overall are not uniform, which contributes to the difficulty in comparing studies.

The field of NFP lacks open data and a way to measure performance comparatively. NFP would benefit significantly from the establishment of one or several benchmark datasets, where the different methods studied could be tested, in order to set the stage for more consolidated research. Research studies remain to be explored to deal with the key issues of NFP for heterogeneous networks.

REFERENCES

- [1] "Calculating the cost of downtime in your business," Axcient, White Paper, 2018. [Online]. Available: <https://axcient.com/blog/calculating-the-cost-of-downtime-in-your-business/>
- [2] "Business value of cisco sd-wan solutions: Studying the results of deployed organizations," IDC, White Paper #US44952119, April 2019. [Online]. Available: https://transform.cisco.com/seller/idcreport_en?xs=99167
- [3] A. Larmo, P. Von Butovitsch, P. Campos Millos, and P. Berg, "Critical capabilities for private 5g networks," Ericsson, White Paper.
- [4] B. Wang, Z. Qi, R. Ma, H. Guan, and A. V. Vasilakos, "A survey on data center networking for cloud computing," *Computer Networks*, vol. 91, pp. 528–547, 2015.
- [5] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, and R. Sahoo, "Blue-gene failure analysis and prediction models," in *International Conference on Dependable Systems and Networks (DSN'06)*. IEEE, 2006, pp. 425–434.
- [6] F. C. Commission, "Docket no.04-35, fcc 04-188," August 2004.
- [7] A. Zolfaghari and F. J. Kaudel, "Framework for network survivability performance," *IEEE journal on selected areas in communications*, vol. 12, no. 1, pp. 46–51, 1994.
- [8] A. Snow, "A survivability metric for telecommunications: insights and shortcomings," in *Proc. Information Survivability Workshop*, 1998.
- [9] —, "The failure of a regulatory threshold and a carrier standard in recognizing significant communication loss." TPRC, 2003.
- [10] D. Chen, S. Garg, and K. S. Trivedi, "Network survivability performance evaluation: A quantitative approach with applications in wireless ad-hoc networks," in *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, 2002, pp. 61–68.
- [11] A. P. Snow, U. Varshney, and A. D. Malloy, "Reliability and survivability of wireless and mobile networks," *Computer*, vol. 33, no. 7, pp. 49–55, 2000.
- [12] U. Varshney, A. P. Snow, and A. D. Malloy, "Measuring the reliability and survivability of infrastructure-oriented wireless networks," in *Proceedings LCN 2001. 26th Annual IEEE Conference on Local Computer Networks*. IEEE, 2001, pp. 611–618.
- [13] A. Snow, U. Varshney, and A. Malloy, "A framework for simulation modeling of reliable available and survivable wireless networks," in *Proceedings of Workshop on Wireless Local Networks*, 2001.
- [14] L. N. Cassel, C. Partridge, and J. Westcott, "Network management architectures and protocols: Problems and approaches," *IEEE Journal on selected Areas in Communications*, vol. 7, no. 7, pp. 1104–1114, 1989.
- [15] F. R. Chung, "Reliable software and communication. i. an overview," *IEEE journal on selected areas in communications*, vol. 12, no. 1, pp. 23–32, 1994.
- [16] Y. Yemini, "A critical survey of network management protocol standards", telecommunications network management into the 21st century (s. aidarous and t. plevyak, eds), 1994."
- [17] C. S. Hood and C. Ji, "Automated proactive anomaly detection," in *International Symposium on Integrated Network Management*. Springer, 1997, pp. 688–699.
- [18] —, "Intelligent agents for proactive fault detection," *IEEE Internet Computing*, vol. 2, no. 2, pp. 65–72, 1998.
- [19] A. Danyluk, F. Provost, and B. Carr, "Telecommunications network diagnosis," 2008.
- [20] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, "Software rejuvenation: Analysis, module and applications," in *Twenty-fifth international symposium on fault-tolerant computing. Digest of papers.*. IEEE, 1995, pp. 381–390.
- [21] M. Kordestani, M. Saif, M. E. Orchard, R. Razavi-Far, and K. Khorasani, "Failure prognosis and applications—a survey of recent literature," *IEEE transactions on reliability*, 2019.
- [22] P. C. Fonseca and E. S. Mota, "A survey on fault management in software-defined networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2284–2321, 2017.
- [23] L. Pan, J. Zhang, P. P. Lee, M. Kalandar, J. Ye, and P. Wang, "Proactive microwave link anomaly detection in cellular data networks," *Computer Networks*, vol. 167, p. 106969, 2020.
- [24] 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, pp. 1–99, 2018.

