DiagNet: towards a generic, Internet-scale root cause analysis solution

Loïck Bonniot, Christoph Neumann, François Taïani

▶ **To cite this version:**

Loïck Bonniot, Christoph Neumann, François Taïani. DiagNet: towards a generic, Internet-scale root cause analysis solution. 2020. hal-02534888v1

HAL Id: hal-02534888

https://hal.science/hal-02534888v1

Preprint submitted on 7 Apr 2020 (v1), last revised 18 May 2021 (v2)

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.

DIAGNET: towards a generic, Internet-scale root cause analysis solution

Loïck Bonniot
InterDigital
Univ Rennes, Inria, CNRS, IRISA
loick.bonniot@interdigital.com

Christoph Neumann
InterDigital
christoph.neumann@interdigital.com

François Taïani
Univ Rennes, Inria, CNRS, IRISA
francois.taiani@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 an end-user's device and the service datacenters. 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 different machine learning techniques can be used for Internet-scale root cause analysis using measurements taken from end-user devices. 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. Our solution, DIAGNET, adapts concepts from image processing research to handle network and system metrics. We evaluate DIAGNET with a multi-cloud deployment of online services with injected faults and emulated clients with automated browsers. We demonstrate promising root cause analysis capabilities, with a recall of 73.9% including causes only being introduced at inference time.

I. INTRODUCTION

Both content providers and Internet service providers (ISPs) strive to provide the best service to their customers, and allocate very significant resources to diagnose and troubleshoot end-user problems. For instance, an ISP should ideally be able to immediately detect and explain a service degradation to its users. Unfortunately, the reason for a problem might lie anywhere between the customer's home and the final data center, and many of the locations involved are not controlled by the ISPs. Worse, as services grow more complex and interdependent, it is becoming increasingly hard to ascertain whether a given perturbation somewhere in the Internet is the cause of a customer's trouble, causing ISPs' customer support to often struggle to diagnose the root cause of an incident [1], [2]. Similarly, content providers closely monitor the Quality of Experience (QoE) of their users across the globe, and seek to rapidly resolve any observed degradation, as even a small drop in QoE can have a tremendous impact in revenue and brand image [3]–[5]. However, it is often tedious for content providers to quickly pinpoint the location of faults, as this often requires costly human expertise to understand whether a degradation is due to their own internal infrastructure or to weak Internet peering to specific ISPs, for instance.

To improve on this situation, numerous prior works have proposed to exploit end-user devices and equipment to diagnose on-line incidents [6]–[8]. These works adopt two main strategies. The first is to execute a set of predefined tests, designed by experts in networking, and use outliers to propose a diagnostic [1], [7]–[9]. These tests are very efficient to detect known configuration issues (DNS failures, aggressive firewall, low quality uplink, ...), but are specific to some technologies (like DNS, TCP or DSL access specific problems [10]), and fall short in understanding more distant Internet failures. The second strategy is to use a shared service status database like downdetector.com to easily discriminate between local or distant fault. Unfortunately, such services are usually centralized and only offer coarse-grained analysis based on manual flagging: as an example, if for a given service a large number of reports usually come from Germany, the only thing that can be inferred is that many users of the service in Germany are encountering a fault and are willing to share that information. The precise root cause location is still to be determined. While the above solutions either provide important insights on service availability or focus on important types of faults, they cover only a small and specific part of the possible root causes for many online services and fail at offering generic internet-scale root cause inference.

To overcome these limitations, we propose DIAGNET, a generic and extensible crowd-sourced root cause analysis method based on data collected from user devices. DIAGNET uses browsers to take measurements, and exploits a versatile inference model to diagnose problems proactively before calling the customer support of the ISP or content provider. Our system relies on a neural network for root cause inference model that does not make any assumption on the underlying network topology and can ingest new types of network measurements without the need for retraining. DIAGNET is based on a set of *landmark servers* that act as reference points for measures. We assume these servers are opportunistically deployed over diverse parts of the Internet independently from any network operator. DIAGNET leverages attention mechanisms, a technique to highlight the input features that were relevant for a particular classification result, in combination with non-overlapping convolutional kernels and pooling mechanisms, borrowing and extending state-of-the art concepts from the image processing commu-

nity [11]–[13]. Doing so, DIAGNET is the first Internet-scale network-diagnostic solution that can infer root causes it never encountered before, and can easily be adapted to different types of online services with very little retraining, while only requiring lightweight, easy to obtain user-side measurements. The principles behind DIAGNET are further not limited to end-user problems, and generalize easily beyond Browser-based services, to distributed B2B APIs.

