Known and Unknown Event Detection in OTDR Traces by Deep Learning Networks

Antonino Maria Rizzo[1*], Luca Magri[1], Davide Rutigliano[1,2], Pietro Invernizzi[2], Enrico Sozio[2], Cesare Alippi[1,3], Stefano Binetti[2] and Giacomo Boracchi[1]

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, Milano, I-20133, Italy.
[2]Cisco Photonics, Cisco Systems, Via Molgora, 48/C, Vimercate, 20871, Italy.
[3]Department, Università della Svizzera Italiana, Via Buffi 13, Lugano, 6900, Switzerland.

*Corresponding author(s). E-mail(s): antoninomaria.rizzo@mail.polimi.it;
Contributing authors: luca.magri@polimi.it; davide.rutigliano@mail.polimi.it; pinverni@cisco.com; esozio@cisco.com; cesare.alippi@polimi.it; sbinetti@cisco.com; giacomo.boracchi@polimi.it;

Abstract

Optical fiber links are customarily monitored by Optical Time Domain Reflectometer (OTDR), an optoelectronic instrument that measures the scattered or reflected light along the fiber and returns a signal, namely the *OTDR trace*. OTDR traces are typically analyzed by experts in laboratories or by hand-crafted algorithms running in embedded systems to localize critical events occurring along the fiber. In this work, we address the problem of automatically detecting optical events in OTDR traces through a deep learning model that can be deployed in embedded systems. In particular, we take inspiration from Faster R-CNN and present the first 1D object-detection neural network for OTDR traces. Thanks to an ad-hoc preprocessing pipeline for OTDR traces, we can also identify *unknown events*, namely events that are not represented in training data but that might indicate rare and unforeseen situations that need to be reported. The resulting network brings several advantages with respect to existing solutions, as these typically classify fixed-size windows of OTDR traces, thus are less accurate in the localization. Moreover, existing solutions do not report events that cannot be safely associated to any label in the training set. Our

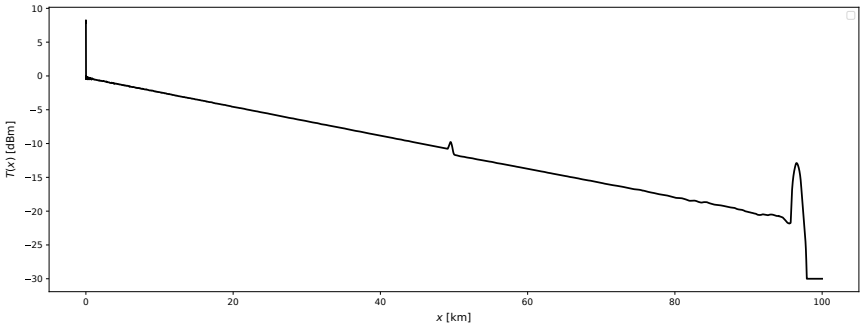


Fig. 1: An OTDR trace that is 100 km long and contains 3 events, a Face-Plate which appears as a downward slope on the left, a Pass-Through, which appears as a bump in the middle of the trace and a Fiber-Cut on the right in which we have an sudden decay of the signal and then noise.

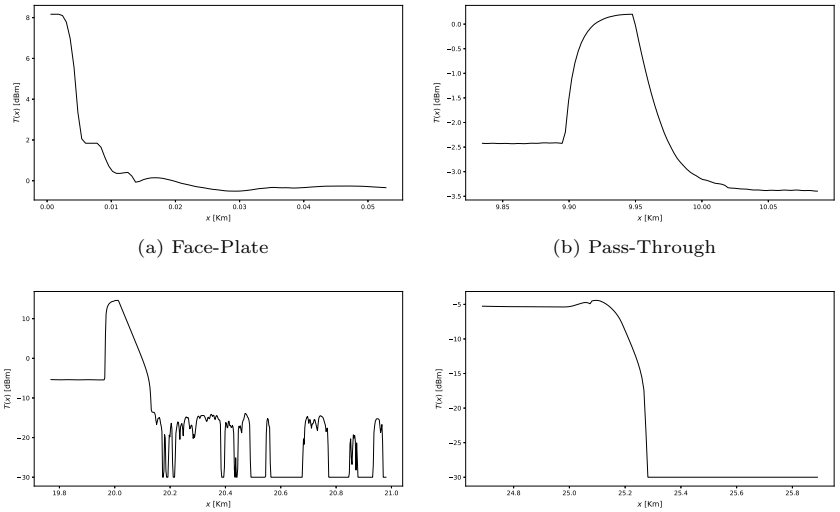
experiments, performed on real OTDR traces, show very promising performance, and can be directly executed on embedded OTDR devices.

Keywords: Detection Network, Open-Set, Recognition, OTDR Events, Time Series.

1 Introduction

Optical fiber links represent one of the major communication technologies, ranging from the backbone world-wide telecommunication systems to the last mile connections reaching our homes and apartments. Optical fibers run below our streets and in impervious areas like deserts, oceans or uninhabited lands. Problems along these links are not uncommon because of breakages, bad splicing or conjunctions that impair transmission quality when not entirely stopping the communication. Not surprisingly, monitoring optical spans and identifying faults is a primary activity for companies managing fiber links and optical equipment, and this has become even a more relevant issue since the introduction of coherent transmissions. In fact, optical-fiber monitoring enables a better characterization of the transmission performance, and a better allocation of optical channels.

A powerful and widely used instrument to test optical fiber links is the Optical Time Domain Reflectometer (OTDR) [1]. This device injects a series of optical pulses into the fiber and measures the light that is scattered or reflected back to the source into the *OTDR trace* (see Fig. 1). OTDR traces are characterized by a background signature, due to the attenuation of the signal because of fiber dispersion, and by multiple "event signatures". Optical events result from Rayleigh scattering and Fresnel reflections caused by physical devices connected to the fiber, bendings, knots or any other flaw along the fiber, and as such indicate potential issues on the fiber link.



(a) Face-Plate (b) Pass-Through

(c) Fiber-Cut (d) Fiber-End

Fig. 2: Instances of OTDR events windows in the training set.

Optical events are either analyzed by domain experts in laboratories, which can recognize patterns characterizing specific conditions (see Fig. 2) or by ad-hoc software running on embedded devices, which implements rather simplistic detection rules. These algorithms analyze fixed-size windows cropped from the OTDR traces, and can only recognize events belonging to the following macro-categories: "*reflection*", "*loss*", and "*dead zones*", but unfortunately, these categories do not cover the entire range of events that need to be routinely identified in most fiber links. Moreover, existing solutions are not able to identify events which can not be traced back to any known event category and, more in general, to categories that are not represented in the training set. Identifying unknown events is indeed a very important aspect as these might represent situations that need to be promptly reported.

Here we present the first deep *OTDR event detection network*, which solves many key problems of OTDR trace monitoring. First of all, our network analyzes an OTDR trace and detects an arbitrary number of events belonging to categories annotated in the training set. Second, each known event is accurately localized estimating its support, rather than a fixed-size window. Third, thanks to a preprocessing of the OTDR trace, we can assign an additional *unknown event* label to those events that cannot be traced back to annotated categories in the training set, thus for which no supervision is provided. To the best of our knowledge, unknown event detection has never been addressed before on OTDR traces. Finally, the proposed network can be trained in an end-to-end manner, and is very efficient at inference to enable deployment on Cisco routing platforms.

Our solution takes inspiration from Faster R-CNN [2] and performs both classification and localization of different types of events in an input trace, resulting in the first 1D event-detection Convolutional Neural Network (CNN) able to detect events in optic signals. The other major contribution, the detection of unknown events, comes from a preprocessing pipeline that first detect local peaks within the OTDR trace, and then classifies fixed-size windows clipped around them by employing an open-set classifier based on open-max [3]. These processing pipeline is only used to identify *unknown events regions*, which are then ignored by the detection network and separately reported. It is important to remark that, despite the object detection network includes a *no-event* (or background) class, this is primarily characterized by smooth decreasing patterns rather than events with peaks. Therefore, unknown events, which might possibly deserve even more attention than known events, are typically misclassified among the known categories [4].

Our experimental campaign, performed over a dataset of real OTDR traces acquired by *Cisco Photonics* from multiple fiber links, shows that the OTDR event detection network is very effective, achieving a mean average precision (mAP) score of 77.0%, representing an improvement in mAP of 26.67% with respect to automatic analysis software currently embedded in Cisco devices. Such high event detection performance is a consequence of a feature extraction network – representing the backbone of the event detection network – which has very accurate event classification capabilities. Indeed, our investigation demonstrates that these layers extract distinctive features for recognizing the event type, and that these can also be used to detect unknown events. Our extensive study in fact demonstrates that the backbone of the detection network can isolate unknown events synthetically introduced in the OTDR traces and that some events that were never shown during training can be detected as unknown events, thanks to a leave-one-class-out validation procedure. Overall, the proposed pipeline for isolating unknown events turn to be very effective in all the event-detection experiments, including those on real traces acquired from Cisco platforms. Moreover, the proposed OTDR event detection network is very efficient, and can run on Cisco routing platforms. Indeed, the event detection capabilities in the Cisco *NCS-1K* will be provided by an event detection network like the one described here.

