

Towards Internet-Scale Convolutional Root-Cause Analysis with DiagNet

Loïck Bonniot, Christoph Neumann, François Taïani

▶ **To cite this version:**

Loïck Bonniot, Christoph Neumann, François Taïani. Towards Internet-Scale Convolutional Root-Cause Analysis with DiagNet. IPDPS 2021 - The 35th IEEE International Parallel and Distributed Processing Symposium, May 2021, Portland / Virtual, United States. 10.1109/IPDPS49936.2021.00084 . hal-02534888v2

HAL Id: hal-02534888
https://hal.science/hal-02534888v2

Submitted on 18 May 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Copyright - All rights reserved

Towards Internet-Scale Convolutional Root-Cause Analysis with DIAGNET

Loïck Bonniot[1,2], Christoph Neumann[1], François Taïani[2]
[1]InterDigital, [2]Univ Rennes, Inria, CNRS, IRISA
{firstname.surname}@interdigital.com, {firstname.surname}@irisa.fr

***Abstract*—Diagnosing problems in Internet-scale services remains particularly difficult and costly for both content providers and ISPs. Because the Internet is decentralized, the cause of such problems might lie anywhere between a user's device and the datacenters hosting the service. Further, the set of possible problems and causes is not known in advance, making it impossible in practice to train a classifier with all combinations of problems, causes and locations.**

In this paper, we explore how machine learning techniques can be used for Internet-scale root cause analysis based on measurements taken from end-user devices. Using convolutional neural networks, we show how to build generic models that (i) are agnostic to the underlying network topology, (ii) do not require to define the full set of possible causes during training, and (iii) can be quickly adapted to diagnose new services. We evaluate our proposal, DIAGNET, on a geodistributed multi-cloud deployment of online services, using a combination of fault injection and emulated clients running within automated browsers. Our experiments demonstrate the promising capabilities of our technique, delivering a recall of 73.9%, including on causes that were unknown at training time.

I. INTRODUCTION

Both on-line content providers and Internet service providers (ISPs) allocate significant resources to troubleshoot end-user problems, as even a small drop in their customers' Quality of Experience (QoE) can strongly impact their revenues and brand image [1], [2]. Unfortunately, pinpointing the cause of a problem can be a tedious process, as the underlying fault might lie anywhere between a customer's home and the final data center, and many of the locations involved are neither controlled by the ISPs, nor by the content provider. Worse, as services grow more complex and interdependent, deciding whether a perturbation somewhere in the Internet has caused a customer's trouble is increasingly challenging, leaving support teams to struggle to diagnose the root cause of many incidents [3].

In this context, automating—even partially—the root-cause analysis of end-user problems appears as particularly beneficial to users, ISPs, and content providers, and numerous prior works have proposed to exploit end-user devices and equipment to this end [4], [5]. These works adopt two main strategies. The first is to execute a set of predefined tests, designed by experts, to identify outliers and propose a diagnostic [4]–[6]. These tests are capable of detecting known configuration issues (DNS failures, aggressive firewall, low quality uplink, ...) with high accuracy, but remain bound to a limited set of problems (e.g. related to DNS, TCP or DSL [7]) and cannot diagnose more distant Internet failures. The second strategy uses a shared service status database such as https://downdetector.com to distinguish between local and distant faults. Unfortunately, such services offer only a coarse-grained picture based on manual user reports and cannot provide a precise root-cause diagnosis. More generally, existing automated diagnostic solutions cover only a small and specific part of the possible root causes of online incidents, and fail at offering generic Internet-scale root-cause inference.

To overcome these limitations, we propose DIAGNET, a distributed platform for the root cause analysis of Internet-based services. DIAGNET exploits novel convolutional techniques derived from machine learning and image processing [8]–[10], and apply these techniques to data collected from user devices and opportunistically deployed probing servers called *landmarks*. By relying on a neural network for root cause inference, DIAGNET does not require any knowledge of the low-level network fabric that connects its target services (such as routers, switches, or peering policies), and can ingest new types of network measurements without the need for retraining. Doing so, DIAGNET can pinpoint root causes in locations of the Internet it never encountered before, and can easily be adapted to different types of online services with very little retraining. The principles behind DIAGNET are further not limited to end-user problems, and generalize easily beyond Browser-based services, to distributed B2B APIs.

We carefully evaluate DIAGNET by injecting faults on a deployment of realistic geodistributed services and clients. Our experiment involves 4 cloud providers in 10 world regions, interdependent services, and emulated users running in automated browsers. We compare DIAGNET against state-of-the-art inference methods and show that it consistently overperforms its competitors in a dynamic context, which is typical of today's Internet services, while delivering close to ideal performances in a static setting. In our evaluation, DIAGNET yields an overall Recall@1 of up to 73.9%.

In the following, we first discuss the problem of Internet-scale Root Cause Analysis in more detail (Section II), before presenting DIAGNET (Section III). We then evaluate our proposal on a geodistributed deployment alongside with two baseline competitors (Section IV), we present related work in Section V, and conclude (Section VI).

Copyright © 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works by sending a request to pubs-permissions@ieee.org. Presented in the 35th IEEE International Parallel and Distributed Processing Symposium (IPDPS '21). For the final published version, refer to DOI 10.1109/IPDPS49936.2021.00084

II. Problem statement and goals

Measurements taken from end-user devices are inexpensive, but the information they provide is limited. To infer some accurate diagnosis from this information, we argue that a root cause analysis system working at Internet-scale should meet a number of key challenges.

A. Network and service agnosticism

Internet-services rely on a wide variety of systems, sub-services, and networks to function properly. This includes data centers, cloud-providers, content delivery networks (CDN), along with diverse operators' networks. The underlying network topologies and distributed architectures of these systems are complex, continuously evolving and often unknown. We argue that an Internet-scale root cause analysis method should not assume any prior knowledge regarding the architecture of its targeted services (e.g. in terms of the cloud regions they are deployed in, or the CDNs used), and regarding the network topology on which they execute (e.g. in terms of peering-points), a property that we call *network and service agnosticism*. This largely departs from common root cause analysis relying on network tomography [11]–[13], bespoke methods for data centers [14], [15] and Software-Defined-Networking [16], [17].

Root cause analysis requires however some location information to pinpoint the area (e.g. cloud region, point of presence, autonomous system[1]) in which a root cause is likely to be located. Our choice in DiagNet is to rely on *landmark servers* to provide this location information while eschewing a precise knowledge of the underlying network. Landmark servers are easy to deploy, cheap to run and maintain, and can provide a good overview of the network health provided they are present in multiple and diverse vantage points. The intuition is that if there is a sufficiently wide deployment of landmarks, some of these landmarks will be located in the topological vicinity of targeted services, or in the path towards them, thus offering indications on the location and family of the incident impacting a user.

B. Anomaly disentanglement

By providing measurements from all over the Internet, landmark servers are bound to record a constant stream of anomalies (a drop in bandwidth here, a high latency there). Most of these anomalies will however be unrelated to a particular end-user's problem with a particular service: long delays between Paris and Brussels are unlikely to explain why a Korean student cannot access her university's web-site. An internet-scale root-cause analysis service should therefore be capable of separating spurious outliers from actual causes when analyzing an incident, a property we have dubbed *anomaly disentanglement*.