Our main contributions are as follows:

- We define a set of properties that must be satisfied for end-user root cause analysis in today's Internet, with no external information on the network topology or inter-services dependencies.
- We propose a simple root cause analysis architecture based only on measurements from end-users devices and a dynamic set of landmark servers. While in our implementation and evaluation we chose to focus on measurements available within a browser, any client metric could be exploited by the proposed architecture and models.
- We build DIAGNET, a root cause inference model that can handle an extensible set of network measurements and therefore pinpoint locations it never encountered before. The proposed model introduces new types of convolutional layers as well as attention mechanisms, which make the model generic and extensible.
- We evaluate our proposal on mock-up online services and clients deployed in 10 world regions and relying on multiple cloud providers with various dependencies between services. We compare DIAGNET against simpler, yet recognized inference methods and show that it consistently overperforms its competitors in a dynamic context, that is typical of today's Internet services, while delivering close to ideal performances in a static setting. More specifically, DIAGNET is able to pinpoint root causes with a Recall@1 of 73.9%, including non-trained root-causes.

The remainder of this paper is structured as follows. We start by specifying our problem and the set of required properties for DIAGNET in section II. In section III, we dive in the internals of our proposal, from the architecture overview to the predictions fine-tuning. Based on a geo-distributed collection of metrics, we propose in section IV an extensive evaluation of DIAGNET, alongside with two baseline proposals for comparison. We present related work in section V, and section VI concludes this document.

II. PROBLEM STATEMENT AND GOALS

After a brief overview of our system's model, we introduce in this section three key properties that we argue are required to design a root cause inference method that is generic and can work at Internet-scale: *network topology agnosticism*, *location agnosticism*, and *root cause extensibility*.

A. System overview

Our vision is that of a *central root cause analysis service* that is reachable from any end-user device (also termed *client*). This service continuously processes measures provided by a subset of clients to maintain a model of the network. This crowd-sourced model is then able to diagnose failures of online services consumed by end-users. We focus in this paper on the design and construction of this central model, but leave the implementation details of the crowd-sourcing mechanisms that are necessary to aggregate individual measurements to future work. Clients produce measurements by actively probing *landmark servers* (see Fig. 1), i.e. stateless public HTTP services that can be provided by different ISPs or third parties. The global network of Speedtest servers [14] is an example of practical public landmark servers deployment. More specifically, we leverage modern web browser capabilities to fetch TCP statistics, latency and bandwidth information from these landmark servers, to which we add some *local system features* (e.g. client CPU and memory load) measured on the client itself. Within a browser this can either be implemented as a JavaScript that is fetched when accessing the online service (the solution we have used for our prototype), or as a browser extension. While we wanted to keep a very simple and restricted set of metrics to bootstrap our work, it is absolutely possible to add more specific measures as additional inputs.

The measures collected by a client i form a vector of m measures $\mathbf{x}_i = (x_{i,j})_{1 \leq j \leq m} \in \mathbb{R}^m$. They constitute 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 [15]. From that data, and assuming that a user is encountering some QoE degradation, our ambition is to propose a ranked list of probable root causes that *explain* this degradation. We design our root causes to be the combination between a possibly coarse-grained *location* and a *fault family*, e.g "abnormal latency within the AWS US east coast region" or "high jitter within local WiFi connection".

Our root cause analysis should rank the probable explanation by decreasing probability while ensuring their *usefulness* for the end-user, informally defined as the *additional intelligence given by that feature to locate and understand a specific QoE degradation*. In our system model, each input feature is representative of a root cause: as an example, if a user encounters a QoE degradation while accessing a video streaming service, and the download bandwidth to a landmark located in Spain abruptly decreases, it makes sense to pinpoint the feature "Spain landmark download" as probable root cause. However, if other landmarks suffer from the same bandwidth degradation, but the local computing load of the client also reported an increase, the "local CPU load" feature shall be

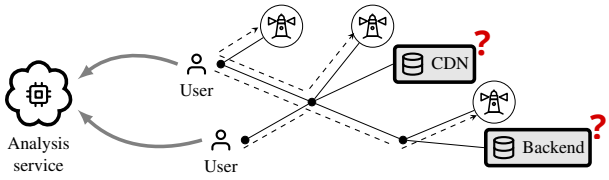

Fig. 1. Toy example of a topology for an online-service relying on a CDN and a backend. Users can evaluate links (solid lines) by actively probing landmarks (dashed lines). Probes are sent to a root cause analysis service, which builds and shares the root cause inference model.

marked as the most probable root cause. Providing that many features are available, this makes our model very expressive without the need for manual expert annotation. This analysis is very different from the problem of *feature selection*, where a restricted set of features are selected for model training. In our case, a useful feature will help understanding and troubleshooting a QoE degradation; as such usefulness must be computed anew for *each diagnostic*. We denote the usefulness of feature j in sample i by $\gamma_{i,j}$.

B. Network topology 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 a range of autonomous systems and operators networks. The underlying network architectures and topologies of all these systems are complex, continuously evolving and often unknown. We argue therefore that an Internet-scale root cause analysis method should not make any assumptions on the hidden network topology, a property that we call *network topology agnosticism*. This largely departs from common root cause analysis that relies on network tomography [16]–[19] or bespoke methods for data centers [20], [21] and Software-Defined-Networking [22], [23].