This paper extends our conference publication [5] where we have presented the OTDR event detection network in conventional closed-set settings, namely as a network returning a label in a set of known event categories. The present manuscript presents an advanced solution to detect unknown events, thus operates in open-set conditions described in Section 3.3, together with an expanded experimental validation (Section 8). To enable unknown-event detection capabilities, we have revisited the entire preprocessing phase of OTDR traces, including an adaptive detrending of each trace based on robust fitting procedures to remove the background signature in each OTDR trace, as described in Section 6.1 and Section 6.2. The entire manuscript has been also expanded to

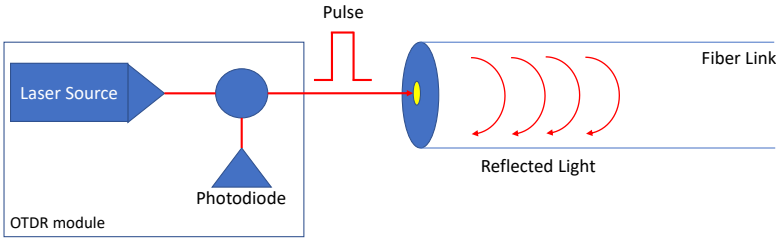


Fig. 3: The typical OTDR module include a laser who sends a pulse to the fiber link and a photodiode which gathers the backscattered and reflected signal.

provide a more comprehensive overview of the OTDR traces and of the related work as well as a more detailed description of the proposed solution.

2 Background on OTDR Traces

OTDR systems operate like one-dimensional radars inside optical fibers, and can accurately locate faults along a fiber link with a resolution up to a few centimeters. The underpinning principle is illustrated in Fig. 3: a short laser pulse is injected at one end of the fiber link and a photodiode at same location measures the backscattered and reflected signal. The OTDR trace accumulates thousands of measurements and represents the optical power (expressed in dB) as a function of the distance along the fiber (expressed in meters). An example of the resulting trace is reported in Fig. 1.

Many optical equipments, including those based on NCS2K and NCS1K platforms produced by Cisco [6, 7], embed the OTDR module to monitor failure conditions in different locations of the network. The analysis of OTDR traces allows the identification of a large number of events or failures that are primarily due to the following optical conditions, whose effects on the OTDR trace are illustrated in Fig. 4:

Rayleigh Back Scattering
: It represents the natural reflection of the fiber, which is due to impurities and in-homogeneous structures resulting from the fabrication process. Rayleigh Back Scattering contributes to the attenuation of optical power as a function of the fiber distance, resulting in a linear decay trend of OTDR traces, usually expressed in dB/km.

Fresnel Reflection
: It originates when the laser pulse hits a physical device like a connector, mechanical splice, bulk attenuator or fiber break, and mirrors back to the photodiode, resulting in a local increase of the reflected light.

Concentrated Losses
: These are due to attenuators, splices or splitters that reduce the optical power returned to the OTDR instrument and the OTDR trace attenuates very abruptly. Finally, when the fiber contains two nearby physical objects, the former might introduce reflection or scattering that masks the latter.

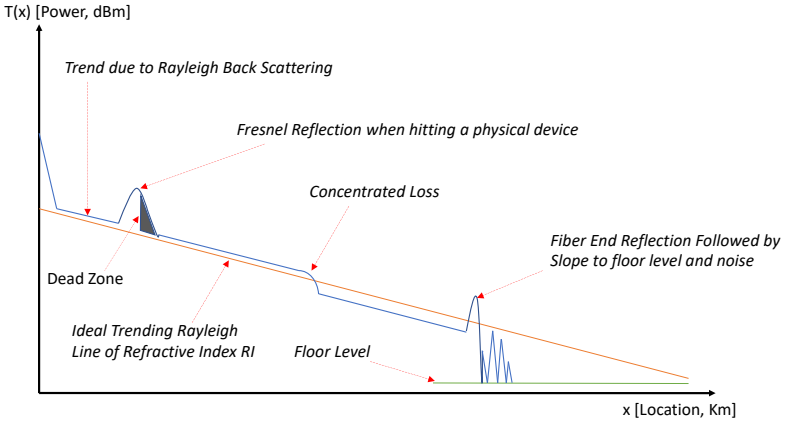


Fig. 4: Illustration of an OTDR trace with multiple events. The text annotations describe the root causes of these events.

Regions where the optical measurements suffer of such masking effect are named *dead zones*.

These optical conditions generates optical events which optical experts or specific softwares analyze to diagnose and monitor the fiber link conditions. In our study we consider four types of events:

- FACE-PLATE: this event originates at the beginning of the span where OTDR tool inject its pulses.
- PASS-THROUGH: this event occurs when two fiber links are connected with a mechanical conjunction.
- FIBER-END: this event is due to a connector that is plugged in a remote equipment to terminate the fiber link.
- FIBER-CUT: this event is due to a cut, which interrupts the fiber and therefore the transmission.

In practice, there might also appear *unknown events*, which cannot be traced back to any of the above categories. On top of these, there are few event types, such as the *Bulk Attenuator*, which are very rare, thus it is not possible to gather a sufficiently large number of training examples. Solutions to detect unknown events are very valuable as these can also report rare events that are not represented in the training set.

3 Related work

This section covers the machine learning literature that is most relevant to event detection in OTDR traces, including object detection networks (Section 3.1), deep neural networks for time series (Section 3.2), and open-set recognition (Section 3.3). We also present the existing solutions for automatically detecting optic events in OTDR traces (Section 3.4), discussing their limitations with respect to the proposed solution.

3.1 Object Detection Networks

Object-detection networks are trained to both localize and classify objects from known categories. These have been primarily investigated in the computer vision community to automatically locate objects in images. We mainly focus on networks based on the R-CNN architecture [2, 8, 9], which have inspired the design of the proposed OTDR event detection network. Detection networks belonging to the R-CNN family are composed of two blocks, where the first generates region proposals for the classification block.

The pioneering object detection network is R-CNN [8], which leverages an external region proposal algorithm that heuristically identifies image regions containing objects. These regions are then cropped from the image, reshaped and fed to a CNN, thus separately classified. Extensions over these models are the Fast R-CNN [9], which speeds up computation w.r.t. R-CNN by directly cropping selected regions from the network latent representations, and the Faster R-CNN [2] which in addition trains a Region Proposal Networks (RPN) to identify region proposals. The Faster R-CNN is very efficient and effective, since the RPN can be trained to maximize detection performance, and its high detection accuracy demonstrates that convolutional features preserve useful information to generate accurate region proposals. On top of these convolutional features, the RPN jointly predicts a correction term for the region proposal and infers an *objectness scores* at each location on a regular grid.

Other detection networks follows a single-shot approach, where the neural network is applied to the entire image to directly detect objects. The most notable example is YOLO [10] and its later versions like [11]. As mentioned in Section 1 we took inspiration from Faster R-CNN, since it is a simpler architecture than YOLO and we do not have strict timing constrains in OTDR monitoring. In fact, OTDR traces have to be analyzed much less frequently than video frames and, being $1D$ signals, are definitely less computationally demanding than images.

3.2 Deep Learning for Time Series

Given the success of CNN in solving visual recognition problems on images, it is not surprising that CNN for 1D inputs (1D CNNs) have reached state-of-the-art performance in many supervised problems concerning time-series. Here we primarily focus on time series detection, which is related to our work, and we do not cover the broad literature concerning time series forecasting.

Performing object detection in time series corresponds to localizing and classifying specific patterns or shapes in the input time series, which in case of OTDR traces are events. Detection problems have been much less investigated in time series domain rather than in images, and the only examples refer to the detection of anomalies in ECG tracings [12], earthquakes in seismic waves [13], and specific words in audio signals [14]. A recently proposed network, SpeechYOLO [15], detects and localizes specific keywords or speech objects in a fixed-length audio signal. Our solution shares the same rationale behind

SpeechYOLO, as we also adapt successful object-detection network designed for images to operate on time series, even though our OTDR detection network is meant for a different domain and takes inspiration from a different detection network. Moreover, our event-detection network can detect unknown events.

3.3 Open-Set Recognition

The vast majority of deep neural networks for classification work in closed-set settings, thus associate each input to a category belonging to the closed-set of labels provided during training. This is a too strict assumption in visual recognition problems, since at test time inputs might belong to unseen categories. Therefore, in Open-Set Recognition (OSR) [16] the network has the capability to spot inputs not belonging to the set of known categories and mark them as unknown. To this end, techniques like thresholding posterior probabilities provided by Softmax, or the use of OpenMax [3] have been proposed. Over the past few years, many alternatives to OpenMax have been proposed, which include some ad-hoc learning procedure [17, 18].

Open-Set recognition in object detection has been addressed only lately [4, 19, 20]. The reason might be that OSR is not straightforward to solve in detection networks, since all these networks include a *background* class that is expected to gather objects that do not belong to known categories. These networks are trained over large datasets of natural images, which typically include many unknown objects on top of the annotated (known) ones. Thus, all the unknown objects are provided to the network during training and implicitly associated to the background label [19]. As a matter of fact, there have been only a few works addressing object detection in more realistic settings where the model is expected to identify unknown objects, and these primarily revolve around the estimation of model uncertainty on each prediction, to then discard any low-confidence detection [21, 22].

3.4 Deep Learning for OTDR Trace Analysis

Deep learning models have been used to automatically analyze OTDR traces for distributed acoustic sensing, where a buried optical fiber link is used to sense external nearby events (e.g., a human walking or some seismic activity). Quite a few recent studies [23–25] demonstrate that deep learning has the potential to classify these type of events, which however have to be preliminary detected by an external algorithm/pipeline. In particular, [26] proposes a $1D$ neural network based on sequence learning, that takes as input a de-noised $1D$ signal and recognizes external intrusion events. In [27] the authors propose a classifier for OTDR traces based on $2D$ CNN that accurately classifies events among 5 categories such as walking or digging.

The proposed OTDR event detection network is however rather different from those mentioned above, both for the type of application and for the deep learning model employed. In terms of application, none of the above network concerns the detection of optic events, since [24] is meant for seismic data,

while others, like [27], deal with general events unrelated with the optic field. The major difference, however, concerns the type of network employed since all these methods formulate event recognition as a classification problem, while we jointly classify and localize events estimating the position and extent of the event along the OTDR trace. Most remarkably, none of the above solutions address the problem of detecting unknown events.