Disentangling real causes from coincidental effects is however far from trivial without any detailed knowledge of a service's internal organization (the fact that the Korean website does not use any resource in Europe for instance), or of the network topologies it executes on (packets circulating in Korea do not normally go through any Paris-Brussels link). Our intuition in DiagNet is that a *learned inference model* should be able to autonomously discover these hidden relationships, by inferring the internal dependencies within a service and within the network from past observations.

C. Location agnosticism

Historically, diagnostic tools operating on Wide-Area Networks have exploited the precise location of every user device (also called *client* in the following) accessing the service to pinpoint failures accurately [18], both at a geographical (from neighborhood to country) and topological level (from subnet to ISP). Obtaining such detailed data from every end-device can be difficult and even undesirable as users might refuse to share their location out of privacy concerns.

In DiagNet, we propose to circumvent the need for precise location information altogether, and argue instead that a root cause analysis model should be *location-agnostic*: the same single model should apply to every end-user device that participates in the root-cause analysis service. However, we believe it is acceptable to have distinct models for distinct services, since they are definitely less numerous than possible client profiles while being possibly very diverse. We show in Section III how this level of expressiveness can be achieved by revisiting multi-layer perceptrons and convolutions in the context of network diagnosis.

D. Root cause extensibility

Because we make very few assumptions on the underlying network, and do not use detailed location information of the clients, the granularity of the measurements we obtain is closely related to the deployment of our landmark servers. Many factors can however alter the *availability* of these landmarks (e.g. failures, maintenance or saturated capacity). Conversely, if the system contains a very high number of landmarks, individual clients cannot be expected to probe every landmark in order to keep overheads low. As an extreme example, it would require at least 99,000 landmark servers to cover every autonomous system[2], a number clearly too high for comprehensive probing.

To address these issues of scale and dynamicity, we require our generic root cause analysis system to be *extensible*: trained models should be able to consume measurements from a varying number of landmarks, depending on their availability at a given time. This property allows for easy maintenance of the landmarks fleet. Since the location of a plausible root cause is directly inferred from landmarks, the better landmarks cover the Internet the more precise the resulting inference can be expected to be. A *root cause extensible* model should still however provide accurate results even when only a subset of landmarks is available. This implies a number of choices in the design of our proposal to avoid frequent model retraining.

[1]On the Internet, an autonomous system (AS) is a management entity in charge of a part of the Internet, usually controlled by some operator.

[2]Data from Regional Internet Registries as of January 1, 2021

III. DIAGNET

A. Overview

The general organization of DIAGNET is shown in Figure 1. DIAGNET combines software probes deployed within the browsers of end-users (called *clients*, more on this below) with dedicated landmark servers. Clients and landmarks continuously produce network measurements, which are combined with ground-truth information (obtained using either fault injection, as in our evaluation, or by some external post-mortem analysis of past incidents) to train a root-cause inference model using machine learning techniques. This inference model is then provided to clients as an online *analysis service*, and used to diagnose failures of online services consumed by clients.

More concretely, each client produces measurements by actively probing ℓ landmark servers, implemented as a set of stateless public HTTP services that can be provided by different ISPs or third parties, similarly to the Speedtest global network (https://www.speedtest.net/speedtest-servers). In our implementation, we leverage modern web browser capabilities to fetch TCP statistics, latency and throughput information from the landmark servers (for a total of k metrics per landmark server and per client), to which we add some *local system features* measured on the clients themselves (e.g. client CPU and memory load). Within a browser this can either be implemented as a JavaScript that is fetched when accessing the online service or as a browser extension. We opted to collect only a limited set of metrics in our prototype, but additional metrics could easily be added. More detail on the implementation of our infrastructure can be found in [19].

The measures collected by a client c_i form a vector of m measures $\mathbf{x}_i = (x_{i,j})_{1 \leq j \leq m} \in \mathbb{R}^m$ ($k \times \ell$ measures from the ℓ landmark servers, plus the local client metrics). They provide the *features* that are fed into the root cause analysis service. (In the following we use the terms *measure* and *feature* interchangeably.) We assume clients also collect the Quality of Experience (QoE) perceived by their users through a binary indicator, that records whether a user is experiencing a problem or not for a given service. This QoE information might be manually provided by users, or automatically estimated. It can be as simple as a page load time or can rely on a method that calculates it [20]. From that data, and assuming that a client c_i is encountering some QoE degradation, DIAGNET processes the vector $\mathbf{x}_i$ of measures collected by c_i, and outputs a list of probable root causes using its learned inference model, ranked according to their likely impact on the incident being diagnosed.

In DIAGNET, a root cause diagnosis combines a (possibly coarse-grained) *location* with a *fault family*, e.g. "abnormal jitter within the AWS US east coast region" or "high latency within local network". In practice, we use individual landmarks to represent remote locations (e.g. a landmark deployed in AWS's east coast region will represent this region), and equate fault families with the network and local metrics we collect (e.g. 'upload bandwidth', or 'CPU load'), as these metrics capture relatively well the behavior of the infrastructure

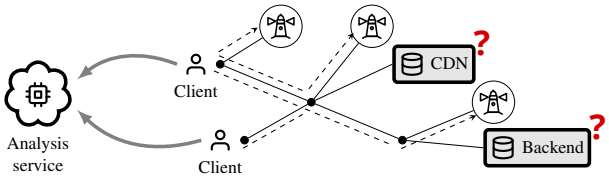


Fig. 1. Toy example of a topology for an online-service relying on a CDN and a backend. Clients can evaluate links (solid lines) by actively probing landmark servers (dashed lines). Probes are sent to a root cause analysis service, which builds and shares the root cause inference model.

on which services execute. As a result, the space of possible root causes of an incident is precisely that of the features we collect: $\ell \times k$ features represent remote root-causes, made up of a remote location (landmark) with a network metric, while the local features capture local root causes.

B. The DIAGNET inference model

At the core of DIAGNET's analysis service lies its inference model, which we implement using a combination of neural network techniques (notably non-overlapping convolutions and pooling layers), and an attention mechanism [10], [21], [22], a technique derived from image-based classifiers that is able to relate a prediction to its inputs.

DIAGNET's inference model exploits the fact that the root causes we seek to predict correspond to the set of features it consumes. This connection between inputs and outputs allows us to decompose the prediction process in two simpler steps: In a first step, we only predict the *family* of encountered fault(s) without any information on their location (what we call a *coarse prediction* in the following). The number of fault families c is fixed (corresponding to network and local metrics), and the resulting prediction is a small-size vector $\mathbf{y} \in [0,1]^c$ of probabilities. We use the following fault families in our prototype: *nominal* (non-faulty); *uplink latency* for gateway malfunction; *remote link latency*, *link jitter*, *link loss* and *link download/upload bandwidth* for end-to-end issues not related to the local uplink; and *local load* for client device overload. This set of fault families covers all problems generally investigated in the networking literature that can be linked to metrics obtained by users. As c (the dimension of coarse predictions) is low ($c \ll m$), we can build accurate inference models for fault families with a reasonably low number of ground-truth samples. Intuitively, coarse predictions help us disentangle features showing hidden relationships. For instance, a high TCP latency often leads to a degraded throughput [23]: in that case, the coarse prediction should return the latency as the root cause of the problem ("remote link latency"), rather the bandwidth ("link download bandwidth").