Root cause analysis requires however some location information to pinpoint the area (e.g. cloud region, point of presence, autonomous system) in which a root cause is likely to be. DIAGNET relies 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 vicinity of targeted services, or in the path towards them, thus offering telling clues regarding the location and family of the incident impacting a user.

Of course, it is probable that some features vary greatly without any impact on the target service. By understanding the relations between features and service's performance, a model could infer useful glimpses of information about a network's internal architecture, without exhaustive and costly modeling of the full network topology. This type of analysis shall be largely sufficient for a user to *understand* the kind of encountered failure, and eventually for an operator to start a deeper investigation with good hints from its customer's devices: this is a first light in the dark for accurate root cause analysis from end devices.

C. Location agnosticism

Historically, crowd-sourced diagnostic tools have exploited the precise location of every client [24], both at a geographical (from neighborhood to country) and topological level (from subnet to ISP), to pinpoint failures accurately. However, obtaining such detailed data from every end-device is often difficult and even undesirable: users might refuse to share their precise location out of privacy concerns, while IP addresses are becoming increasingly difficult to locate due to voluntary obfuscation, carrier-grade NATs, roaming and ongoing deployment of IPv6.

In DIAGNET, we propose to circumvent the need for location altogether, and argue instead that a root cause analysis model should be *location-agnostic*: the same single model should apply to every participant in a crowd-sourcing network. 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. The need for extensibility

In our system model, each feature is representative of a root cause. In practice, each client takes a number of active network-based measurements from available landmarks, which are then fed along with local metrics into the DIAGNET inference model. Many factors can however alter the *availability* of these landmarks (for instance failures, maintenance, network partition or saturated capacity). Conversely, if the system contains a very high number of landmarks, individual clients can not be expected to probe every landmark in order to keep overheads low. As an extreme example, it would require at least 94 000 landmark servers to cover every autonomous system[1], a number clearly too high for exhaustive 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 critical property allows for easy maintenance of the landmarks fleet, that should be as large and diverse as possible. 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 are available. This implies a number of critical choices in the design of our proposal to avoid frequent model retraining.

[1]Data from Regional Internet Registries as of January 1, 2020

III. DIAGNET

Similarly to most statistical classification models, DIAGNET takes as input a vector $\mathbf{x}_i \in \mathbb{R}^m$ of features measured by a client c_i (a "sample") and outputs a probability vector of likely classes (the predicted root causes). Contrary to a typical statistical classifier, however, DIAGNET can adapt without re-training to a variable number of input features m (provided by a varying number of landmarks), while supporting a dynamic number of diagnostic classes.

A. General architecture

Our strategy consists in building models that in a first phase only predict the *family* of encountered faults (if any) without any information on the location or on the root cause features (what we call a *coarse prediction* in the following). The number of fault families c is fixed, and the resulting prediction is a small-size vector $\mathbf{y} \in [0,1]^c$ of probabilities. We arbitrary selected the following set of common families: *nominal* (non-faulty); *uplink latency* for gateway malfunction; *remote link latency*, *link jitter*, *link loss* and *link download bandwidth* for end-to-end issues not related to the local uplink; and *local load* for client device overload. By keeping the coarse dimension low and constant ($c \ll m$), we build accurate models without an excessive number of samples.

In a second phase, DIAGNET uses the coarse prediction vector $\mathbf{y} \in [0,1]^c$ 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 can use any attention mechanism in that step: such mechanisms infer the *weight* of each input feature in the coarse model's prediction, often without any additional training.

The global architecture of DIAGNET 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 subsection III-B), ③ ④ processing all features with a fully-connected network and obtaining the final coarse prediction (detailed in subsection III-C). The second phase involves the step of ⑤ returning back to the input features via attention mechanisms (subsection III-D).

B. Non-overlapping convolutions with pooling

In image analysis, convolutional neural networks [11] 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. RTT, throughput) recorded by c_i w.r.t to the landmark λ. For example, $x_{i,1}[\lambda]$ might store the RTT 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

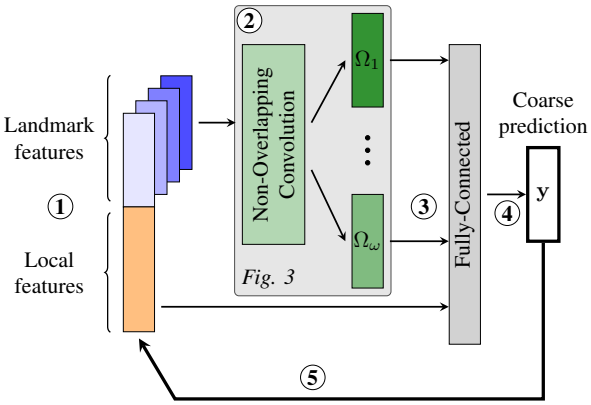

Fig. 2. Architecture of DIAGNET. ① Landmark features are first separated from local features and ② fed in the LandPooling layer with multiple parallel global pooling operations. ③ A hidden fully-connected layer is applied after concatenating the LandPooling output with local features. ④ The coarse fault 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.

non-overlapping convolutions to each client/landmark measure vector $\mathbf{x}_i[\lambda]$. These convolutions are captured by a kernel $\mathbf{K} \in \mathbb{R}^{f \times k}$ and 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 [12], [25], 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. The hyperparameters and global functions we used in our implementation of DIAGNET are listed in Table I.