4 Problem Formulation

Here we provide a formal description to the problem of detecting known and unknown events in OTDR traces. We describe each OTDR trace as a time series of lenght n, which can be stored in a vector. For the sake of convenience, we re-sample the trace over a uniform grid, i.e.,

$$T = \{T(x_1), T(x_2), ..., T(x_n)\} \tag{1}$$

where $T(x_i) \in \mathbb{R}$ represents the reflection loss at the i-th position in the fiber, and $n \in \mathbb{N}$ is the length of the trace T. OTDR traces might contain a varying number of events. Each event is represented by triplet $e = (y, x_s, x_e)$ that corresponds to a portion of the OTDR trace $\{T(x_s), ..., T(x_e)\}$ between the starting point x_s and ending point x_e where the trace exhibits a pattern corresponding to an event of type y. We assume we are provided with a labeled training set $TR = \{(T_j\ ,\ E_j)\ ,\ j = 1, \ldots, N\}$, containing N traces. Each trace T_j in the training set is paired with a set of events $E_j = \{(y, x_s, x_e)_i\ ,\ i = 1, \ldots, M_j\}$. Due to the special nature of optic events in OTDR traces, the training set TR might not cover all the possible event types, thus the label set $Y = Y_K \cup Y_U$ is the union of known events Y_K, which are represented in TR, and unknown events Y_U, which might appear only during testing. In our experiments, we consider as Y_K the common events: $Y_K = \{$ FACE-PLATE, PASS-THROUGH, FIBER-END, FIBER-CUT $\}$ described in Section 2. Obviously $Y_K \cap Y_U = \emptyset$.

Our goal can be formulated as designing a neural network that can automatically detect all the events in an input trace T, estimating, for each event its location and label in Y_K. Moreover, whenever the trace T contains an unknown event – whose label belongs to Y_U – the model should detect it and return the *unknown event* label. The neural network will be trained over the training set TR and has to be sufficiently lightweight to be executed in an embedded OTDR device.

5 OTDR Event Detection Network

The architecture of the proposed OTDR event detection network is inspired by the R-CNN family, which have been modified to take as input $1D$ time series rather than images. The output of our network is a collection of detected events $\{(y, x_s, x_e)\}$. The major advantage of adopting a detection network is that events are not detected by analyzing a fixed window selected over the trace, but rather the network itself identifies the best window providing the

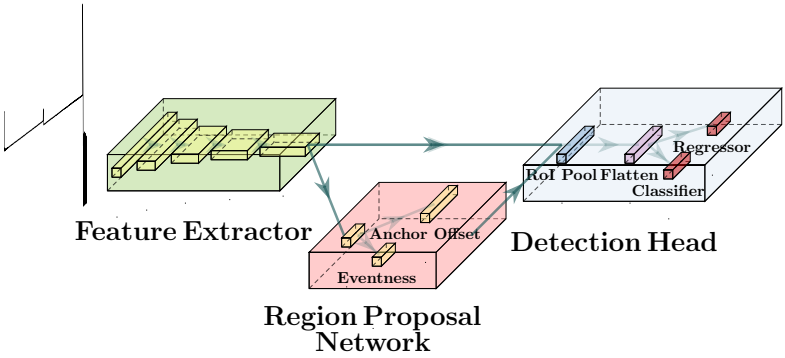


Fig. 5: Architecture of the proposed OTDR Event Detection Network.

largest evidence for each specific event. Our event-detection network has been already deployed onto Cisco NCS-1001 optical platform, offering an on-demand feature for automatic OTDR trace analysis. In this section, we describe the architecture of the network to detect events from known classes Y_K.

5.1 Network Architecture

Fig. 5 illustrates the proposed OTDR event detection network, which mimics the Faster R-CNN architecture [2]. Here we primarily discuss the major differences which are due to the fact that our detection model is a 1D CNN, and we refer the reader to [2] for additional details. The network has overall 103.108 parameters, which is much less than $2D$ object detection networks. The three major components of the OTDR event detection network are: *i*) the feature extraction network, *ii*) the region proposal network, and *iii*) the detection head.

Feature Extraction Network (FEN)

We took inspiration from the family of ResNet CNNs [28] to design our FEN. These have demonstrated to be effective in extracting informative features for image classification, thus we expect its 1D counterpart to be very effective as FEN. According to the training procedure in [2], the FEN is pretrained as a classification network, specifically we train our FEN to classify OTDR events in TR. Details concerning FEN training are reported in Section 7. The FEN is composed of 22 convolutional layers and has receptive field of 278 and effective stride of 8. As depicted in Fig. 6, the FEN is composed by a convolutional layer with 13 filters of size 1×3 followed by a Batch Normalization and a Max Pooling layer of size 1×2, then three pairs of *1D-Conv-Projection* (Fig. 7a) and *1D-Conv-Identity* blocks (Fig. 7b) consisting of 3 and 4 convolutional layers, respectively. Being a 1D CNN, the resulting FEN is very lightweight and, despite its overall 22 convolutional layers, it contains only 83.888 parameters.

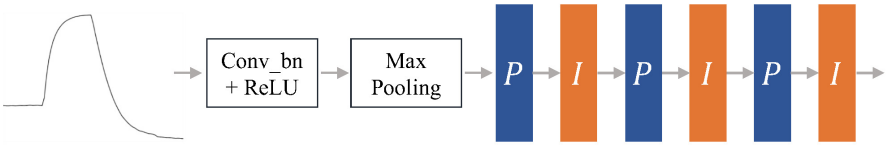
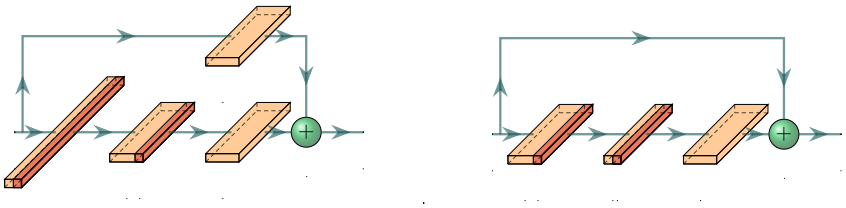


Fig. 6: Architecture of the Feature Extraction Network. The P blocks colored in blue are the 1D-Conv-Projection block, the I blocks colored orange are the 1D-Conv-Identity block.



(a) 1D-Conv-Projection block (b) 1D-Conv-Identity block

Fig. 7: Building blocks of our network architecture.

Region Proposal Network (RPN)

The RPN is a CNN introduced in [2] to preliminary estimate the location of objects to be detected in an input image. In our case, the RPN is trained to return candidate locations for events in an OTDR trace. The RPN takes as input the features extracted from the FEN and outputs a set of *region proposals*, which are defined upon *anchors*, namely predefined segments in the anchor set A, which play the role of the 2D bounding boxes in images [2].

As illustrated in Fig. 5, the RPN has two sibling output layers, and is applied in sliding window fashion over the feature maps extracted by the FEN. The two output layers return, per each spatial location and per each anchor in A, an estimate of region proposals. In practice, each region proposal is associated with an *eventness score* p and an *offset* t. The eventness score p encodes the confidence of having an event in that specific location over the region of the corresponding anchor. The offset t is meant to be applied to this region to match at best the region of a nearby annotated event. As in [2], the offset associated to the anchor $a \in A$ is described as $t = (t_{a,x},\ t_{a,w})$, where $t_{a,x}$ is a scale-invariant translation term, and $t_{a,w}$ is the log-space width scaling:

$$t_{a,x} = \frac{x - x_a}{w_a}; \qquad t_{a,w} = \log \frac{w}{w_a} \tag{2}$$

being x_a (x) the location of the corresponding anchor (estimated region), and w_a (w) the width of the corresponding anchor (estimated region). In our OTDR event detection network we use three different anchors thus, at each location on the feature maps provided by the FEN we get $|A| = 3$ estimates.

Detection Head

As illustrated in Fig. 5, the detection head takes as input the feature maps from the FEN and the region proposals from the RPN. The two are fed together to a RoI (Region of Interest) pooling layer, which is meant to return a fixed-length feature vector for each region proposal. Indeed, region proposals from different anchors have different spatial extent. The output of RoI pooling is fed to two sibling output layers: a *classification layer* and a *regression layer*. The classification layer consists in a fully connected layer followed by a Softmax, and returns – for each RoI – a vector of $|Y| + 1$ posterior probabilities, one per each known class in Y_K, plus one for the NO-EVENT class. The regression layer provides $|Y|$ outputs, each corresponding to the offset to adjust the event locations estimated by the RPN for a specific class. Thus, different refinements might be provided based on the event class. As anticipated in Section 4, the OTDR event detection network is trained to detect events of type FACE-PLATE, PASS-THROUGH, FIBER-END, and FIBER-CUT.

6 Known / Unknown Event Detection in OTDR Traces

In this section, we describe the procedure for detecting both known and unknown events in OTDR traces. The proposed OTDR event detection network is not able to report unknown events. The most straightforward solution of modifying the detection head to an OSR classifier (e.g., OpenMax) is not a viable option, since unknown events would be most probably associated to the *background* class, which is necessarily included in these type of networks. Therefore, we detect unknown events by a preliminary analysis of the trace to identify candidate regions.