In a second step, DIAGNET uses the vector $\mathbf{y} \in [0,1]^c$ of predicted coarse predictions to return to the input feature space of dimension m and locate the fault, in effect equating the final predicted classes with the space of input features. We use an *attention mechanism* for this step, a machine learning technique that is able to compute the *weight* of each input

feature in the coarse model's prediction, usually without the need for any additional training.

The global architecture of DIAGNET's inference engine is depicted in Fig. 2. The coarse prediction phase involves the steps of ① separating landmark features from local features, ② processing the landmark features with a specific type of *convolutional neural network* (detailed in Section III-C), ③ ④ processing all features with a fully-connected network and obtaining the final coarse prediction (detailed in Section III-D). The second phase involves the step of ⑤ returning back to the input features via attention mechanisms (Section III-E).

C. Non-overlapping convolutions with pooling

In image analysis, convolutional neural networks [8] have been used with considerable success to classify images. Their convolutional layers extract patterns over multiple pixels by applying small filters over each pixel and its neighbors. We borrowed this idea of *pattern extraction* to extract *common patterns* between different landmarks, with some differences.

First, in contrast with image pixels, we want to combine measures of different nature (linked to "fault families", such as latency and bandwidth). For a landmark λ, and a client c_i, we note $\mathbf{x}_i[\lambda] \in \mathbb{R}^k$ the vector of measures (e.g. latency, throughput) recorded by c_i w.r.t the landmark λ. For example, $x_{i,1}[\lambda]$ might store the RTT (Round Trip Time) from c_i to λ, and $x_{i,2}[\lambda]$ the throughput. In this first phase, we seek to extract recurring patterns from each landmark in isolation. To this aim, we apply a set of f *non-overlapping* convolutions to each client/landmark measure vector $\mathbf{x}_i[\lambda]$. These convolutions are commonly parameterized by a kernel $\mathbf{K} \in \mathbb{R}^{f \times k}$ and a bias $\mathbf{b} \in \mathbb{R}^f$. Formally:

$$\forall \lambda \in \{1, \ldots, \ell\}, \mathbf{F}[\lambda] = \mathbf{K}.\mathbf{x}_i[\lambda] + \mathbf{b}.$$

At this stage, the $\ell \times k$ landmark features have been projected into a new feature space of dimension f (the number of filters). Since the $\mathbf{K}$ and $\mathbf{b}$ parameters are shared for every landmark, we believe that *common patterns between landmarks* are learned: our model shall hopefully extract useful information about the underlying network architecture. Nevertheless, it is still required to return a vector which size is independent of the number of available landmarks. We thus leverage global pooling layers [9], a popular mechanism to support variable-size inputs and ensure good generalization in image analysis. In our case, we apply a global function Ω on every landmark's convolution feature element-wise:

$$\mathbf{F} = \bigoplus_{\lambda=1}^{\ell} \mathbf{K}.\mathbf{x}_i[\lambda] + \mathbf{b}, \mathbf{F} \in \mathbb{R}^f$$

We define this process as a new kind of neural network layer and call it "LandPooling" by reference to landmarks. An illustration of this landmark-flattening process is depicted in Fig. 3. We note that any commutative function that can be applied with a generic number of arguments can be chosen for Ω. We explored several combinations of hyperparameters and kept the best configuration listed in Table I.

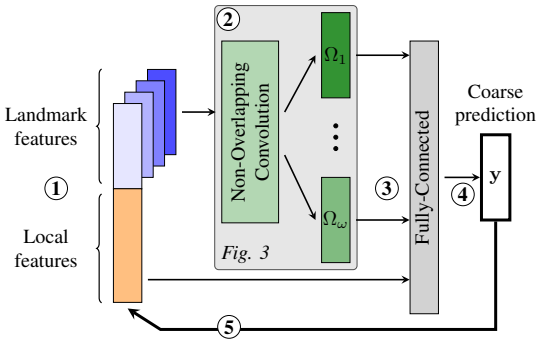


Fig. 2. Architecture of DIAGNET. ① Landmark features are first separated from local features and ② fed in the LandPooling layer with multiple parallel global pooling operations. ③ A hidden fully-connected layer is applied after concatenating the LandPooling output with local features. ④ The coarse fault prediction is obtained by applying a series of non-linearities. ⑤ Finally, an attention model is applied on the coarse prediction to return to the feature space and propose a fine-grained fault localization.

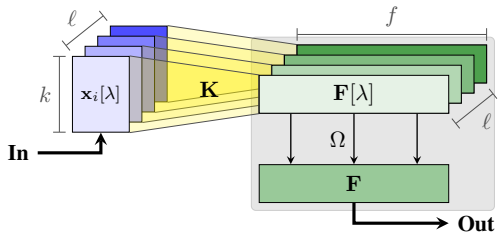


Fig. 3. Overview of the non-overlapping convolutional layer with pooling (LandPooling). For each landmark λ, the k features of that landmark $\mathbf{x}_i[\lambda]$ are transformed to a new feature space $\mathbf{F}[\lambda]$ of size f through a shared kernel $\mathbf{K}$. To return a fixed-size output of size f, the results for the ℓ landmarks are combined through a global Ω function, such as maximum, average or others.

D. Tailoring to specific services

Similarly to classical classification tasks relying on convolutional architectures, we add a multi-layer perceptron after the LandPooling mechanism presented in the previous subsection. The main purpose of these additional layers is to increase the expressivity of DIAGNET, by permitting a non-linear combination of the results of the global pooling and the local features resulting in coarse fault predictions. As illustrated in Fig. 2, this perceptron (also called "Fully-Connected layers") accepts

TABLE I
NOTATIONS AND HYPERPARAMETERS

ℓ	Total number of landmarks (10)
f	Number of convolutional filters (24)
k	Number of features per landmark (5)
m	Number of features per sample ($\ell \times k +$ local features $= 55$)
c	Number of coarse fault families (7)
Ω	Global pooling operations (min, max, avg, variance, p10, …, p90)
$\mathbf{x}_i$	$= (x_{i,j})_{1 \leq j \leq m}$, input sample of client c_i
$\mathbf{y}_i$	$= (y_{i,j})_{1 \leq j \leq c}$, coarse predictions for client c_i
$\hat{\gamma}_i$	$= (\hat{\gamma}_{i,j})_{1 \leq j \leq m}$, predicted features usefulness for c_i
2 hidden layers $(512 \times 1), (128 \times 1)$ with ReLUs; optimizer: SGD with Nesterov (learning rate = 0.05, decay = 0.001)	
Auxiliary model: Random Forest (Gini impurity criterion, 50 estimators, maximum depth = 10)	

multiple inputs: the global pooling functions $\Omega_1, \ldots, \Omega_\omega$ along with the "local features" that are independent of available landmarks. This additional expressivity is necessary to model the dependencies between services and input features. By default, DIAGNET uses one single *general* set of final fully-connected layers to diagnose multiple services.