- [25] S. Cherrared, S. Imadali, E. Fabre, G. Gössler, and I. G. B. Yahia, "A survey of fault management in network virtualization environments: Challenges and solutions," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1537–1551, 2019.
- [26] Y. Yu, X. Li, X. Leng, L. Song, K. Bu, Y. Chen, J. Yang, L. Zhang, K. Cheng, and X. Xiao, "Fault management in software-defined networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 349–392, 2019.
- [27] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 393–430, 2019.
- [28] D. Mulvey, C. H. Foh, M. A. Imran, and R. Tafazolli, "Cell fault management using machine learning techniques," *IEEE Access*, vol. 7, pp. 124 514–124 539, 2019.
- [29] A. Snow, P. Rastogi, and G. Weckman, "Assessing dependability of wireless networks using neural networks," in *MILCOM 2005 - 2005 IEEE Military Communications Conference*, Oct 2005, pp. 2809–2815 Vol. 5.
- [30] M. L. Alahdab, "Study of solutions for automatic reconfiguration of future transport networks based on flexible optical layer: Etude de solutions pour une reconfiguration automatique des futurs réseaux de transport basés sur une couche optique flexible," Ph.D. dissertation, Institut polytechnique de Paris, 2019.
- [31] A. Pellegrini, P. D. Sanzo, and D. R. Avresky, "A machine learning-based framework for building application failure prediction models," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, May 2015, pp. 1072–1081.
- [32] Y. Kumar, H. Farooq, and A. Imran, "Fault prediction and reliability analysis in a real cellular network," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2017, pp. 1090–1095.
- [33] E. Alpaydin, "Introduction to machine learning, 3rd editio. ed," 2014.
- [34] M. Ibrar, L. Wang, G.-M. Muntean, A. Akbar, N. Shah, and K. R. Malik, "Prepass-flow: A machine learning based technique to minimize acl policy violation due to links failure in hybrid sdn," *Computer Networks*, vol. 184, p. 107706, 2021.
- [35] Y. Y. Liu, M. Yang, M. Ramsay, X. S. Li, and J. W. Coid, "A comparison of logistic regression, classification and regression tree, and neural networks models in predicting violent re-offending," *Journal of Quantitative Criminology*, vol. 27, no. 4, pp. 547–573, 2011.
- [36] C. Cortes, L. D. Jackel, W.-P. Chiang et al., "Limits on learning machine accuracy imposed by data quality," in *KDD*, vol. 95, 1995, pp. 57–62.
- [37] T. Kohonen, "Learning vector quantization," in *Self-organizing maps*. Springer, 1995, pp. 175–189.
- [38] Z. Wang, M. Zhang, D. Wang, C. Song, M. Liu, J. Li, L. Lou, and Z. Liu, "Failure prediction using machine learning and time series in optical network," *Opt. Express*, vol. 25, no. 16, pp. 18 553–18 565, Aug 2017. [Online]. Available: <http://www.opticsexpress.org/abstract.cfm?URI=oe-25-16-18553>
- [39] M. Boldt, S. Ickin, A. Borg, V. Kulyk, and J. Gustafsson, "Alarm prediction in cellular base stations using data-driven methods," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1925–1933, 2021.
- [40] E. Osuna, R. Freund, and F. Girosit, "Training support vector machines: an application to face detection," in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE, 1997, pp. 130–136.
- [41] S. Suthaharan, "Machine learning models and algorithms for big data classification," *Integr. Ser. Inf. Syst.*, vol. 36, pp. 1–12, 2016.
- [42] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [43] Xu Lu, Huiqiang Wang, Renjie Zhou, and Baoyu Ge, "Using hessian locally linear embedding for autonomic failure prediction," in *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, Dec 2009, pp. 772–776.
- [44] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.
- [45] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.
- [46] S. Zhang, Y. Liu, W. Meng, Z. Luo, J. Bu, S. Yang, P. Liang, D. Pei, J. Xu, Y. Zhang, Y. Chen, H. Dong, X. Qu, and L. Song, "Prefix: Switch failure prediction in datacenter networks," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 2, no. 1, Apr. 2018. [Online]. Available: <https://doi.org/10.1145/3179405>