The proposed solution is detailed in Algorithm 1. The first operation to be performed on an input OTDR trace T is detrending (line 2), as described in Section 6.1. OTDR trace detrending eases the detection of local maxima (line 3) by standard procedures that are based on comparisons over a fixed support for identifying the maximum, and a threshold that requires the peak to stand out a minimum height from its neighbors. Around each peak, we select a fixed-size window W containing 300 samples (thus $w = 150$ at line 5), which is then fed to an OSR classifier $\mathcal{C}$ (line 6), which is trained as described in Section 6.2. When the window W is classified as UNKNOWN (line 8), the detected unknown event is stored in E_U and accordingly reported. Any other label returned by $\mathcal{C}$ is ignored, since classification of known events over W is deemed less accurate than from the OTDR event detection network. After having analyzed all the peaks, the OTDR event detection network $\mathcal{E}$ is run on the original trace T (line 11). We have indeed experienced a superior event detection performance when training (and running) $\mathcal{E}$ over original traces T rather than detrended traces T', probably because the trend retains some useful information for event detection. In particular, certain events are more likely to be found at specific locations of the trace, for example, the FACE-PLATE is always at the beginning of the trace,

Algorithm 1 Known / Unknown Event Detection Network

Input: OTDR trace T, Event Detection Network $\mathcal{E}$, OSR classifier C
Output: E_K set of known events detected, E_U set of unknown events detected

1: $E_K = [],\ E_U = []$; ▷ Initialize the output sets
2: $T' = \text{Detrend}(T)$ ▷ using Algorithm 2.
3: Detect the set of local peaks P in the trace T'
4: **for each** $x_0 \in P$ **do** ▷ Test each candidate event using OSR Classifier C
5: Define the window $W = \{T'(x_0 - w + 1), \ldots, T'(x_0 + w)\}$
6: Classify W, $\hat{y} = \mathcal{C}(W)$
7: **if** $\hat{y} ==$ "UNKNOWN" **then**
8: $E_U = [E_U, (\text{"UNKNOWN"}, x_0 - w + 1, x_0 + w)]$
9: **end if**
10: **end for**
11: Detect events $E_K = \mathcal{E}(T)$. ▷ Event Detection Network
12: Remove from E_K any unknown event detected inside E_U.

while FIBER-END and FIBER-CUT are typically far from the beginning of the trace. In these cases, the network can identify this information when analyzing T values, but not from values of T'. Detections from $\mathcal{E}$ constitute the set E_K of known events, and we remove from this any event that is found within the support of an unknown event E_U (line 12). In what follows we provide more details concerning the adopted detrending procedure (Section 6.1), and the training of the OSR classifier (Section 6.2).

6.1 OTDR Trace Detrending

As illustrated in Section 2, each OTDR trace features a steady decrease of intensity, due to the natural attenuation of the impulse along the fiber. Such a trend depends on the specific type of optical fiber in place and, in our preliminary experiments, we have experienced that it can impair the detection of unknown events. To mitigate this problem, we present a detrending technique, detailed in Algorithm 2, to estimate and remove a tilt to compensate for possible linear trends affecting an input OTDR trace. We assume that the trends altering the OTDR are linear, thus trend estimation boils down to fit a line, described as the zero set of a first degree equation $T(x) = mx + q$, where $m, q \in \mathbb{R}$. The key is to employ a *robust* fitting method that prevents outliers, namely samples departing from the OTDR trend like events, to impair the trend estimate, as it would happen if a fragile least-square fitting was adopted instead. Specifically, we borrow from the Computer Vision literature a variant of LO-RanSaC [29]. LO-RanSaC follows a two-steps hypothesize-then-verify procedure. The hypothesize step is iterative: at each iteration k, we select a pair of points $(x_i, T(x_i))$ and $(x_j, T(x_j))$ (see Algorithm 2, line 3), to instantiate a line ℓ_k (line 4). The verification steps consist in keeping track of the line ℓ^* that best explains the data (lines 5 - 11) among all the lines sampled so far.

Algorithm 2 OTDR Trace Detrending by Lo-MSAC

Input: OTDR trace T, $\epsilon = 0.5$ dBm tolerance, $k_{\max} = 100$ max iteration
Output: T' detrended trace

1: $J^* = -\infty$, $k = 0$; ▷ Initialization
2: **while** $k < k_{\max}$ **do** ▷ Estimate linear trend
3: Select randomly two samples from the trace $(x_i, T(x_i))$,$(x_j, T(x_j))$
4: Compute the line ℓ_k passing by $(x_i, T(x_i))$,$(x_j, T(x_j))$
5: Evaluate $J(\ell_k)$, the score of the line ℓ_k as in (3)
6: **if** $J(\ell_k) > J^*$ **then** ▷ Update the best solution
7: Run optimization to fit ℓ_k over inliers
8: $\ell^* = \ell_k$
9: $J^* = J(\ell_k)$
10: **end if**
11: $k = k + 1$
12: **end while**
13: $T' = T - \ell^*$ as in (5) ▷ Detrend the trace

Lines are assessed by the score $J(\cdot)$ (line 3) defined as:

$$J(\ell) = \sum_{x_i} \rho\Big((x_i, T(x_i)), \ell\Big) \tag{3}$$

where $\rho(\cdot, \cdot)$ is a robust function that measures the distance between a line ℓ and a sample $(x_i, T(x_i))$ taking advantage of a user-specified tolerance ϵ that dichotomizes between inliers and outliers. Specifically, ρ is defined as follows:

$$\rho\Big((x_i, T(x_i)), \ell\Big) = \begin{cases} d\Big((x_i, T(x_i)), \ell\Big) & \text{if } d\Big((x_i, T(x_i)), \ell\Big) < \epsilon \\ \epsilon & \text{otherwise.} \end{cases} \tag{4}$$

where $d((x_i, T(x_i)), \ell) = |T(x_i) - mx_i - q|$ is the point-line distance along the vertical axis. The tolerance ϵ depends on the amount of noise in the data, and we set it to $\epsilon = 0.5$ in all our experiments since the noise is well below this value. In other words, when $(x_i, T(x_i))$ is an inlier for the line ℓ, $\rho(x_i, \ell)$ returns the distance between the line and the sample along the vertical direction. In contrast, when x_i lies outside the tolerance ϵ from the line ℓ, a constant penalty ϵ is returned. Since using only two points to fit a line lacks of statistical efficiency, the returned line parameters might be very noisy. Therefore, an additional optimization step is used to refine the parameters of ℓ on the inlier set every time a better line is detected (line 7). After iterating the process $m = 100$ times, the line ℓ^* yielding the minimum score is returned. The best line equation ℓ^* is hence used (line 13) to counter-tilt the trace T by subtracting the value of the line in correspondences of the samples of the signal:

$$T'(x) = T(x) - m^* x - q^*. \tag{5}$$

where m^* and q^* are the parameters of the line ℓ^*.

6.2 Open-Set Recognition of OTDR Events

We initially train the OSR classifier $\mathcal{C}$ as a traditional (i.e., *closed-set*) classifier over known events Y_K, and then modify its last layer to accommodate for unknown events according to open-set recognition solutions. In particular, the classifier $\mathcal{C}$ corresponds to the FEN of the OTDR event detection network and takes as input a fixed-window W containing $2w = 300$ samples. After the convolutional blocks, we introduce a Global Averaging Pooling (GAP) [30] and a Softmax layer at the network top to return estimates of class posterior.

The CNN is trained to classify windows W that are either selected around annotated events in the training set TR or randomly cropped from portions of the time series that do not include any event. These latter are associated to the *background* class. To train the classification network $\mathcal{C}$, we adopt the same data-augmentation procedure used for training the FEN (described in Section 7). However, here the entire training is performed over detrended traces T', since $\mathcal{C}$ operates in cascade to detrending. Once the CNN $\mathcal{C}$ has been trained, we modify the final layer to obtain an OSR classifier. More specifically, we have implemented two OSR baselines: the *Softmax*, a natural approach that assesses the confidence of the network in its prediction and *OpenMax* [3] an alternative that exploit demonstrates to achieve valuable results.

OSR by Softmax thresholding

The output of the Softmax activation can be interpreted as the confidence of the network in its prediction and is expected to be small when the network is uncertain. We can exploit this uncertainty by applying a threshold to the output of Softmax in order to classify as unknown the input samples for which the network is uncertain. In particular, the input is classified as UNKNOWN when the following condition is satisfied:

$$\max_i(\boldsymbol{\pi})_i < \tau \tag{6}$$

where $\boldsymbol{\pi} \in \mathbb{R}^{|Y_K|}$ is the probability computed by the classifier and τ is a threshold that has been tuned to reduce the number of false alarms (NFA) in the training set [31].

OSR by OpenMax

This method estimates the probability that an input is unknown by inspecting the distribution of the training activation vectors. In the following, we review the main steps, while for an exhaustive description, the reader is referred to [3]. OpenMax computes the activation vector $AV(W) \in \mathbb{R}^{|Y_K|}$ for each correctly classified training window $W \in \mathbb{R}^{2w}$, then, for each class y, OpenMax: i) groups all the activation vectors depending on their ground-truth class y, so to obtain

a set $A_y = \{AV(W) \mid \mathcal{C}(W) = y\}$; *ii*) averages the set A_y to obtain a mean activation vector $\mu_y \in \mathbb{R}^{|Y_K|}$; *iii*) computes the distance between the mean activation vector μ_y and all the activation vectors in A_y; *iv*) exploits Extreme Value Theory to fit a Weibull distribution to μ_y and the δ farthest activation vectors in A_y. At test time, given an input W, OpenMax uses the previously computed Weibull to recalibrate the input activation vector obtaining $\widehat{AV}(W)$. The difference between the updated activation vector $\widehat{AV}(W)$ and the original one $AV(W)$ denotes the uncertainty of the network prediction. Therefore, the unknown score U^{W} is computed as:

$$U^{\mathrm{W}} = \sum_{i=1}^{|Y_K|} AV_i(W) - \widehat{AV}_i(W) \tag{7}$$

Finally, OpenMax concatenates U^{W} to the recalibrated activation vector $\widehat{AV}(W)$ thus introducing an additional fictitious class UNKNOWN. The vector $\left[\widehat{AV}_i(W) \mid U^{\mathrm{W}}\right]$ is fed to a Softmax to get the recalibrated probabilities of the known classes and the probability of the unknown class. Therefore, when the probability of the class UNKNOWN is the largest, or the maximum probability is below a threshold τ, input W is classified as UNKNOWN.