However, such *general* model could demonstrate irregular performance if the set of monitored services is very diverse: not all Internet services have the same network requirements and dependencies. For example, while the latency is critical in multiplayer games, it might intuitively not be the case for video streaming systems where the available bandwidth is usually the bottleneck. It is thus possible to build one *specialized* DIAGNET model per service to improve its accuracy, by learning a dedicated set of fully-connected layers for that service. We detail and evaluate this property in Section IV-F.

E. Fine-grained inference via attention mechanisms

To offer a fully extensible model, we need a mechanism to evaluate the *importance* of each input feature (each possible root cause) in the coarse-grained fault prediction. There exist techniques to directly evaluate such importance in simple models (e.g. decision trees), but it is well-known that this kind of *attention evaluation* is non-trivial for neural networks. While some generic techniques are applicable to any black-box model including ours [21], we instead propose to compute the *gradients* of the coarse predictions with respect to the input features. This method has already been tested in image analysis with great success [10], [22], and takes advantage of the fact that we can observe the internal weights and architecture of the coarse model (white-box setup). Given a coarse prediction $\mathbf{y} = (y_j)_{1 \leq j \leq c} \in \mathbb{R}^c$ (step ④ of Fig. 2), we first compute the *ideal label vector* $\mathbf{y}^\star$ that would have been given during the training for the input sample. (For readability we removed the i indices of all notations.)

$$\forall j \in \{1, \ldots, c\}, y_j^\star = \begin{cases} 1 & \text{if } \max(\mathbf{y}) = y_j \\ 0 & \text{otherwise} \end{cases}$$

We define $L^\star(\mathbf{y}) = -\sum_{j=1}^{c} y_j^\star \log y_j = -\log y_{\arg\max(\mathbf{y})}$ the cross-entropy loss that is minimal for the ideal label vector. By applying a single backpropagation step as done during the training phase, and thanks to the complete knowledge of the coarse model architecture, we can compute the *gradient* of this loss function with respect to the input features. We make the assumption that each partial derivative $\nabla_j = \frac{\partial L^\star}{\partial x_j}$ represents the *usefulness* of each feature j. It must be normalized according to the absolute value of ∇_j to account for both positive and negative derivatives.

$$\hat{\gamma}_{i,j} = \frac{|\nabla_j|}{\sum_k |\nabla_k|} \tag{1}$$

In our early experiments, we observed that the attention mechanism (Equation 1) used alone as a predictor of root causes gave inaccurate results. This is because a pure gradient-based backpropagation does not fully exploit the information provided by the multi-layer perceptron (④ in Fig. 2). To overcome this problem, we give a bonus to the most relevant root causes that belong to the same family fault as the most probable coarse cause returned by the coarse prediction. For instance, if the model predicts a *remote link latency* problem, we use this hint to increase the predicted usefulness of every latency-related feature while penalizing other features. The weighting mechanism is shown in Algorithm 1.

Algorithm 1: Multi-label score weighting

input : Predictions $\hat{\gamma}$ and coarse predictions $\mathbf{y}$
output: Tuned predictions $\hat{\gamma}'$

▷ *Isolate the best coarse prediction*
1 $\phi \leftarrow \arg\max(\mathbf{y})$
2 $p \leftarrow \{$*indices of features with same family as* $\phi\}$
▷ *Compute the relative weight*
3 $w \leftarrow \frac{y_\phi}{\sum y_i}; s \leftarrow \sum_{j \in p} \hat{\gamma}_j$
4 **if** $s = 0 \vee s = 1$ **then** $\hat{\gamma}' \leftarrow \hat{\gamma}$ ▷ *Extreme case*
5 **else**
6 **foreach** $j \in p$ **do** $\hat{\gamma}_j{}' \leftarrow \hat{\gamma}_j \frac{w}{s}$ ▷ *Bonus*
7 **foreach** $j \notin p$ **do** $\hat{\gamma}_j{}' \leftarrow \hat{\gamma}_j \frac{1-w}{1-s}$ ▷ *Penalty*

Given a coarse prediction vector $\mathbf{y}$, the algorithm first selects a set of features p related to the most significant class in $\mathbf{y}$ (in practice of the same family) at line 2. In our implementation, we manually assign each feature to a coarse class. Then, a ratio w is computed between the model's confidence in its coarse prediction and the sum s of related features' usefulness (line 3). The tuned $\hat{\gamma}'$ are computed in line 6 and line 7. By construction, Algorithm 1 always returns a normalized vector.

F. Ensemble model averaging

The architecture of Fig. 2 is designed to naturally extend to new landmarks without retraining. As a result, however, it loses information compared to more direct methods such as random forests. To further boost our solution, and reap the benefits of both worlds, we use *ensemble model averaging* as a last optimization step, a popular method to combine multiple specialized models [24]. We average the tuned attention predictions with another prediction from an *auxiliary* model, designed to be simpler and specialized in *known root causes*. We chose a random forest approach as our auxiliary model and give more insights about this choice in the next section.

We briefly formalize this last optimization. Let $\mathcal{U}$ be the set of unknown landmark's features, not seen during training. Let $\hat{\gamma}'$ and $\hat{\alpha}$ be the prediction obtained from the tuned attention mechanism and the auxiliary model, respectively. We define $w_\mathcal{U}$, the probability that the root cause is explained by an unknown landmark's features, as predicted by the tuned attention mechanism. Since $w_\mathcal{U} \in [0, 1]$ by definition, the final prediction of DIAGNET after model averaging is given by

$$w_\mathcal{U}\ \hat{\gamma}'\ +\ (1 - w_\mathcal{U})\ \hat{\alpha} \quad \text{with} \quad w_\mathcal{U} = \sum_{j \in \mathcal{U}} \hat{\gamma}'_j$$

IV. Evaluation

To evaluate DiagNet, we deployed a multi-cloud geo-distributed network of clients, online services, and landmark servers. In this section, we present our methodology and introduce baselines offering similar properties as DiagNet.