- [47] M. Pal, "Random forest classifier for remote sensing classification," *International journal of remote sensing*, vol. 26, no. 1, pp. 217–222, 2005.
- [48] R. L. Chang and T. Pavlidis, "Fuzzy decision tree algorithms," *IEEE Transactions on systems, Man, and cybernetics*, vol. 7, no. 1, pp. 28–35, 1977.
- [49] Y. Qian, X. Wei, R. Huang, H. Meng, and Q. Liu, "Fault prediction in iptv using improved adaboost algorithm," in *2016 8th International Conference on Wireless Communications & Signal Processing (WCSP)*. IEEE, 2016, pp. 1–5.
- [50] Y. Freund, R. E. Schapire et al., "Experiments with a new boosting algorithm," in *icml*, vol. 96. Citeseer, 1996, pp. 148–156.
- [51] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho et al., "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.
- [52] H. A. Chipman, E. I. George, and R. E. McCulloch, "Extracting representative tree models from a forest," in *IPT GROUP, IT DIVISION, CERN*. Citeseer, 1998.
- [53] Y. Wang and I. H. Witten, "Induction of model trees for predicting continuous classes," 1996.
- [54] L. Velasco and D. Rafique, "Fault management based on machine learning," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2019, pp. 1–3.
- [55] T. Mitchell, "Machine learning," 1997.
- [56] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.
- [57] C. Zhang, M. Wang, M. Zhang, D. Wang, C. Song, L. Guan, and Z. Liu, "Adaptive failure prediction using long short-term memory in optical network," in *2019 24th OptoElectronics and Communications Conference (OECC) and 2019 International Conference on Photonics in Switching and Computing (PSC)*. IEEE, 2019, pp. 1–3.
- [58] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [59] H. Zhuang, Y. Zhao, X. Yu, Y. Li, Y. Wang, and J. Zhang, "Machine-learning-based alarm prediction with gans-based self-optimizing data augmentation in large-scale optical transport networks," in *2020 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2020, pp. 294–298.
- [60] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.
- [61] C. S. Hood and C. Ji, "Proactive network-fault detection [telecommunications]," *IEEE Transactions on Reliability*, vol. 46, no. 3, pp. 333–341, Sep. 1997.
- [62] M. Thottan and C. Ji, "Anomaly detection in ip networks," *IEEE Transactions on signal processing*, vol. 51, no. 8, pp. 2191–2204, 2003.
- [63] —, "Proactive anomaly detection using distributed intelligent agents," *Ieee network*, vol. 12, no. 5, pp. 21–27, 1998.
- [64] A. W. Williams, S. M. Pertet, and P. Narasimhan, "Tiresias: Black-box failure prediction in distributed systems," in *2007 IEEE international parallel and distributed processing symposium*. IEEE, 2007, pp. 1–8.
- [65] L. D. Solano-Quinde and B. M. Bode, "Module prototype for online failure prediction for the ibm blue gene/l," in *2008 IEEE International Conference on Electro/Information Technology*. IEEE, 2008, pp. 470–474.
- [66] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.
- [67] B. Kröse, B. Krose, P. van der Smagt, and P. Smagt, "An introduction to neural networks," 1993.
- [68] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming autoencoders," in *International conference on artificial neural networks*. Springer, 2011, pp. 44–51.
- [69] D. Rafique and L. Velasco, "Machine learning for network automation: Overview, architecture, and applications invited tutorial," *J. Opt. Commun. Netw.*, vol. 10, no. 10, pp. D126–D143, Oct 2018. [Online]. Available: <http://jocn.osa.org/abstract.cfm?URI=jocn-10-10-D126>
- [70] S. Weisberg, *Applied linear regression*. John Wiley & Sons, 2005, vol. 528.
- [71] J. Beirlant, G. Dierckx, Y. Goegebeur, and G. Matthys, "Tail index estimation and an exponential regression model," *Extremes*, vol. 2, no. 2, pp. 177–200, 1999.
- [72] S. R. Gunn et al., "Support vector machines for classification and regression," *ISIS technical report*, vol. 14, no. 1, pp. 5–16, 1998.