7 OTDR Event Detection Network Training

The training procedure of the proposed OTDR Event Detection Network follows that of Faster R-CNN [2]. In particular, before training the entire network to perform event detection, we train the FEN to solve an auxiliary event-classification problem over a fixed input size. Such a preliminary training is meant to initialize the FEN to extract meaningful features, which enables training the RPN and accordingly the detection head. In particular, we adapt the "alternating training" procedure in [2] to train the entire OTDR network to detect events, according to the following four steps:

1. We freeze the weights of the pretrained FEN layers and fine-tune the RPN layers for solving the region proposal task. In particular, we minimize the loss in (9) which ignores event types.
2. We freeze the RPN and fine-tune both the FEN layers and the detection head on the proposals provided by the RPN. Here we minimize the event detection loss in (10), which considers both event location and labels.
3. We freeze the FEN layers and the detection head, and then fine-tune specifically the layers of the RPN, minimizing the loss in (9).
4. We freeze the FEN backbone and the RPN, and then fine-tune only the layers of the detection head to minimize the loss in (10).

Training FEN

To train the FEN we adopt the same CNN architecture and training procedures we used for the OSR classifier $\mathcal{C}$. In particular, we add a GAP layer as a

regularizer to achieve better translation invariance, then a Dense layer followed by a Softmax. We prepare a training set for event classification by cropping fixed-size windows W of $2w = 300$ points around events of the selected four types plus the no-event (these latter will concur to the background class). The event classification loss for a window W is the categorical cross-entropy over the known event types Y_K given by:

$$\mathcal{L}(y, \hat{y}) = -\sum_{k=1}^{|Y_K|} y_k \log(\hat{y}_k), \tag{8}$$

where y_k and $\hat{y}_k$ are the ground-truth and predicted one-hot encoded labels, respectively. Once trained, the GAP and Softmax layers are removed, and the trained layers represent the FEN backbone of the 1D-Faster R-CNN (see Fig. 5).

To cope with the severe class imbalance and the limited amount of annotated events, we adopt the following data-augmentation procedure during training. In each batch, we randomly select 10% of events and apply a right/left translation of the OTDR trace by a random amount between 5% and 25% of the original size. Furthermore, we randomly select 5% of the events in the batch and add a random offset within $[-8, +8]$ to shift the values of the time series. On top of these "standard" augmentation transformations, we adopt mix-up data augmentation [32], which has shown to be very beneficial when training deep CNNs for solving several tasks both in image and time series domains. Mix-up creates new training samples that are convex combinations of two randomly selected windows, which are associated to the very same linear combination of labels (one-hot encoded) as target. Therefore, mix-up extends the training distribution by including linear interpolation of training samples. Despite events generated by mix-up do not have a true physical meaning, we found experimentally this technique is very effective to improve generalization capabilities of our OTDR event detection network.

Training RPN

In contrast with the initialization of the FEN layers, the RPN is trained over OTDR traces to minimize a loss function which assesses how close the estimated anchors are to the annotated events, ignoring their class. Each anchor is associated to an *eventness score* which measure its membership to one of the known categories rather than no-event. When training the RPN, we minimize the following loss function over a mini-batch:

$$\mathcal{L}\left(\boldsymbol{y}, \hat{\boldsymbol{y}}, \boldsymbol{t}, \hat{\boldsymbol{t}}\right) = \sum_i \mathcal{L}_{cls}\left(y_i, \hat{y_i}\right) + \lambda \cdot \sum_i y_i \cdot \mathcal{L}_{reg}\left(t_i, \hat{t}_i\right). \tag{9}$$

where $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ denote the ground-truth and predicted eventness scores while $\boldsymbol{t}$ and $\hat{\boldsymbol{t}}$ gather all the ground-truth and predicted offsets. The event loss $\mathcal{L}_{cls}$ is the binary cross-entropy over event/no-event classes. The terms y_i and $\hat{y}_i$ denote the ground-truth and predicted eventness score of the $i-$th anchor. We

set $y_i = 1$ when the Intersection-over-Union (IoU) between the i-th anchor and at least one annotated event is higher than 0.5, otherwise we set $y_i = 0$. According to [2] we adopt as regression loss $\mathcal{L}_{reg}$ the $\mathcal{L}_1$ smoothed loss, namely $\mathcal{L}_{1,smooth}$, which is a variant of $\mathcal{L}_1$ loss function that is smoothed at the origin:

$$\mathcal{L}_{1,smooth}\left(t_i, \hat{t}_i\right) = \begin{cases} \frac{1}{2}\left(t_i - \hat{t}_i\right)^2, & \text{if } |t_i - \hat{t}_i| < 1 \\ |t_i - \hat{t}_i| - \frac{1}{2}, & \text{otherwise.} \end{cases} \tag{10}$$

The $\mathcal{L}_{1,smooth}$ loss is applied on each component of the predicted offset $\hat{t}_i$ and ground-truth offset t_i. Regression loss is multiplied by y_i because we want to assess localization errors only for positive anchors. The hyper-parameter λ balances the two terms of this multi-task loss function. To summarize, the loss in (9) combines the classification error for the eventness scores and a regression error for the estimated anchor offsets in the mini-batch.

Training Fast R-CNN

The multi-task loss computed on each RoI follows from [2] and is defined as:

$$\mathcal{L}\left(y_i, \hat{y}_i, t_i, \hat{t}_i\right) = \mathcal{L}_{cls}\left(y_i, \hat{y}_i\right) + \lambda \cdot \left[y_i \geq 1\right] \cdot \mathcal{L}_{reg}\left(t_i, \hat{t}_i\right) \tag{11}$$

where y_i, $\hat{y}_i$ denote the ground-truth and predicted class for each RoI, while t_i and $\hat{t}_i$ represent the offset of the ground-truth and predicted event with respect to the generic anchor. The classification loss $\mathcal{L}_{cls}$ consists in the categorical-cross entropy defined in (8). As in (9), the hyper-parameter λ balances the classification $\mathcal{L}_{cls}$ and regression $\mathcal{L}_{reg}$ terms of the multi-task loss. The term $[y_i \geq 1]$ evaluates to 1 when $y_i \geq 1$ and 0 otherwise, being $y_i = 0$ the no-event class. This latter factor is used to assess regression loss only at optic events. The regression loss $\mathcal{L}_{reg}$ is defined as in (9).

8 Experiments

In this section, we quantitatively assess the proposed OTDR event detection network to detect both known and unknown events. First, we describe the employed datasets (Section 8.1) and the figures of merit (Section 8.2). Then, we analyze the classification performance on both known and unknown events (Section 8.3) and the event-detection performance over both known and unknown events (Section 8.4).

Since there are no publicly available OTDR event-detection network that we can use in our experiments, we compare against the OTDR event detection algorithm running in Cisco devices NCS1001, NCS2K and NCS1010 [6, 7] and two OSR approaches, both for classification and event detection, the OpenMax and the Softmax (described in Sec. 6.2).

All our networks have been trained using Adam optimizer with learning rate 0.001, a batch size of 8 when pretraining the layers of the FEN, and 2

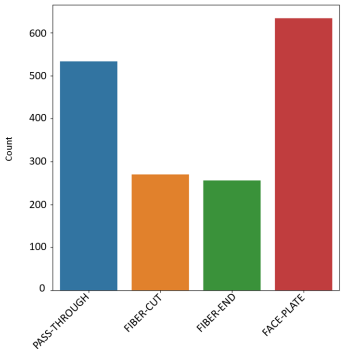


Fig. 8: Distribution of event types in the dataset.

when training the event-detection network, 200 epochs on each fold and set $\lambda = 5$ in (9) and (11). In addition, we use early-stopping criteria in all of our experiments to reduce the risk of overfitting.

Lastly, as regard the execution time, the inference only lasts 124.923 milliseconds on a *NVIDIA RTX A6000* running TensorFlow 2.6.2. We have also measured the inference time on the *NCS1001*, which is the embedded device where the deployed network runs. It is equipped with an *Intel Atom® C2718* CPU with 1.99 GHz and runs Tensorflow 1.15, On this platform, we obtain an average inference time of 5.347 seconds.

8.1 Datasets

The dataset of OTDR traces used to validate our method has been acquired in Cisco facilities over a long span of optical fiber where different devices were connected at different locations along the fiber link to generate optical events. OTDR recordings are stored in "SOR" file format [33], which includes – together with all the raw measurement – several information about the OTDR module and the tested link. However, to the purpose of event detection, we only consider the time series of raw measurements, which is paired with the location along the fiber where each measurement was acquired. As a preprocessing step we normalize all the power values of the trace to have intensity within $[0, 1]$. All the traces have been annotated by locating the initial and final points of each event, and labeled in Y_K by a specific annotation tool developed to this purpose. Overall, we have collected 628 traces with 1674 labeled events, excluding no-events. The distribution of events along the four classes is illustrated in Fig. 8.