C. Tailoring to specific services

Similarly to classical classification tasks relying on convolutional architectures, we propose to 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

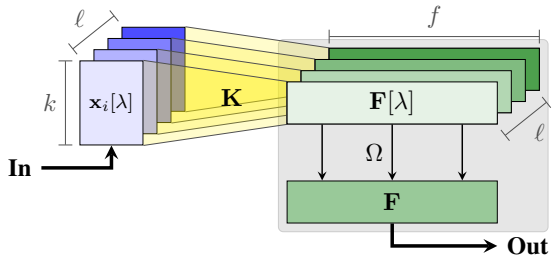

Fig. 3. Graphical view of the proposed non-overlapping convolutional layer with pooling (LandPooling). For each landmark λ, the k features of that landmark $\mathbf{x}_i[\lambda]$ are transformed to a new feature space $\mathbf{F}[\lambda]$ of size f through a shared kernel $\mathbf{K}$. To return a 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.

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
	Fully-Connected layers: 2 hidden layers $(512 \times 1), (128 \times 1)$
	Learning algorithm: SGD with Nesterov momentum
	(learning rate = 0.05, decay = 0.001)

and the local features resulting in coarse fault predictions. As illustrated in Fig. 2, this perceptron (also called "Fully-Connected layers") accepts 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 variable 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 absolutely critical in multiplayer games, it might intuitively not be the case for video streaming systems where the 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 subsection IV-E.

D. 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 [26], 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 [13], [27], 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.[2]

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

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, and can 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 very 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 (e.g. latency, bandwidth) 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.

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 8 and line 9. By construction, algorithm 1 always returns a normalized vector.

E. 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

[2]For readability, and without ambiguity since we are now working on a single sample i, we removed the i indices of all notations.

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*
$\phi \leftarrow \arg\max(\mathbf{y})$
$p \leftarrow \{$*indices of features with same family as* $\phi\}$

▷ *Compute the relative weight*
$w \leftarrow \frac{y_\phi}{\sum y_i}$
$s \leftarrow \sum_{j \in p} \hat{\gamma}_j$
if $s = 0 \vee s = 1$ **then**
  $\hat{\gamma}' \leftarrow \hat{\gamma}$ ▷ *Extreme case*
else
  foreach $j \in p$ **do** $\hat{\gamma_j}' \leftarrow \hat{\gamma}_j \frac{w}{s}$ ▷ *Bonus*
  foreach $j \notin p$ **do** $\hat{\gamma_j}' \leftarrow \hat{\gamma}_j \frac{1-w}{1-s}$ ▷ *Penalty*

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 [28]. 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

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 and Table II. Three of these regions (GRAV, SEAT, SING) also host mock-up online services to evaluate the QoE with diverse setups. 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 5000 lines of Python and Go code. We used Tensorflow 1.13.1 as our machine learning framework.

Landmark features. Live network metrics (Table III) are obtained by querying each landmark through HTTPS endpoints. This choice allows end-users to access landmark features via their web browsers only. To estimate download and upload bandwidths, we measure the time required for large GET and POST HTTP requests. We avoided the classic overhead of HTTP requests for Round Trip Time (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. (For completeness, we add that we used the BBR congestion algorithm for every communication.)

Methodology. Clients periodically fetch network features from landmarks and visit mockup services to evaluate their QoE from performance timings, both operations using a headless Chromium process. We inject artificial network faults using Linux `tc` Network Emulator rules [29], 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.

Root cause extensibility. We trained and tested root cause models on two different sets of landmarks to assess the extensibility capabilities of our approach. For all experiments in this paper, three landmarks were "hidden" during training: EAST, GRAV and SEAT, named *new landmarks* 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.

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. We injected the following 6 families of faults in different regions, leading to 24 different fault scenarios:

- Download bandwidth shaping (capped at 8 Mbits/sec),
- Additional service latency (50 msec),
- Additional gateway latency (50 msec),
- Additional jitter (up to 100 msec),
- Increased packet loss (8%),
- Large CPU stress (this is critical for headless Chromium).

Faults were uniformly distributed between regions and families to avoid bias towards more frequent root causes. In many cases, we observed that the QoE was not degraded despite the injected fault(s). For instance, the QoE of a small HTML

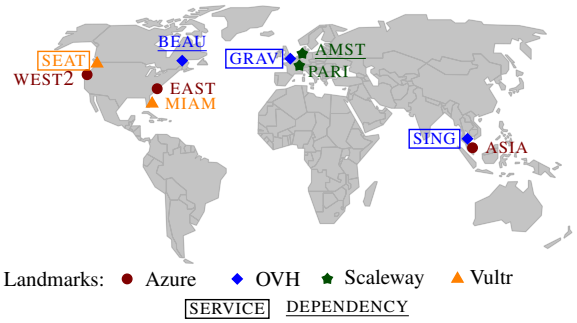

Fig. 4. Locations of landmarks and services in our multi-cloud experimental deployment. We emulated clients in every location (region).