A. Experimental setup

a) Deployment: In order to train and evaluate the root cause analysis models, we deploy one landmark and multiple clients in each of the ten regions listed in Fig. 4. Three of these regions (GRAV, SEAT, SING) also host mock-up online services to evaluate the QoE with diverse setups (Table II). Some services only require a single HTML file, while others download resources from distant regions. (Recall that the nature of individual services, and hence the relations between regions and services are hidden during model training.) Region locations are chosen to benefit from both the diversity of a worldwide multi-cloud deployment and the proximity of co-located regions for fault localization. At the time of writing, our experimental pipeline was made of roughly 5,000 lines of Python and Go code. We used Tensorflow 1.13.1.

b) Landmark features: Live network metrics are obtained by querying each landmark through HTTPS endpoints [19]. To estimate download and upload throughputs, we measure the duration of large GET and POST HTTP requests. We avoid the classic overhead of HTTP requests for RTT estimation by upgrading the connection to WebSocket. Finally, we use the `getsockopt` linux syscall on each landmark server to make raw TCP statistics available to landmarks' clients. We mainly extract the ratio of reordered and retransmitted packets from these statistics.

c) Methodology: Clients periodically fetch network features from landmarks and visit mockup services to evaluate their QoE from `window.performance` JavaScript timings, both operations using an automated Chromium browser. Each client also periodically estimates the RTT to its local network gateway. We inject artificial network faults in each cloud region using Linux `tc` Network Emulator rules, a realistic and popular emulation method for reproducible experiments. QoE information was then used to flag samples as "nominal" or "faulty" with the (known) root-cause ground-truth as class label for model training. As presented in Section III-D, we first trained a *general* DiagNet model based on 8 mockup services, and then built a *specialized* model for each service by retraining only the last fully-connected layers. All the scores presented in this evaluation section are computed using the specialized models. We give more details about this method in Section IV-F, along with an evaluation of training cost.

d) Root cause extensibility: We trained and tested root cause models on two different sets of landmarks to assess the extensibility capabilities. For all experiments in this paper, three landmarks were "hidden" during training: EAST, GRAV and SEAT, named *new landmarks* and denoted by $\star$ in this paper, as opposed to *known landmarks* (the remaining seven). We chose these landmarks due to their immediate proximity to the mock-up services and several injected faults, and limited the availability of their features to model evaluation only. In doing so, we reduced the quality of the measures available to training, and made faults located close to the hidden landmarks particularly hard to detect, as neither these faults, nor the measures they impact most are used to train the models.

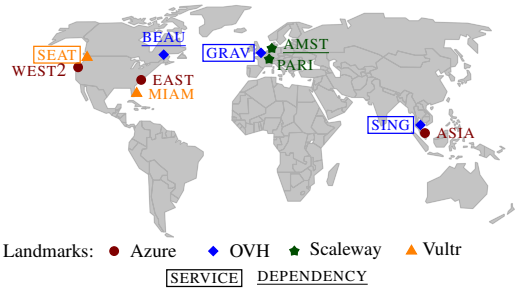


Fig. 4. Locations of landmarks and services in our multi-cloud experimental deployment. We deployed emulated clients in every location (region). EAST, GRAV and SEAT landmarks were hidden during training.

e) Dataset: We ran our experiment during the last two weeks of December 2019, using different hours of day and days of week to ensure large coverage of traffic and congestion patterns between cloud providers. 213,000 of "nominal" samples along with 30,000 "faulty" samples were collected during our experiment. 80% of each kind of samples were used for training, while the other 20% were reserved for testing. We injected the following 6 families of faults in regions involving services (SEAT, BEAU, GRAV, AMST and SING), leading to diverse fault scenarios: (i) download bandwidth shaping (capped at 8 Mbits/sec), (ii) additional service latency (50 msec), (iii) additional gateway latency (50 msec), (iv) additional jitter (up to 100 msec), (v) increased packet loss (8%), (vi) large CPU stress (impacts Chromium's navigation).

Faults were uniformly distributed between regions and families to avoid bias towards more frequent root causes. In some scenarios, we injected multiple faults at the same time, but at most one fault was the real root cause for QoE degradation in a given region. In many cases, we observed that the QoE was not degraded despite the injected fault(s). For instance, the QoE of a small HTML website was not affected by shaped bandwidth or CPU stress. We flag these samples as "nominal".

As explained just above, 3 landmarks out of 10 were hidden during training: samples with faults at these landmarks were forced to appear only in the testing set. Of these samples with "hidden faults", a fraction did not exhibit any QoE degradation, and were therefore flagged as "nominal", resulting at the end in 23% of the testing samples with degraded QoE to involve faults in hidden regions that were not seen at training time.

B. Comparison baselines

We propose two baselines that use common classification or outlier detection models and offer the same three key properties as DiagNet, namely location and topology agnosticism, along with root cause extensibility.

TABLE II
ONLINE SERVICES USED IN EXPERIMENTS.

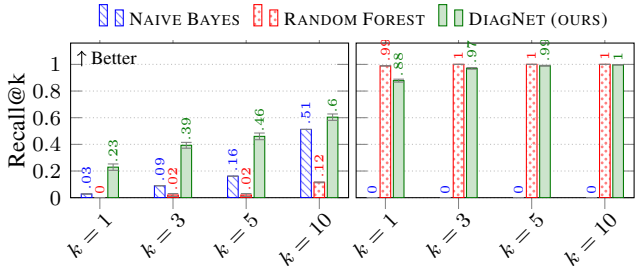
Service	Description
1. single	Static HTML page with no dependency
2. script.far	Requires a JS file in BEAU
3. script.cdn	Requires a JS file from nearest region
4. image.local	Loads a 5MB image from same server (using the same HTTP connection)
5. image.far	Loads a 5MB image from BEAU
6. image.cdn	Loads a 5MB image from nearest region

a) Extensible Random Forest Classifier: A random forest ensemble classifier is built by constructing a large set of small decision trees. The final classification is done through a majority vote on trees outcomes. This method has been previously used in failure classification in NetPoirot [25], showing great accuracy. To train an extensible random forest, we naively set the features dimension to the maximum possible size, and we set to zero the missing landmarks values in each sample. We also add a special "unknown" output class, selected when the given sample is classified as "nominal". We evenly redistribute the score obtained for this special class to every other class: this allow non-trained faults to have a non-null score in the final prediction. This model is used as-is in the ensemble averaging optimization presented in Section III-F.

b) Extensible Naive Bayes Classifier: We propose another approach for extensibility, based on the merger of several probability distributions. Using Bayes theorem and making the "naive" assumption that the value of one feature is not dependent from other features, it is easy to compute the posterior probability that one sample belongs to a class C_k given the estimated prior and likelihood probabilities. Equation 2 presents the application of the Bayes theorem for classification.

$$P(C_k \mid \mathbf{x}_i) \propto P(C_k) \prod_{j=1}^{m} P(x_{i,j} \mid C_k) \tag{2}$$

To add a basic support for model extensibility, we adapt the classic model in the following way. First, it is highly probable that one particular root cause C_k has not been seen during the training phase, and the prior probability for class k is unknown. Thus, we define the prior probability of each class C_k as $P(C_k) = 1$ for every root cause. This also has the positive side-effect of canceling bias with unbalanced datasets. Then, we use a Kernel Density Estimation (KDE) [26] function to construct the likelihood probabilities $P(x_{i,j} \mid C_k)$. In contrast with the more common Gaussian model, the KDE increases the expressivity of this baseline model. Finally, we build *generic aggregate likelihoods* for unknown features or new classes. For each measure family t collected in landmarks (such as uplink latency or download bandwidth), we build a *generic* likelihood $P(x_{i,t} \mid C_t)$, defined as the union KDE of the measures *for every landmark available during training*. This generic likelihood is used when no specific likelihood is available for a given feature or class.