- [73] W. J. Anderson, *Continuous-time Markov chains: An applications-oriented approach*. Springer Science & Business Media, 2012.
- [74] J. A. J. M. N. J. C. D. Lopez, "Computer network events," 2019. [Online]. Available: <https://dx.doi.org/10.21227/dbgv-zz92>
- [75] "Anomaly detection in cellular networks." [Online]. Available: <https://www.kaggle.com/c/anomaly-detection-in-cellular-networks>
- [76] "Network anomaly telemetry datasets." [Online]. Available: <https://github.com/cisco-ie/telemetry>
- [77] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, 2018.

AUTHORS

KILLIAN MURPHY obtained an engineering degree in telecommunications at Institut Polytechnique de Paris/Telecom SudParis, France in 2020.

He worked as a Research Intern at Lulea Tekniska Universitet EISLAB Machine Learning in 2019, and as an Engineer Intern at Bouygues Telecom in 2019-2020. He works at SPIE ICS, in Malakoff France, since 2020 as a Research Engineer while preparing his PhD on network fault prediction.

ANTOINE LAVIGNOTTE received the B.Sc. degree in networking from Xamk - South Eastern Finland University of Applied Sciences, Mikkeli, Finland, in 2005; the M.Sc. degree (Networking) from Telecom Saint-Etienne, France, in 2008; and the PhD degree (Computer Science) from University Jean-Monnet, Saint-Etienne, France in 2014.

He is Full Professor with the Networks and Telecommunication & Services department of IMT/Télécom SudParis, Institut Polytechnique de Paris since 2021. He was an associate professor at Télécom SudParis between 2014 and 2021. He is also a teacher at Ecole Polytechnique (X). Before joining Telecom SudParis, he was an engineer at SPIE ICS, a European (Network and System Integrator).

He is the head of the chair "Future networks for the services of tomorrow" created with five French leading companies in the networking field (Bouygues Telecom, CNS communications, Nokia, SNCF, SPIE).

His research activities cover optical networks, security in industry 4.0 (smart factory) and machine learning techniques in networking.

CATHERINE LEPEPERS received the PHD degree (Chaotic Dynamics in Lasers) from University of Lille, France, in 1993. Since 2008, she is Full Professor with the department of Electrical Engineering and Dean for Faculty Affairs since 2019, at Institut Polytechnique de Paris/Telecom SudParis. She was the Head of the Optics and Photonics Group in her department from 2008 to 2021. Before joining IP Paris/Telecom SudParis, she was an Associate Professor at University of Lille where

she conducted research on Dynamics in Lasers and Photonics. From 2000 to 2008, she performed research on OCDMA in optical communications at IP Paris/Telecom Paris as associate researcher.

Her present research interests in SAMOVAR Lab. include machine learning for optical networks and visible light communications. She managed projects devoted to home networking, ROADM node evaluation and multilayer network dimensioning. She supervised a MOOC on Optical Access Networks.

She is a full member of the Light Communications Alliance (LCA). She is Deputy Head of the research committee in DIGICOSME Labex. She is a member of the Academic Council from IP Paris.

...