Event-Detection Deep Neural Network for OTDR Trace Analysis

Davide Rutigliano[⋆†], Giacomo Boracchi[⋆], Pietro Invernizzi[‡], Enrico Sozio[‡], Cesare Alippi[⋆§], Stefano Binetti[‡]

[⋆]Politecnico di Milano, Milano, Italy, [‡]Cisco Photonics, Vimercate, Italy, [§]Universitá della Svizzera Italiana, Lugano, Switzerland

Abstract. The Optical Time Domain Reflectometer (OTDR) is an optoelectronic instrument used to characterize an optical fiber using the measure of scattered or reflected light from points along the fiber. The resulting signal, namely the *OTDR trace*, is commonly used to identify and localize possible critical events in the fiber. In this work we address the problem of automatically detecting optical events in OTDR traces, and present the first 1D object-detection neural network for optical trace analysis. Our approach takes inspiration from a successful object detection network in images, the Faster R-CNN, which we adapt to time series domain. The proposed network can both classify and localize many optical events along an input trace. Our results show that the proposed solution is more accurate than existing software currently analyzing OTDR traces, improving the mean average precision score by 27.43%. In contrast with existing solutions that are not able to distinguish many types of events, our algorithm can be trained in an end-to-end manner to detect potentially any type of optic event. Moreover, our network has been deployed on embedded OTDR devices to be executed in real-time.

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 apartments. Optical fibers run below streets of our cities, but also 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 even stopping communication entirely. Promptly detecting problems along the fiber, namely localizing and possibly classifying them into known categories, is one of the primary concern of communication companies, as this translates in time and money saving. Moreover, optical-fiber monitoring enables a better characterization of the transmission performance, and a better allocation of optical channels. This latter has become a very important aspect since the introduction of coherent transmission and of mixed modulation schemes.

[†] D. Rutigliano is currently with Ericsson. This work was done when this author was an intern with Cisco.

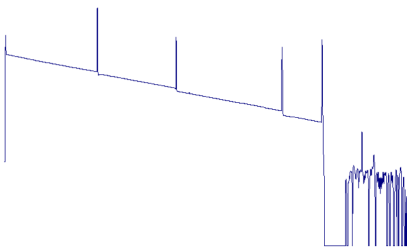

Fig. 1 (a): OTDR trace example

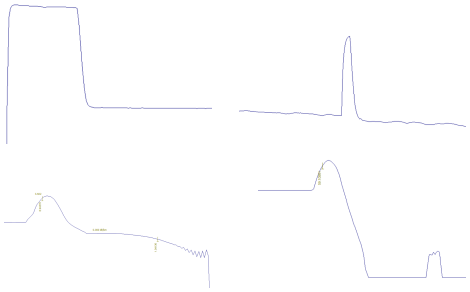

Fig. 1 (b): OTDR events examples: face-plate, pass-through, fiber-cut, fiber-end

A powerful and widely used instrument to test optical fiber links is the Optical Time Domain Reflectometer (OTDR) [1]. This instrument injects a series of optical pulses into the fiber and extracts the light that is scattered or reflected back to the source. Then, the reflection loss is measured and plotted as a function of the distance, resulting in the *OTDR trace* (see Fig. 1a). In OTDR traces, the background signature refers to the attenuation of the signal due to the fiber dispersion constant. On top of this, multiple "event signatures" might appear, resulting from Rayleigh scattering and Fresnel reflections caused by physical devices, bendings, knots or general flaws along the fiber.

Optical experts can visually recognize events along a trace by their specific patterns (see Fig. 1b). Moreover, there exist ad-hoc software that can automatically analyze OTDR traces acquired in laboratories. However, these solutions are rather simplistic, and are able to recognize only events belonging to two macro-categories: "*reflection*" and "*loss*". Here we address the problem of providing advanced event-detection capabilities to embedded devices that can be deployed in any network element along the fiber link and that – due to their minimal electronic – acquire traces characterized by higher noise levels than laboratory setup.

In particular, we analyze OTDR traces and automatically detect events by means of a deep learning model that can be trained in an end-to-end manner. The proposed model takes as input an optical trace and localize an arbitrary number of optic events belonging to different categories, easing the root cause analysis. Our solution takes inspiration from Faster R-CNN [2] and performs both classification and localization of different types of events in an input trace. This results in the first 1D event-detection Convolutional Neural Network (CNN) able to detect events in optic signals. The proposed network is very effective over our dataset, achieving a mean average precision (mAP) score of 88% and an improvement in mAP of 27.43% with respect to automatic analysis software currently embedded in Cisco devices. Our model can be easily re-trained to detect more event categories than those considered in this work. Finally, the proposed network is very efficient and has recently been deployed on Cisco routing platforms.

2 Related Work

Here we briefly introduce deep learning networks meant for object detection in images, which inspired our solution. Then, we briefly survey deep learning networks designed to analyze time series and OTDR traces.