(a) New landmarks (b) Known landmarks

Fig. 5. Recall values for failures near new and known landmarks, for different levels of recall k. DIAGNET consistently overperforms its competitors on new landmarks, while delivering close to ideal performances on known ones. By comparison RANDOM FOREST works perfectly for known landmarks, but degrades starkly on new ones, while NAIVE BAYES offers reasonable performance with new landmarks but is lost on known landmarks.

C. Recall evaluation

The final goal of root cause analysis is to return a *ranked list of probable causes*. We use the Recall@k metric for model evaluation: for a set of known real causes and a ranking method, the Recall@k is the number of correctly predicted causes *within the first* $k \geq 1$ *causes* divided by the total number of causes. A high recall would demonstrate that a method of ranking (model) can be useful to users, being able to quickly pinpoint the real root cause of a QoE degradation among a set of possible causes. In our setup, we argue that it is acceptable to return the expected cause within the first $k \leq 5$ predictions from 55 possible root causes.

Fig. 5 shows the Recall@k for two types of fault: faults injected near *new landmarks* in (a), and faults injected near *known landmarks* in (b). (As a reminder, new landmarks' features are hidden during training.) DIAGNET offers the best recalls for faults near new landmarks (a), thanks to its attention mechanism that fully exploits the information coming from the new features without additional training. Our proposal also shows close to ideal results for faults injected near known landmarks (b), thanks to the "hybrid" mode of operation offered by ensemble averaging (Section III-F). The combined Recall@1 for DIAGNET (including faults near known and new landmarks) is 73.9%, a very good score given the high number of probable root causes. By contrast, RANDOM FOREST works perfectly for known landmarks, but its recall degrades dramatically in the case of new landmarks; the described extensible random forest model essentially gives *completely random predictions* in this second case. By contrast, NAIVE BAYES shows extremely poor results for known landmarks, with its best score reached for high values of k in (a). This is due to a severe bias towards new features that systematically get high prediction scores *even for known failure types*.

Fig. 6 presents each recall per family of fault and per location. We clearly see NAIVE BAYES's bias towards some fault families and new landmarks GRAV and SEAT. DIAGNET is the only model showing good recalls for every family of fault near both known and new landmarks regions.

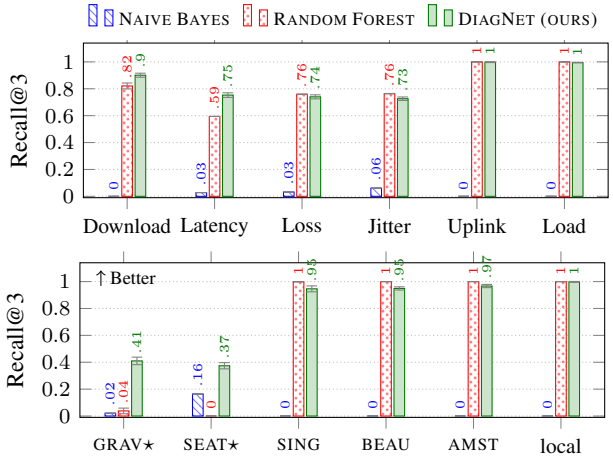


Fig. 6. Recall per fault family (top) and fault region (bottom). Regions hidden during training are indicated with a star ⋆. Again, RANDOM FOREST gives best results for known landmarks, but DIAGNET is the only solution able to adapt to the different scenarios, with close to optimal results for local faults.

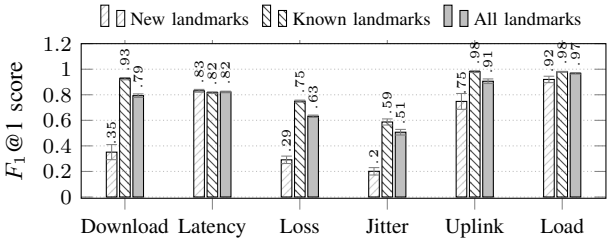


Fig. 7. Performance of the coarse classifier, with details for samples including known and new landmark's faults. The coarse classifier's accuracy is 0.70 ± 0.013 for faults near new landmarks and 0.85 ± 0.005 for faults near known landmarks (ratio of correct predictions over evaluated samples).

D. Evaluation of DIAGNET's coarse classifier

To shed light on the results just presented, and validate the design of DIAGNET's convolutional neural network, we evaluated the F_1 score of DIAGNET's coarse classifier (corresponding to step ④ in Fig. 2) for each fault family. As explained in Section III-E, the output of this coarse classifier is used by the attention mechanism and eventually averaged with the random forest classifier to provide the final classification. Thereby, we expect this coarse classifier to be critical for the final root cause ranking.

Fig. 7 presents separately the results for samples containing faults near known landmarks, and those with faults near new landmarks. As expected, samples with faults close to known landmarks are overall better classified than samples with faults close to unknown landmarks. The F_1 scores also show that some fault families are easier to classify than others (Latency, Uplink and Load). Overall scores demonstrate the value of DIAGNET's convolutional neural network.

E. Effect of client diversity

To validate the *location agnosticism* property of DIAGNET, we gradually increase the location *diversity* of participating clients. (Put differently, we vary the *number of regions with active clients submitting samples*.) The results of that experiment are shown in Fig. 8, with the aggregate Recall@5 for all families of faults near newly-introduced landmarks. For completeness, we note that we measured the Recall@5 for every possible combination of active clients to eliminate potential discrepancies between configurations. The key takeaway is that DIAGNET is able to give the best predictions for all scenarios of client diversity, showing great stability. Our results hint that DIAGNET is truly able to distinguish between dissimilar clients (e.g. clients in America vs. Asia or Europe).

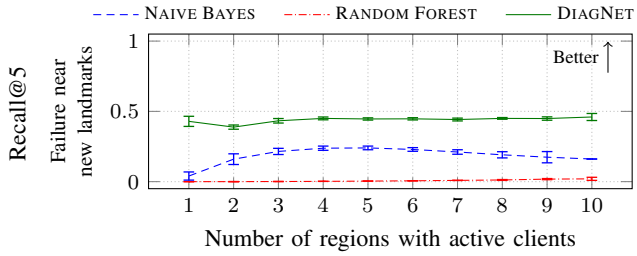


Fig. 8. Comparison of models' performance for new landmarks with increasing diversity of clients (we modified the number of regions with active clients). DIAGNET can scale very well for landmarks unseen during training.

In contrast, the NAIVE BAYES model prefers to handle few regions at a time. This is explained by the *KDE merge* process of this baseline: with more diverse clients, merged KDEs are "flattened" and converge to uniform distributions, biasing the model towards unknown features as seen in Fig. 5 and Fig. 6. RANDOM FOREST is less sensitive to client diversity, with only a slight recall increase probably due to the larger number of available training samples.