TABLE II
REGIONS AND ONLINE SERVICES USED IN EXPERIMENTS.

Provider	Regions
Azure	US (EAST, WEST2), Singapore (ASIA)
OVH	Beauharnois (BEAU), Gravelines (GRAV) Singapore (SING)
Scaleway	Paris (PARI), Amsterdam (AMST)
Vultr	Miami (MIAM), Seattle (SEAT)
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 the nearest region
4. image.local	Loads a 5MB image from the same server (using the same HTTP connection)
5. image.far	Loads a 5MB image from from BEAU
6. image.cdn	Loads a 5MB image from the nearest region

website was not affected by shaped bandwidth or CPU stress. In this case we flag the samples as "nominal". 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.

B. Comparison baselines

To the best of our knowledge, we are the first to propose a root cause analysis method that is *extensible*, both in the features and causes dimensions, and that does not exploit additional information on network dependencies. We propose two baselines that use common classification models and offer the same three key properties as DIAGNET, namely location and topology agnosticism, along with root cause extensibility.

TABLE III
FEATURES COLLECTED DURING OUR EXPERIMENTS. THIS SET OF FEATURES CAN EASILY BE EXTENDED.

Type	Features
Service QoE	Timings from `window.performance` JavaScript (full page and per resource)
Landmarks (x10)	Download, Upload, Round-trip time, number of TCP reorderings and retransmissions
Local	Total and available memory, disk and CPU load, round-trip time to gateway

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 is well-understood and has been previously used in failure classification in NetPoirot [30], showing great stability and accuracy in the face of diverse machines. 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 subsection III-E.

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.
- Second, we use a Kernel Density Estimation (KDE) [31] 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 $P(x_{i,t} \mid C_t)$. This generic likelihood is the union KDE of the measures *for every landmark available during training*, and becomes the default when no specific likelihood is available for a given feature or class.

C. Recall evaluation

The final goal of root cause analysis is to return a *ranked list of probable causes* to users and operators. We propose to leverage 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* $k \geq 1$

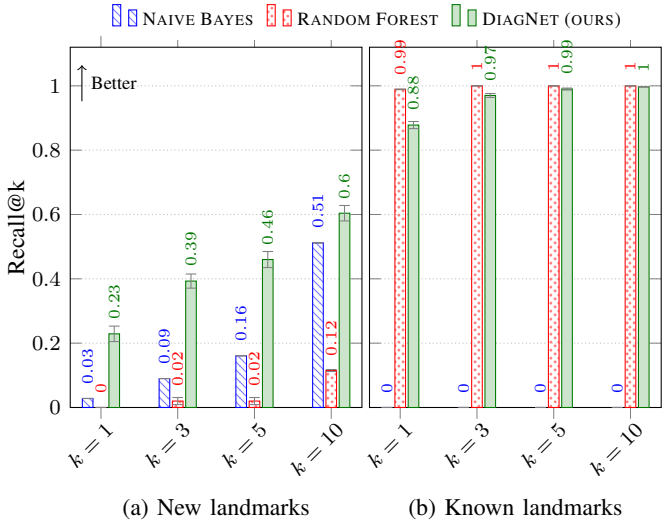

Fig. 5. Evaluation of models recalls for failures near new and known landmarks, for different levels of recall k. DIAGNET consistently overperforms its competitors on new landmarks, while delivering close to ideal performances on known ones. By comparison RANDOM FOREST works perfectly for known landmarks,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

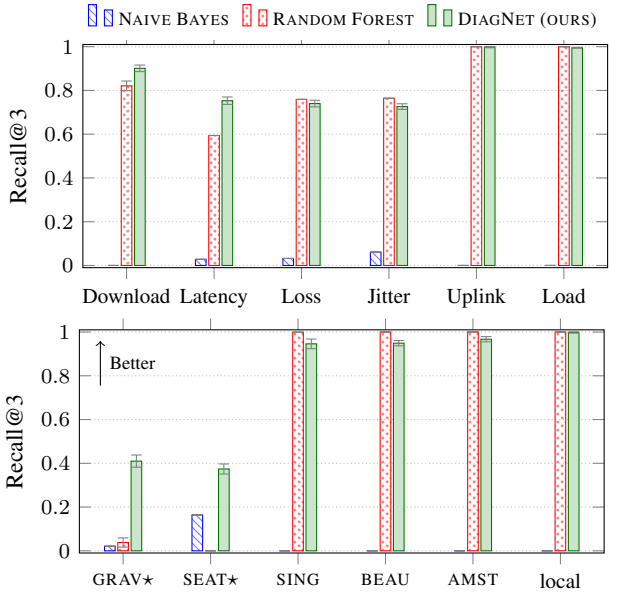

Fig. 6. Detail of models recalls per fault family (top) and fault region (bottom). Regions hidden during training are indicated with a star ⋆. Again, RANDOM FOREST gives best results for known landmarks, but DIAGNET is the only solution able to adapt to the different scenarios, with optimal results for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

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 (subsection III-E). 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. This is understandable, as 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*.