Note that the above dataset does not contain any unknown event (either labeled or unlabeled). Therefore, to assess the detection performance over unknown events we gathered two OTDR traces from customers that contain a new event termed Bulk Attenuator, which are only used for testing purposes. In addition, to extensively assess the OSR capabilities of the classifier $\mathcal{C}$ we resort to 1D time series from a completely different domain, namely ECG

tracings. More specifically, we test our OSR classifier $\mathcal{C}$ over 633 heartbeats from a public dataset [34]. We select heartbeats as they also have a peak like OTDR events, but at the same time ECG tracings exhibit a very different shape than OTDR events, thus can be safely considered unknown events. We compensate for the intensity difference between heartbeats and OTDR traces by scaling each heartbeat to bring the average peak of all the 633 heartbeats equal to K-times the average peak over OTDR events. To investigate the OSR performance of $\mathcal{C}$ we consider different values $K \in \{0.125, 0.25, 0.5, 1, 2, 4, 8\}$. Values of $K >> 1$ result in very apparent changes, while $K \to 0$ makes changes impossible to perceive.

8.2 Figures of Merit

In our experiments we assess both event classification and event detection performance, over both known and unknown events. In event classification over fixed-size windows W, we report both the classification accuracy as a confusion matrix (including both known and unknown events), as well as standard figures of merit for each class, namely the accuracy, precision, recall, F_1 score and the area under the ROC curve (ROC-AUC). These metrics are also used in the OSR scenario and applied to unknown events as being all of the same class. Due to the relatively small size of the dataset, we evaluate the above figures of merit in a $K-$Fold Cross-Validation framework. We split our dataset in $K = 5$ different folds and performance are estimated over the union of all the test folds.

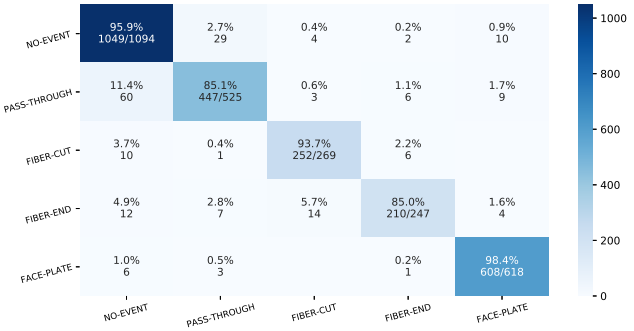
To assess the detection performance of the proposed model both for known and unknown events we resort to object-detection metrics used in the computer-vision literature, specifically the mean average precision (mAP) score, which is a global measure of classification and localization accuracy introduced in the PASCAL VOC challenge [35]. We also adopt the COCO [36] metric, denoted as mAP@[.5 : .05 : .95]: which evaluates mAP at 10 different intersection over union thresholds. Despite both metrics are specifically designed for $2D$ boxes, their adaptation to line segments is straightforward.

8.3 Event Classification

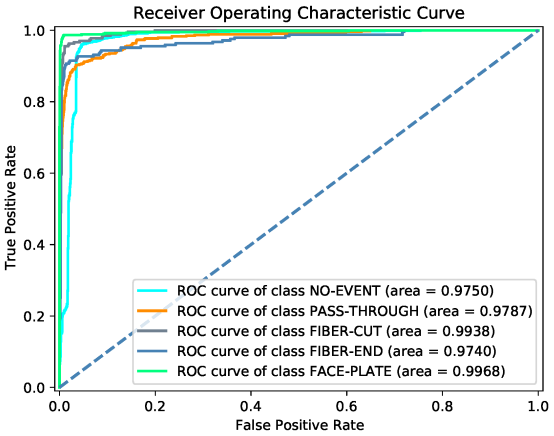
We assess the classification performance, on both known and unknown events, to test the network's capability to correctly recognize events on fixed-length windows. As regards unknown events classification, we compare Softmax thresholding with $\tau = 0.75$ and OpenMax with $\tau = 0.35$ and $\delta = 25$ on two experiments. In a first experiment, OTDR event classes are removed one at a time from the training set so that this class can be used for testing. In a second experiment, we directly injected ECG signals in the traces at test time.

Known Event Classification

We start by analyzing the classification results on known events. The confusion matrix in Fig. 9a shows that the layers of the FEN successfully identify specific events such as NO-EVENT and FACE-PLATE, while for the other classes we have



(a) Confusion Matrix



(b) ROC-AUC

Fig. 9: Classification performance of the known events. We report the confusion matrix in Fig. (a) and the ROC Curve in Fig. (b).

	Accuracy	Precision	Recall	F_1 Score	ROC-AUC
Pass-Through	0.851	0.918	0.851	0.883	0.979
Fiber-Cut	0.937	0.923	0.937	0.930	0.994
Fiber-End	0.850	0.933	0.850	0.890	0.974
Face-Plate	0.984	0.964	0.984	0.974	0.997
No-Event	0.959	0.923	0.959	0.940	0.975
Mean	0.916	0.932	0.916	0.923	0.984
Std. Dev.	0.062	0.018	0.062	0.037	0.011

Table 1: Classification performance computed by Cross-validation.

slightly worse performance. By visual inspection of the results, we discovered that often incorrectly classified pass-through events are very low bumbs, which

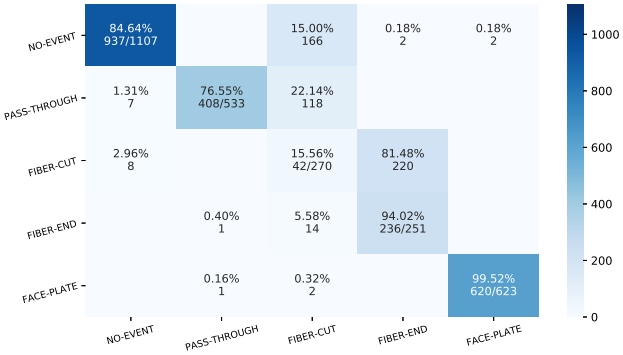
are confused with the noise, and therefore classified as no-event. The classification of fiber-cut and fiber-end is very challenging due to the class imbalance in our dataset, since we have roughly 250 examples for fiber-end and fiber-cut class, while at least 500 examples for the other classes (see Fig. 8). A summary of the results is shown in Table 1, our approach achieves, on average, more than 90% in terms of accuracy, precision, recall, F_1 Score and ROC-AUC. For the latter, the ROC curves of Fig. 9b confirm the discrimination capability of our model.

Unknown Event Classification with Leave-One-Class-Out

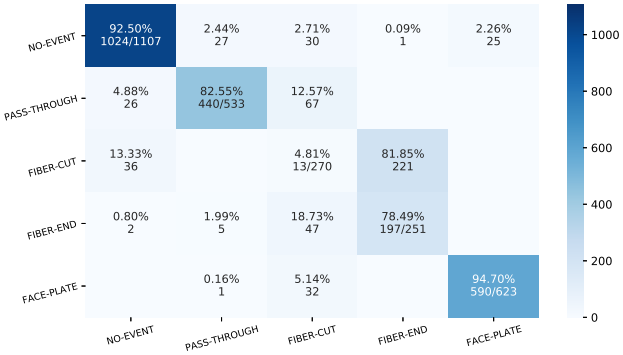
To assess the unknown event classification performance, we remove one at a time a known class from the training set and consider the corresponding OTDR events as unknown. At test time, we feed the network with a test-set containing also instances of the left out class and we expect these to be classified as unknown. Table 2 reports figures of merit relatively to the unknown class for both OpenMax and Softmax thresholding. Each row specifies a left-out class and shows the performance of the model when trained on the remaining ones. For all the classes except fiber-cut, OpenMax overcomes the Softmax thresholding. As shown in Fig 10, the performance on fiber-cut is very low because it is confused with fiber-end class. Indeed the two classes of events are very similar and we speculate that without supervision of both events, the network has not learned specific characteristics to discriminate them, which makes fiber-cut instances classified as fiber-end. In contrast, when we left out the fiber-end class, we have more 74% accuracy with OpenMax. This is because different features are learnt by the same network when trained using different data, therefore there is no guarantee of symmetry in the behavior of leaving out either the fiber-cut or the fiber-end.

Unknown Event Classification with ECG Tracings

To conclude the validation on unknown event classification we considered OTDR traces into which ECG signals have been injected. Fig. 11a illustrates the TPR of the two methods as K changes. As K increases, OpenMax detects more and more beats as unknown and converges to a TPR close to 1. Conversely, Softmax gradually worsens. This can be ascribed to the normalization of the scores carried out by Softmax. For large values of K, the ECG beats are very different from OTDR windows, therefore the network cannot extract any relevant feature. Therefore the scores fed to Softmax are all very low and with minimal differences. In this case, the normalization amplifies the small fluctuations between the scores bringing a random class to stand out. For $K = 1$ we analyse the case where the average peak in heartbeats is equal to the average peak over OTDR events. In this scenario we show in Fig. 11b the confusion matrix without any OSR mechanism where all heartbeat are classified as fiber-end. Fig. 11c shows that Softmax thresholding detects as unknown 57.5% of the heartbeats while 42.5% of them are classified as fiber-end. Fig. 11d shows that the best result is given by OpenMax which detects 73.14% of heartbeats as unknown events.



(a) Softmax thresholding



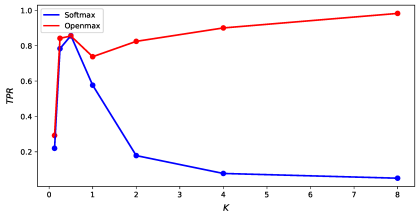
(b) OpenMax

Fig. 10: Fig. (a) shows the confusion matrix of the model embedded with the thresholding on the Softmax probabilities. Fig. (b) shows the confusion matrix of the model embedded with OpenMax.