F. Training cost of new service models

To remove the need for the complete retraining of DIAGNET when new online services are being added, we assume that the weights learned in the non-overlapping convolution are shared between services, as they extract global network features; and that the final layers of DIAGNET capture the behavior of each service. We now give the details of the DIAGNET learning procedure, that has been used in the whole evaluation section and is based on that assumption. We first train a *general* model on a subset of eight initial services, taking the union of services' problems as the expected model output. Then, we freeze the weights of the non-overlapping convolution, and optimize the weights of the final layer for each of a set of additional services, not contained in the original set. This leads to one *specialized* model per additional service. With the hyperparameters from Table I, the general model must learn 215,312 parameters while specialized models hold 65,664 trainable parameters (the remaining 149,648 parameters are set to their value in the general model).

Learning losses on training and validation sets are plotted in Fig. 9 through learning epochs, for the general model and for a subset of service models. We consider that the training is done when the validation loss is no longer decreasing (an indication of overfitting). Although the training time on the

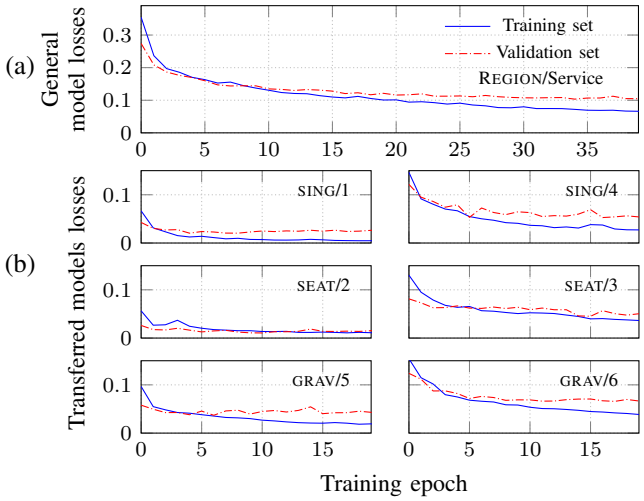


Fig. 9. Evaluation of model transferability: after building a general model on 8 services chosen uniformly at random (a), it is possible to build specialized models for other services while freezing convolutional kernels (b). A relatively low loss rate is quickly reached for most services.

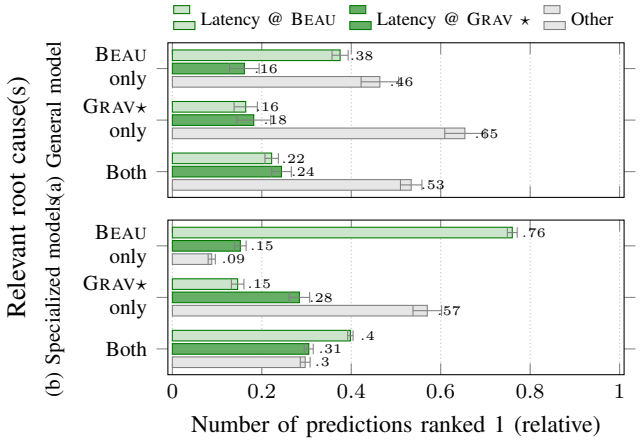


Fig. 10. Predicted root causes for general and specialized DIAGNET models, with simultaneous injected faults (latency near the BEAU and GRAV regions). Depending on the evaluated service, the relevant root cause(s) causing the QoE degradation is either one of the two injected faults or both (y-axis). GRAV metrics were not used during training (⋆). Overall, the specialized models (lower chart) deliver better predictions for all 3 combinations of relevant faults.

general model is higher (around 20 learning epochs), service models converge in less than 5 epochs on average. This indicates that specialized service models per service are easy to learn once one global model exists. On a commodity laptop using the CPU only, it takes 32 seconds to train the general model and 4 seconds to train each service model. Root causes are inferred in 45 ms on average.

G. Accuracy with simultaneous faults

The critical task of a root cause model is to find the *correct* root cause when multiple simultaneous anomalies are detected. DIAGNET uses specialized models *for each web service*: some services might be impacted by a given anomaly, when other services might not. To evaluate this property, we simultaneously injected two latency faults near the BEAU and GRAV regions and quantified the number of predictions towards each region. Fig. 10 gives the detailed results for DIAGNET's general model (a) and for the specialized model of each service (b). The results for the general model are quite poor, with a confusion between BEAU and GRAV regions and a lot of other faults predicted. Specialized models provide sharper predictions, with a recall of 76% for the latency root cause near BEAU, 28% for the latency root cause near GRAV—a region unseen during training—and 71% when both faults are actually root causes. This analysis confirms the benefit to specialize models for each monitored service.

V. RELATED WORK

Root cause analysis solutions either rely on *passive* or *active* measurements. Passive measurement relies on existing traffic, does not introduce any overhead and is therefore used in large-scale systems [18], [27]. Depending on the setup, different sources of measurements are available such as local system or router insights [25], [28], [29], request path annotations [30], [31], routing monitors [32] or more recently from Internet background radiation [33]. Still, passive observations fall short in low-traffic environments with little information about the underlying network architecture and no control over routing paths [34]. DIAGNET leverages active probing to perform accurate root cause analysis from end-devices alone, that have a very narrow view of the underlying network topology. We note that our method can also be used in conjunction with passive monitoring for bootstrapping a system [18] or testing hypotheses [35].

Numerous works have been performed in enterprise networks and datacenters, where the full network topology is known (e.g. Clos-like topologies or SDN-driven networks [16], [17], [36]). This topology information allows network tomography techniques to be applied [11]–[15], [37], [38], pinpointing faulty links or components accurately at scale despite complex dependencies between components [39].

From a metrics perspective, root cause analysis tools are often specialized towards a set of metrics and thus a narrow set of faults. For example, some focus on poor TCP statistics [11], [14], [25], on invalid BGP announcements [28], [32] or Virtual Hard Drive failures [39]. Netalyzr [4] and Fathom [5] collect end-user connectivity statistics to propose automated troubleshooting using predefined expert rules. By contrast, our work keeps a generic approach. Nevertheless, the specific metrics of the aforementioned methods are complementary and could be used as additional input features if available.

Regarding machine learning methods, belief networks have been used [40], [41] to model the complex dependencies between network components and online services. However, such methods require many approximations in the modeling to remain tractable, while being very sensitive to errors in topology identification [42], [43]. Random forest models are also known to perform well in understanding network failures,

as demonstrated by NetPoirot [25]. In contrast, DIAGNET supports variable number of input features (landmarks) using convolutional neural networks. DIAGNET relies on the good expressivity of non-linear models to learn features dependencies. It ensures a low training cost by proposing generic models applicable to multiple network configurations and services without complex dependency modeling.

Crowd-sourcing measurements and root cause analysis is a promising approach, where multiple vantage points share their results to better estimate fault root causes [18], [41], [44], [45]. Distributed Hash Tables (DHT) have historically been used for that purpose [46], [47], ensuring the scalability of such decentralized systems. This line of work is complementary to DIAGNET: while we currently assume that the analysis process and data collection is handled by a centralized location, DHTs or other distributed system approaches could be used to distribute the root cause analysis service.