Diving into the details, 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 demonstrating its versatility with good recalls for every family of fault in both known and new landmarks regions.

D. 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. 7, with the aggregate Recall@5 for all families of faults near newly-introduced landmarks (a) and near known landmarks (b). 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 take-away 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).

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 in the (a) case, probably due to the growing number of training samples.

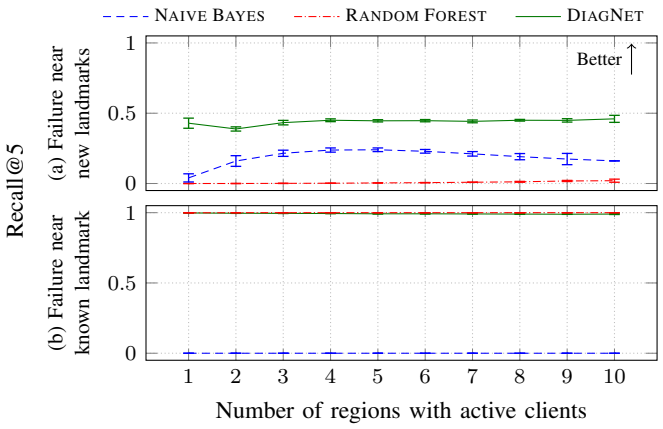

Fig. 7. Comparison of models' performance with increasing diversity of clients as the number of regions with active clients. DIAGNET can scale very well for both known and unknown landmarks, with stable recall. The NAIVE BAYES approach seems to be optimal at 5 different regions with users, and does not scale with more available regions.

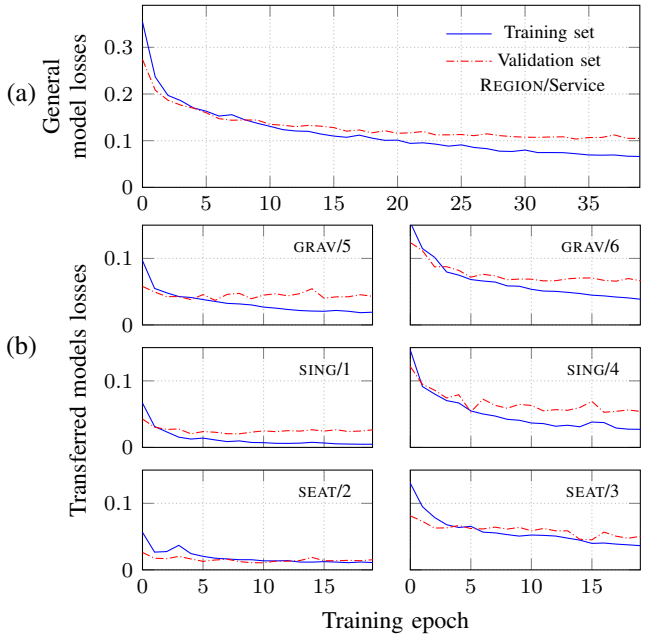

Fig. 8. Evaluation of model transferability: after building a general model on 8 services chosen uniformly at random (a), it is possible to build specialized models for other services while freezing convolutional kernels (b). A relatively low loss rate is quickly reached for most services.

E. Training cost of new service models

DIAGNET combines a weighted attention mechanism with a non-overlapping convolutional kernel for generalization, and pooling layers for extensibility. 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.

Learning losses on training and validation sets are plotted in Fig. 8 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 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. We note that while the general model can be trained with a subset of services and landmarks, it can later be generalized to more landmarks and services with minimal re-training time.

V. RELATED WORK

Two measurement approaches exist for data collection in root cause analysis solutions: *passive observation* and *active probing*. Passive measurement relies on existing traffic and does not introduce any overhead and has therefore been previously used in large-scale systems [24], [32]. Depending on the setup, different sources of measurements are available such as local system or router insights [30], [33], [34], request path annotations [35], [36], routing monitors [37] or more recently from Internet background radiation [38]. Still, passive observations fall short in low-traffic environments with little information about the underlying network architecture and no control over routing paths [39]. 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 this method can also be used in conjunction with passive monitoring for bootstrapping a system [24], or testing hypotheses [40].

Numerous works have been performed in enterprise networks and datacenters, where the full network topology is known (e.g Clos-like topologies in Azure, or SDN-driven networks [22], [23]). This topology information allows network tomography techniques to be applied [16]–[21], [41], [42], pinpointing faulty links or components accurately at scale despite complex dependencies between components [43].