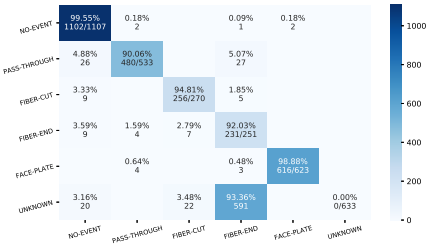
	Accuracy		Precision		Recall		F_1 Score		ROC-AUC	
	Soft	Open	Soft	Open	Soft	Open	Soft	Open	Soft	Open
Pass-Through	0.681	**0.758**	0.616	**0.655**	0.681	**0.758**	0.647	**0.703**	0.790	**0.897**
Fiber-Cut	**0.156**	0.048	**0.123**	0.068	**0.155**	0.048	**0.137**	0.056	**0.487**	0.232
Fiber-End	0.661	**0.745**	0.641	**0.702**	0.661	**0.741**	0.651	**0.721**	0.855	**0.865**
Face-Plate	0.299	**0.737**	0.439	**0.727**	0.299	**0.737**	0.355	**0.732**	0.594	**0.829**
Mean	0.449	**0.572**	0.455	**0.538**	0.4492	**0.571**	0.448	**0.553**	0.681	**0.706**

Table 2: Leave-One-Class-Out results. We remove an event type at a time from the dataset (the one specified in each row) and we train the network on the remaining data. Then we test using also instances of the left out event which has to be labelled as unknown. For each metric we compare Softmax and OpenMax.

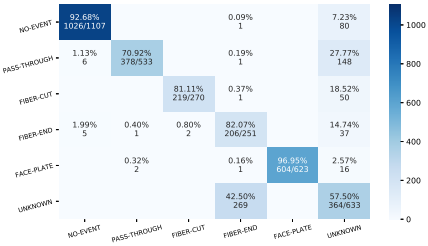
Finally, as depicted in Fig. 12, we verified that the distance of the activation vectors from the corresponding class follow a Weibull distribution. This is a fundamental assumption underpinning the OpenMax framework, and such a



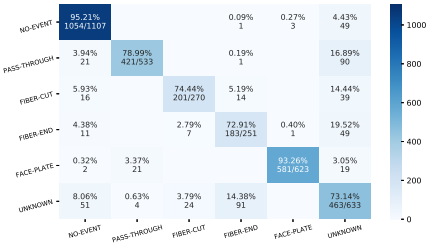
(a) Comparison between Softmax thresholding and OpenMax



(b) Base classifier



(c) Softmax thresholding



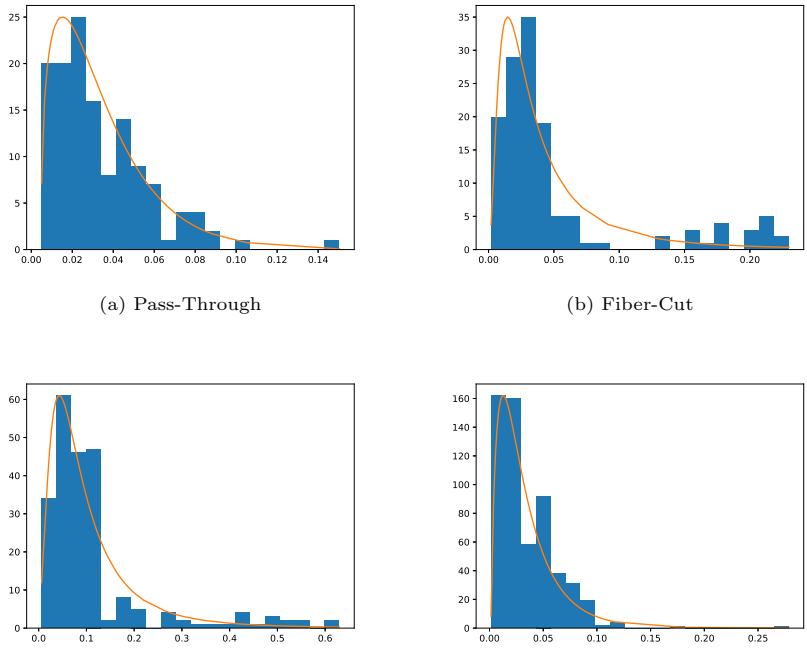
(d) OpenMax

Fig. 11: We assess the network on a test set of OTDR windows to which we have added ECG beat windows. Fig. (a) shows the True Positive Rate as the average intensity of the heartbeat changes. Figures (b), (c), and (d) show the confusion matrices in the case of CSR, with threshold on the Softmax probability and with OpenMax.

close match between histograms and the Weibull plot suggest this can be the reason for such a superior performance of OpenMax over Softmax.

8.4 Event Detection

The detection assessment, on both known and unknown events, is meant to test the network's capability to correctly localize events that span windows of different size over the trace and correctly classify them as one of the known classes or as unknown. As regards unknown detection, we exploit OpenMax with $\tau = 0.35$ and $\delta = 25$. In Sec. 8.4, we test the performance of our model on traces containing only known events, while in Sec. 8.4 we evaluate the OSR performance by directly injecting ECG beats in the traces and expecting the network to recover them as unknown. Finally, to qualitatively assess the detection on OTDR events, we evaluate the trace on traces containing a new event termed Bulk-Attenuator. Notice that if we had followed the same Leave-One-Class-Out strategy as for classification, we would have discarded all OTDR traces that contained at least one event of the unknown class. Since each trace contains from two to three classes of events, in varying numbers and distances, this strategy would have significantly reduced the dataset and prevented training.



(a) Pass-Through (b) Fiber-Cut

(c) Fiber-End (d) Face-Plate

Fig. 12: Distribution of distances of the activation vectors from the centroid of the corresponding class. As shown in [3], they follow a Weibull distribution.

	TP	FP	FN	AP@.5
Pass-Through	109	34	21	0.779
Fiber-Cut	41	25	3	0.840
Fiber-End	51	20	17	0.680
Face-Plate	120	18	18	0.789
mAP@.5	–	–	–	0.772
mAP@[.5:.05:.95]	–	–	–	0.445

Table 3: Performance of the OTDR detection network when tested against known events.

Event Type	NCS-1K	Ours
Reflective	0.25	**0.76**
Non-Reflective	0.77	**0.78**
End of Fiber	0.49	**0.76**
mAP	0.50	**0.77**

Table 4: NCS-1001 Comparison.

Known Event Detection

Our OTDR event detector achieves very good detection performance, achieving mAP@0.5 equal to 77.21% (PASCAL VOC) among all the classes and mAP@[.5:.05:.95] equal to 44.5% (COCO), see Table 3. We also compare our approach with existing solutions currently embedded in Cisco NCS-1001 (or shortly NCS-1K) for OTDR events detection. To enable a fair comparison,

	Base	OSR-DH	Ours
TE	**0.772**	0.767	0.744
TE + ECG	0.496	0.502	**0.751**

Table 5: Method comparisons overview in terms of mAP@0.5. *Base* denotes the trained OTDR detection network with no unknown detection mechanism. *OSR-DH* denotes the method in which OpenMax is embedded in the detection head classifier. *Ours* denotes the method described in Algorithm 1. *TE* denotes the test set with only known OTDR events. Finally *TE + ECG* denotes the dataset of OTDR traces where ECG heartbeats have been injected.

	TP		FP		FN		AP@.5	
	Ours	OSR-DH	Ours	OSR-DH	Ours	OSR-DH	Ours	OSR-DH
Pass-Through	**106**	104	47	**31**	**24**	26	0.707	**0.744**
Fiber-Cut	37	**40**	**24**	32	7	**4**	**0.790**	0.729
Fiber-End	**50**	47	**22**	133	**18**	21	**0.634**	0.273
Face-Plate	**120**	116	**18**	19	**18**	22	**0.789**	0.758
Unknown (ECG)	**106**	5	**39**	51	**15**	116	**0.835**	0.007
mAP@.5	–	–	–	–	–	–	**0.751**	0.502
mAP@[.5:.05:.95]	–	–	–	–	–	–	**0.414**	0.288

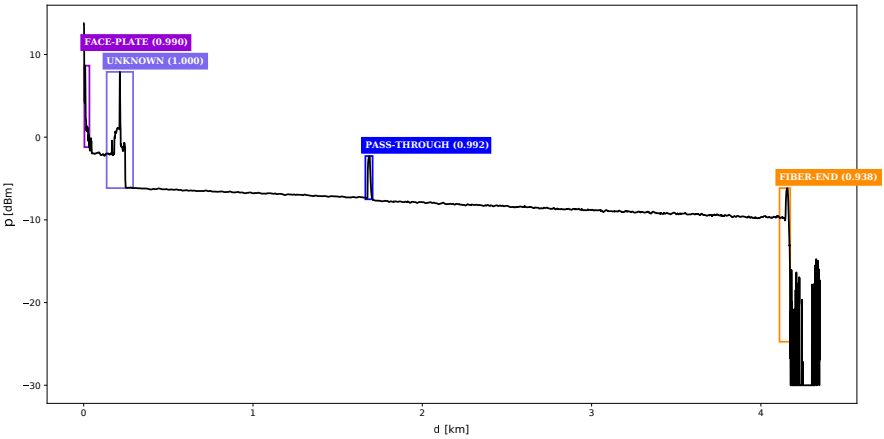
Table 6: Comparison between our method (see Algorithm 1) and the detection network with OpenMax embedded in the detection head. The two methods are tested against known OTDR events and unknown events (ECG hearbeats).

we have mapped predicted event types to the standard categories detected by existing solutions, which are fewer than those provided by the proposed OTDR detection network. These events are REFLECTIVE, NON-REFLECTIVE, and FIBER-END. Results in Table 4 show that the proposed detection network substantially outperforms existing solutions on NCS-1K devices, which implement hand-crafted detectors characterized by thresholds that have been tuned by optical experts, and that operate under strict assumptions on the event position and size. On top of a broad label set, the proposed OTDR event-detection network provides a more accurate localization since it does not process the trace on a fixed-size window basis.