VI. CONCLUSION

Root cause analysis at the scale of the Internet is recognized as a hard problem given the decentralized design of the network. In this work, we have proposed DIAGNET, a generic and extensible root cause analysis method based on active landmark probing. DIAGNET does not depend on prior network topology or service knowledge which makes it practical for end-users that have a very limited view of the Internet topology past their gateway. The inference model of DIAGNET relies on a new type of convolutional network and attention mechanism, along with several optimizations (multi-label score weighting and ensemble model averaging). While we demonstrated that Random Forest models can be very insightful when diagnosing in a *static* setting and that Naive Bayesian approaches can also be leveraged for some faults in more dynamic settings, DIAGNET shows good results in *all* scenarios, i.e. it can diagnose local and remote failures in static and dynamic network settings, even with very diverse participating clients from across the globe.

REFERENCES

[1] D. Z. Joumblatt *et al.*, "Predicting User Dissatisfaction with Internet Application Performance at End-Hosts," in *INFOCOM*, 2013.

[2] H. Nam *et al.*, "QoE matters more than QoS: Why people stop watching cat videos," in *INFOCOM*, 2016.

[3] G. Dimopoulos *et al.*, "Identifying the Root Cause of Video Streaming Issues on Mobile Devices," in *CoNEXT*, 2015.

[4] C. Kreibich *et al.*, "Netalyzr: Illuminating The Edge Network," in *IMC*, 2010.

[5] M. Dhawan *et al.*, "Fathom: a browser-based network measurement platform," in *IMC*, 2012.

[6] S. Sundaresan *et al.*, "Broadband Internet Performance : A View From the Gateway," in *SIGCOMM*, 2011.

[7] Y. Jin *et al.*, "NEVERMIND, the problem is already fixed: proactively detecting and troubleshooting customer DSL problems," in *CoNEXT*, 2010.

[8] Y. LeCun *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, 1989.

[9] K. He *et al.*, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *TPAMI*, vol. 37, no. 9, 2015.

[10] K. Simonyan *et al.*, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *CoRR*, 2014.

[11] D. Rubenstein *et al.*, "Detecting shared congestion of flows via end-to-end measurement," *Transactions on Networking*, vol. 10, no. 3, 2002.

[12] N. Duffield, "Network Tomography of Binary Network Performance Characteristics," *IEEE Transactions on Information Theory*, vol. 52, no. 12, dec 2006.

[13] A. Dhamdhere *et al.*, "NetDiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data," in *CoNEXT*, 2007.

[14] B. Arzani *et al.*, "007: Democratically Finding the Cause of Packet Drops," in *NSDI*, 2018.

[15] C. Tan *et al.*, "NetBouncer : Active Device and Link Failure Localization in Data Center Networks," in *NSDI*, 2019.

[16] P. Tammana *et al.*, "Fault localization in large-scale network policy deployment," in *ICDCS*, 2018.

[17] Y. M. Ke *et al.*, "SDNProbe: Lightweight fault localization in the error-prone environment," in *ICDCS*, 2018.

[18] V. N. Padmanabhan *et al.*, "NetProfiler: Profiling wide-area networks using peer cooperation," in *IPTPS*, 2005.

[19] L. Bonniot *et al.*, "DiagSys: network and third-party web-service monitoring from the browser's perspective (industry track)," in *Middleware*, 2020.

[20] D. N. da Hora *et al.*, "Narrowing the Gap Between QoS Metrics and Web QoE Using Above-the-fold Metrics," in *PAM*, 2018.

[21] M. T. Ribeiro *et al.*, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *SIGKDD*, 2016.

[22] R. R. Selvaraju *et al.*, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *ICCV*, 2017.

[23] R. Eskola and J. K. Nurminen, "Performance Evaluation of WebRTC Data Channels," in *ISCC*, 2015.

[24] D. Madigan *et al.*, "Bayesian Model Averaging," in *AAAI*, 1996.

[25] B. Arzani *et al.*, "Taking the Blame Game out of Data Centers Operations with NetPoirot," in *SIGCOMM*, 2016.

[26] M. Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function," *The Annals of Mathematical Statistics*, vol. 27, 1956.

[27] R. N. Mysore *et al.*, "Gestalt: Fast, Unified Fault Localization for Networked Systems," in *ATC*, 2014.

[28] H. Yan *et al.*, "G-RCA: A generic root cause analysis platform for service quality management in large IP networks," in *CoNEXT*, 2010.

[29] A. Roy *et al.*, "Passive Realtime Datacenter Fault Detection and Localization," in *NSDI*, 2017.

[30] M. Chen *et al.*, "Path-based failure and evolution management," in *NSDI*, 2004.

[31] R. Fonseca *et al.*, "X-trace: A pervasive network tracing framework," in *NSDI*, 2007.

[32] V. Giotsas *et al.*, "Detecting Peering Infrastructure Outages in the Wild," in *SIGCOMM*, 2017.

[33] A. Guillot *et al.*, "Chocolatine: Outage Detection for Internet Background Radiation," in *TMA*, 2019.

[34] W. W. Fok *et al.*, "MonoScope: Automating network faults diagnosis based on active measurements," in *IM*, 2013.

[35] I. Rish *et al.*, "Real-time problem determination in distributed systems using active probing," in *NOMS*, 2004.

[36] I. A. Stewart, "A general algorithm for detecting faults under the comparison diagnosis model," in *IPDPS*, 2010.

[37] R. Castro *et al.*, "Network Tomography: Recent Developments," *Statistical Science*, vol. 19, no. 3, 2004.

[38] D. Ghita *et al.*, "Netscope: Practical network loss tomography," in *INFOCOM*, 2010.

[39] Q. Zhang *et al.*, "Deepview: Virtual Disk Failure Diagnosis and Pattern Detection for Azure," in *NSDI*, 2018.

[40] P. Bahl *et al.*, "Towards highly reliable enterprise network services via inference of multi-level dependencies," in *SIGCOMM*, 2007.

[41] E. Kiciman *et al.*, "Fast Variational Inference for Large-scale Internet Diagnosis," in *NIPS*, 2008.

[42] M. Coates *et al.*, "Maximum likelihood network topology identification from edge-based unicast measurements," in *SIGMETRICS*, vol. 30, no. 1, 2002.

[43] Y. Huang *et al.*, "Practical issues with using network tomography for fault diagnosis," *SIGCOMM CCR*, vol. 38, no. 5, 2008.

[44] D. D. Clark *et al.*, "A Knowledge Plane for the Internet," in *SIGCOMM*, 2003.

[45] H. J. Wang *et al.*, "Automatic Misconfiguration Troubleshooting with PeerPressure," in *OSDI*, 2004.

[46] D. R. Choffnes *et al.*, "Crowdsourcing service-level network event monitoring," in *SIGCOMM*, 2010.

[47] K. H. Kim *et al.*, "DYSWIS: Crowdsourcing a home network diagnosis," in *ICCCN*, 2014.