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 [1], [16], [20], [30], on invalid BGP announcements [33], [37] or Virtual Hard Drive failures in the recent DeepView work [43]. Netalyzr [7] and Fathom [8] collect end-user connectivity statistics to propose automated troubleshooting using predefined expert rules.

By contrast, our work keeps a generic approach and avoids

solutions exclusive to known topologies or specific connectivity rules. Nevertheless, the specific metrics extracted from the aforementioned methods are complementary to DIAGNET and could be used as additional input features if available, allowing for narrower and more accurate root cause diagnostics.

Regarding machine learning methods, belief networks have been heavily used in root cause analysis strategies [44]–[46] to model the complex dependencies between network components and online services. However, such methods require many approximations in the modeling and the solving to remain tractable, while being very sensitive to errors in topology identification [47], [48]. Random forest models are also known to perform well in understanding network failures, as demonstrated by NetPoirot [30]. To the best of our knowledge, DIAGNET is the first attempt to model a network with a variable number of input features (landmarks) using convolutional neural networks. DIAGNET relies on the good expressivity of non-linear models to learn features dependencies, and 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 [24], [46], [49], [50]. Distributed Hash Tables (DHT) have historically been used for that purpose [51], [52], 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 AND FUTURE WORK

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 crowd-sourced 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.

Our future research is now focusing on deploying and validating DIAGNET for a large set of real online services. There is still a lot of possible improvements in the neural network architecture, for instance by modifying non-linear operations or adding more convolutional or hidden layers. One avenue for future work would be the integration of historic data in landmark features. We are also working towards a decentralized version of DIAGNET to propose a more scalable and privacy-preserving solution, thanks to recent advances in federated learning and training over encrypted data.

REFERENCES

[1] S. Sundaresan, Y. Grunenberger, N. Feamster, D. Papagiannaki, D. Levin, and R. Teixeira, "WTF? Locating Performance Problems in Home Networks," *CoRR*, 2013. [Online]. Available: http://hdl.handle.net/1853/46991

[2] G. Dimopoulos, I. Leontiadis, P. Barlet-ros, K. Papagiannaki, and P. Steenkiste, "Identifying the Root Cause of Video Streaming Issues on Mobile Devices," in *CoNEXT*, 2015.

[3] D. Z. Joumblatt, J. Chandrashekar, B. Kevton, and R. Teixeira, "Predicting User Dissatisfaction with Internet Application Performance at End-Hosts," in *INFOCOM*, 2013.

[4] H. Nam, K. H. Kim, and H. Schulzrinne, "QoE matters more than QoS: Why people stop watching cat videos," in *INFOCOM*, 2016.

[5] K. Eaton. (2012) How One Second Could Cost Amazon $1.6 Billion In Sales. [Online]. Available: https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales

[6] M. Dischinger, M. Marcon, S. Guha, K. P. Gummadi, R. Mahajan, and S. Saroiu, "Glasnost : Enabling End Users to Detect Traffic Differentiation," in *Design*, 2010.

[7] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: Illuminating The Edge Network," in *IMC*, 2010.

[8] M. Dhawan, J. Samuel, R. Teixeira, C. Kreibich, M. Allman, N. Weaver, and V. Paxson, "Fathom: a browser-based network measurement platform," in *IMC*, 2012.

[9] S. Sundaresan, R. Teixeira, G. Tech, N. Feamster, A. Pescapè, and S. Crawford, "Broadband Internet Performance : A View From the Gateway," in *SIGCOMM*, 2011.

[10] Y. Jin, N. Duffield, A. Gerber, P. Haffner, S. Sen, and Z.-L. Zhang, "NEVERMIND, the problem is already fixed: proactively detecting and troubleshooting customer DSL problems," in *CoNEXT*, 2010.

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, 1989.

[12] M. Lin, Q. Chen, and S. Yan, "Network In Network," *CoRR*, 2013. [Online]. Available: https://arxiv.org/abs/1312.4400

[13] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *CoRR*, 2014. [Online]. Available: https://arxiv.org/abs/1312.6034

[14] Ookla. (2019) Speedtest Servers. [Online]. Available: https://www.speedtest.net/speedtest-servers

[15] D. N. da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the Gap Between QoS Metrics and Web QoE Using Above-the-fold Metrics," in *PAM*, 2018.

[16] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *Transactions on Networking*, vol. 10, no. 3, 2002.

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

[18] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "NetDiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data," in *CoNEXT*, 2007.

[19] Y. Zhao, Y. Chen, and D. Bindel, "Towards Unbiased End-to-End Network Diagnosis," *Transactions on Networking*, vol. 17, no. 6, 2009.

[20] B. Arzani, S. Ciraci, H. H. Liu, J. Padhye, B. T. Loo, G. Outhred, S. Design, and I. Nsdi, "007: Democratically Finding the Cause of Packet Drops," in *NSDI*, 2018.