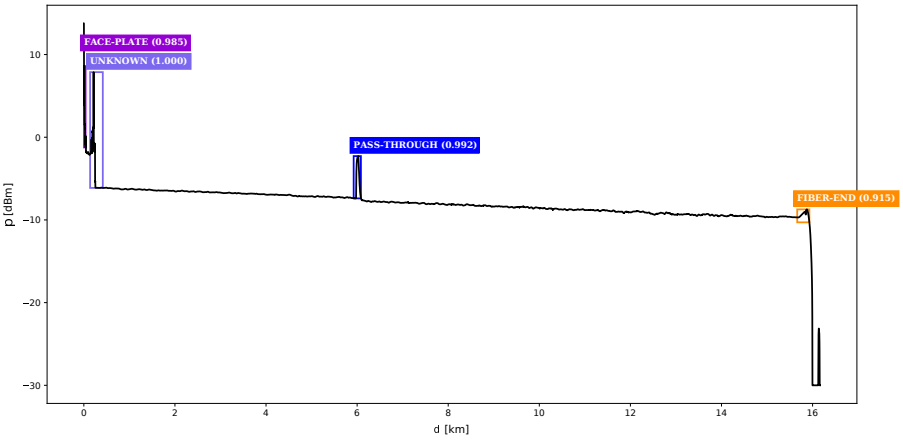
Unknown Event Detection

In order to assess the performance of our model for unknown event detection we considered OTDR traces with ECG signals. Table 5 collects the detection results on two test beds, one composed entirely by known OTDR events (TE) and a second one where ECG tracings have been added (TE + ECG). We compare the mAP@0.5 of our algorithm with a base detection network (Base) and a variant where OpenMax has been embedded in the detection head (OSR-DH).

Results demonstrates that the presence of ECG tracings in the test set reduces the mAP@0.5 to 49.61%. OSR-DH gets mAP@0.5 equal to 76.72%, which is only 1% less than the case without OpenMax. When tested against ECG tracings OSR-DH gets mAP@0.5 equal to 50.24%, which is only 1%



(a)



(b)

Fig. 13: During training, *Bulk-Attenuator* instances were not provided, so the network either ignores or incorrectly classifies these events. The OTDR traces depicted in (a) and (b) show that our algorithm is able to detect the *Bulk-Attenuator* instances as unknown events in addition to the known events.

improvement compared to Base. This happens because the detection network does not produce region proposals for unknown events and therefore the impact of OpenMax is minimal.

Our solution (proposed in Algorithm 1) achieves mAP@0.5 equal to 74.36% when tested against the known OTDR events and 75.10% when tested against the data set with ECG tracings. From Table 6 we see that we achieved a mAP@0.5 equal to 83.52% on unknown events. As a final observation, from Table 6 we see that OpenMax in the detection head has the advantage of reducing the number of false positives, as these will be detected as unknown.

The unknown detection performance are qualitatively assessed using two OTDR traces gathered from CISCO customers. These traces contains a new OTDR event termed Bulk-Attenuator. Fig. 13 shows that we are able to successfully detect the unknown event together with the known ones while the traditional network correctly detect only the known ones.

9 Conclusions

In this paper we present a deep learning model to detect events in OTDR traces. This is a very valuable alternative to existing solutions, which are based on expert-driven rules, thus are not flexible enough to identify different types of events along the optical fiber. We show that the proposed OTDR event detection network is not only able to recognize more event types than existing algorithms, but it is also more accurate in localizing them. Remarkably, we have presented a solution to report unknown events, including rare ones that do not appear in the training set. Our experiments show that the proposed approach can effectively solve the detection of optical events and can be used in a real-world environment, being employed in an embedded device and running in comparable time with respect to current industrial solutions (less than 6 seconds). As a future work, we plan to extend the set of event types and to remove the limitation of the fixed-size window for the classification of a given event.

10 Acknowledgments

This research was supported by research funds from *Cisco Photonics*.

11 Conflict of Interest

The authors state that they have no conflict of interest.

References

[1] Barnoski, M.K., Rourke, M.D., Jensen, S.M., Melville, R.T.: Optical time domain reflectometer. Applied Optics **16**(9), 2375–2379 (1977). https://doi.org/10.1364/AO.16.002375

[2] Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)

[3] Bendale, A., Boult, T.E.: Towards open set deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1563–1572 (2016)

[4] Dhamija, A., Gunther, M., Ventura, J., Boult, T.: The overlooked elephant of object detection: Open set. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 1021–1030 (2020)

[5] Rutigliano, D., Boracchi, G., Invernizzi, P., Sozio, E., Alippi, C., Binetti, S.: Event-detection deep neural network for otdr trace analysis. In: Iliadis, L., Macintyre, J., Jayne, C., Pimenidis, E. (eds.) Proceedings of the 22nd Engineering Applications of Neural Networks Conference, pp. 190–201. Springer, Cham (2021)

[6] Cisco: Cisco Network Convergence System 1001 OTDR Line Card Data Sheet. https://www.cisco.com/c/en/us/products/collateral/optical-networking/network-convergence-system-1000-series/datasheet-c78-742294.html. Accessed: 2022-01-28

[7] Cisco: Cisco Transport Node Controller and Transport Shelf Controller Cards Data Sheet - OTDR Functionality. https://www.cisco.com/c/en/us/products/collateral/optical-networking/ons-15454-series-multiservice-transport-platforms/datasheet-c78-602903.html#OTDRFunctionality. Accessed: 2022-01-28

[8] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)

[9] Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)

[10] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)

[11] Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271 (2017)

[12] Özal Yıldırım, Pławiak, P., Tan, R.-S., Acharya, U.R.: Arrhythmia detection using deep convolutional neural network with long duration ecg signals. Computers in Biology and Medicine **102**, 411–420 (2018)

[13] Wu, Y., Lin, Y., Zhou, Z., Bolton, D.C., Liu, J., Johnson, P.: Deepdetect: A cascaded region-based densely connected network for seismic event detection. IEEE Transactions on Geoscience and Remote Sensing **57**(1), 62–75 (2018)

[14] Palaz, D., Synnaeve, G., Collobert, R.: Jointly learning to locate and

classify words using convolutional networks. In: INTERSPEECH, pp. 2741–2745 (2016)

[15] Segal, Y., Fuchs, T.S., Keshet, J.: Speechyolo: Detection and localization of speech objects. Proc. Interspeech 2019, 4210–4214 (2019)

[16] Scheirer, W.J., de Rezende Rocha, A., Sapkota, A., Boult, T.E.: Toward open set recognition. IEEE transactions on pattern analysis and machine intelligence **35**(7), 1757–1772 (2012)

[17] Oza, P., Patel, V.M.: C2ae: Class conditioned auto-encoder for open-set recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2307–2316 (2019)

[18] Chen, G., Peng, P., Wang, X., Tian, Y.: Adversarial reciprocal points learning for open set recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1–1 (2021). https://doi.org/10.1109/TPAMI.2021.3106743

[19] Joseph, K., Khan, S., Khan, F.S., Balasubramanian, V.N.: Towards open world object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5830–5840 (2021)

[20] Saito, K., Hu, P., Darrell, T., Saenko, K.: Learning to detect every thing in an open world (2021) https://arxiv.org/abs/2112.01698

[21] Miller, D., Nicholson, L., Dayoub, F., Sünderhauf, N.: Dropout sampling for robust object detection in open-set conditions. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 3243–3249 (2018). IEEE

[22] Li, Y., Košecká, J.: Uncertainty aware proposal segmentation for unknown object detection. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 241–250 (2022)

[23] Aktas, M., Akgun, T., Demircin, M.U., Buyukaydin, D.: Deep learning based multi-threat classification for phase-OTDR fiber optic distributed acoustic sensing applications. In: Fiber Optic Sensors and Applications XIV, vol. 10208, p. 102080 (2017). International Society for Optics and Photonics

[24] Shiloh, L., Eyal, A., Giryes, R.: Deep learning approach for processing fiber-optic DAS seismic data. In: Optical Fiber Sensors, p. 22 (2018). Optical Society of America

[25] Liehr, S., Jäger, L.A., Karapanagiotis, C., Münzenberger, S., Kowarik, S.: Real-time dynamic strain sensing in optical fibers using artificial neural

networks. Optics express **27**(5), 7405–7425 (2019)

[26] Wu, H., Chen, J., Liu, X., Xiao, Y., Wang, M., Zheng, Y., Rao, Y.: One-dimensional CNN-based intelligent recognition of vibrations in pipeline monitoring with DAS. Journal of Lightwave Technology **37**(17), 4359–4366 (2019)

[27] Shi, Y., Wang, Y., Zhao, L., Fan, Z.: An event recognition method for ϕ-OTDR sensing system based on deep learning. Sensors **19**(15), 3421 (2019)

[28] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[29] Chum, O., Matas, J., Kittler, J.: Locally optimized ransac. In: Michaelis, B., Krell, G. (eds.) Pattern Recognition, pp. 236–243. Springer, Berlin, Heidelberg (2003)

[30] Lin, M., Chen, Q., Yan, S.: Network in network. In: 2nd International Conference on Learning Representations, ICLR (2014)

[31] Vaze, S., Han, K., Vedaldi, A., Zisserman, A.: Open-set recognition: A good closed-set classifier is all you need. In: International Conference on Learning Representations (2022). https://openreview.net/forum?id=5hLP5JY9S2d

[32] Zhang, H., Cissé, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: 6th International Conference on Learning Representations, ICLR (2018)

[33] Telcordia Technologies: Optical Time Domain Reflectometer (OTDR) Data Format, Sr-4731 edn. (2011). Telcordia Technologies

[34] Goldberger, A.L., Amaral, L.A., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.-K., Stanley, H.E.: Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. circulation **101**(23), 215–220 (2000)

[35] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge **88**(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4

[36] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: European Conference on Computer Vision, pp. 740–755 (2014). Springer