[21] C. Tan, Z. Jin, C. Guo, T. Zhang, H. Wu, K. Deng, D. Bi, and D. Xiang, "NetBouncer : Active Device and Link Failure Localization in Data Center Networks," in *NSDI*, 2019.

[22] P. Tammana, C. Nagarajan, P. Mamillapalli, R. Kompella, and M. Lee, "Fault localization in large-scale network policy deployment," in *ICDCS*, 2018.

[23] Y. M. Ke, H. C. Hsiao, and T. H. J. Kim, "SDNProbe: Lightweight fault localization in the error-prone environment," in *ICDCS*, 2018.

[24] V. N. Padmanabhan, S. Ramabhadran, and J. Padhye, "NetProfiler: Profiling wide-area networks using peer cooperation," in *IPTPS*, 2005.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *TPAMI*, vol. 37, no. 9, 2015.

[26] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *SIGKDD*, 2016.

[27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *ICCV*, 2017.

[28] D. Madigan, A. E. Raftery, C. T. Volinsky, and J. Hoeting, "Bayesian Model Averaging," in *AAAI*, 1996.

[29] *tc-netem(8) Linux User's Manual*, November 2011.

[30] B. Arzani, S. Ciraci, B. T. Loo, A. Schuster, and G. Outhred, "Taking the Blame Game out of Data Centers Operations with NetPoirot," in *SIGCOMM*, 2016.

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

[32] R. N. Mysore, R. Mahajan, A. Vahdat, and G. Varghese, "Gestalt: Fast, Unified Fault Localization for Networked Systems," *ATC*, 2014.

[33] H. Yan, L. Breslau, Z. Ge, D. Massey, D. Pei, and J. Yates, "G-RCA: A generic root cause analysis platform for service quality management in large IP networks," in *CoNEXT*, 2010.

[34] A. Roy, H. Zeng, J. Bagga, and A. C. Snoeren, "Passive Realtime Datacenter Fault Detection and Localization," *NSDI*, 2017.

[35] M. Chen, A. Accardi, E. Kiciman, J. Lloyd, D. Patterson, A. Fox, and E. Brewer, "Path-based failure and evolution management," in *NSDI*, 2004.

[36] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, and I. Stoica, "X-trace: A pervasive network tracing framework," in *NSDI*, 2007.

[37] V. Giotsas, C. Dietzel, G. Smaragdakis, A. Feldmann, A. Berger, and E. Aben, "Detecting Peering Infrastructure Outages in the Wild," in *SIGCOMM*, 2017.

[38] A. Guillot, R. Fontugne, P. Winter, P. Merindol, A. King, A. Dainotti, and C. Pelsser, "Chocolatine: Outage Detection for Internet Background Radiation," in *TMA*, 2019.

[39] W. W. T. Fok, X. Luo, R. Mok, W. Li, Y. Liu, E. W. W. Chan, and R. K. C. Chang, "MonoScope: Automating network faults diagnosis based on active measurements," in *IM*, 2013.

[40] I. Rish, M. Brodie, N. Odintsova, and G. Grabarnik, "Real-time problem determination in distributed systems using active probing," in *NOMS*, 2004.

[41] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network Tomography: Recent Developments," *Statistical Science*, vol. 19, no. 3, 2004.

[42] D. Ghita, H. Nguyen, M. Kurant, K. Argyraki, and P. Thiran, "Netscope: Practical network loss tomography," in *INFOCOM*, 2010.

[43] Q. Zhang, C. Guo, Y. Dang, N. Swanson, X. Yang, R. Yao, M. Chintalapati, A. Krishnamurthy, T. Anderson, and G. Yu, "Deepview: Virtual Disk Failure Diagnosis and Pattern Detection for Azure," in *NSDI*, 2018.

[44] M. Steinder and A. Sethi, "End-to-end service failure diagnosis using belief networks," in *NOMS*, 2002.

[45] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," in *SIGCOMM*, 2007.

[46] E. Kiciman, D. Maltz, and J. C. Platt, "Fast Variational Inference for Large-scale Internet Diagnosis," in *NIPS*, 2008.

[47] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang, "Maximum likelihood network topology identification from edge-based unicast measurements," in *SIGMETRICS*, vol. 30, no. 1, 2002.

[48] Y. Huang, N. Feamster, and R. Teixeira, "Practical issues with using network tomography for fault diagnosis," *SIGCOMM CCR*, vol. 38, no. 5, 2008.

[49] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A Knowledge Plane for the Internet," in *SIGCOMM*, 2003.

[50] H. J. Wang, J. C. Platt, Y. Chen, R. Zhang, and Y.-M. Wang, "Automatic Misconfiguration Troubleshooting with PeerPressure," in *OSDI*, 2004.

[51] D. R. Choffnes, F. E. Bustamante, and Z. Ge, "Crowdsourcing service-level network event monitoring," in *SIGCOMM*, 2010.

[52] K. H. Kim, H. Nam, V. Singh, D. Song, and H. Schulzrinne, "DYSWIS: Crowdsourcing a home network diagnosis," in *ICCCN*, 2014.