2.1 Object Detection Networks for Images

Object-detection networks, namely models that are jointly trained to localize and classify objects in an input image, have been widely investigated in the computer vision community. Pioneering solutions like R-CNN [3] and Fast R-CNN [4] leverage an external region proposal algorithm, that typically implement some heuristic to preliminary identify regions containing objects. Each region is then classified separately. Faster R-CNN [2] extracts region proposal in an efficient and elegant way, by introducing Region Proposal Networks (RPN) which share the same convolutional layers of a Fast R-CNN detection network [4]. The effectiveness of this model demonstrates that convolutional feature maps used by region-based detectors can generate accurate region proposals. On top of these convolutional features, the RPN jointly performs regression over region bounds and classification over *objectness scores* at each location on a regular grid.
Detection systems based on R-CNN are composed of two blocks, the first is applied at multiple locations and scales on the input image and generates region proposals for the detection algorithm, which represents the second block. YOLO [5] pursues a different approach where a single neural network is applied to the entire image. This makes the inference extremely fast – till processing video frames in real-time – at a cost of a slight decrease the detection accuracy with respect to Faster R-CNN. Later versions of YOLO are the de-facto standard for object detection problems in computer vision.
As mentioned in Section 1 we took inspiration from Faster R-CNN, because this is a simpler architecture than YOLO and we do not have strict timing constrains during execution. In fact, OTDR signals have to be analyzed much less frequently than video frames and, moreover, processing $1D$ signals is definitely less computationally demanding than images.

2.2 Deep Learning for Time Series

Given the success of CNN in image classification, it is natural to think that CNN can also discover patterns in 1D signals. Not surprisingly, Deep Neural Networks have also reached state-of-the-art performance in many fields of time-series processing. Recent models perform classification over audio recordings, electroencephalogram (EEG) or electrocardiogram (ECG) as time series.
Time series classification is a challenging problem in data mining, and a plethora of algorithms have been proposed, thanks to the increasing availability of time series data. Recently, a few efforts have been devoted to exploit deep neural networks for end-to-end time series classification. ConvTimeNet [6] demonstrates that fully convolutional neural networks achieve great performance, even without

using pooling layers to reduce the input dimensionality. More recently, InceptionTimeNet [7] has shown that deeper CNN models coupled with residual connections can further improve classification, reaching state-of-the-art performance in time series classification. A review of most recent deep learning approaches to time series classification is given in [8, 9].
The counterpart of object detection in time series corresponds to localizing and classifying specific patterns or shapes in the input time series, which in our case are referred as "events". This problem has been however much less investigated and the only examples refer to anomaly detection in heartbeats / ECG tracings [10], earthquake detection in seismic waves [11], specific spoken words in audio signals [12].

2.3 Deep Learning for OTDR Trace Analysis

Recently, deep learning models have been successfully applied to OTDR trace analysis. A few studies [13, 14, 15], focus on event recognition in distributed optical fiber sensors by means of neural networks, and demonstrate the high potential of deep learning for OTDR systems. In particular, [16] proposes a 1D neural network based on sequence learning, that takes as input a de-noised $1D$ signal and recognizes external intrusion events. In [17] the authors propose a classifier for OTDR traces based on $2D$ CNN that accurately classifies events among 5 categories such as walking or digging. Even if some of these methods focus on recognition of events from OTDR signals, their purpose is rather different from the one considered in our paper. For instance, [14] applies a deep neural network for processing seismic data, while others like [17] takes into account event categories that are quite general and not directly related with the optic field. Furthermore, despite the similar application context, all the methods presented above formulate events recognition as a classification problem, while our purpose is to jointly classify and localize events, estimating also the location and extent of the event along the OTDR trace.

3 Problem Formulation

In this section we provide a formal description of the problem of detecting optic events in OTDR traces. Each OTDR trace can be described as a vector[1] $T = \{x_1, x_2, ..., x_n\}$ where $x_i \in \mathbb{R}$ represents the reflection loss at the i-th position in the fiber, and $n \in \mathbb{N}$ is the lenght of the trace. Each OTDR trace T might contain a different number of events, and each event is represented by triplet $e = (y, start, end)$ that corresponds to a portion $\{x_{start}, ..., x_{end}\}$ of the OTDR trace exhibiting a pattern associated to a known class $y \in Y$, where Y is the set of event types. In our experiments we consider the most common events: $Y = \{$ face-plate, pass-through, fiber-end, fiber-cut $\}$.

[1] We assume that the necessary processing and re-sampling of the acquired time series has been already performed.

Our goal is to design a model able to automatically detect each event in an input trace T, estimating its location and label. This model has to be sufficiently lightweight to be executed in an embedded OTDR device. We assume a labeled training set $D = \{(T_j\,,\,E_j)\,,\,j = 1,\ldots,N\}$, containing N traces is provided, where $E_j = \{(y_e, start_e, end_e)\,,\,e = 1,\ldots,M_j\}$ is the set of M_j annotated events over the trace T_j.

4 Deep Detection Network for OTDR Traces

Models from the R-CNN family take as input a $2D$ matrix of pixels, and provide as output pairs of predicted classes and *bounding boxes* encoded as the top left corner of the box and its height and width. In our case, the input is a time series represented by a $1D$ vector of equally spaced points, while the output consists of pairs of estimated classes y and line segments that are encoded by two coordinates $(start_e, end_e)$, indicating the starting and ending point of the detected event. The event-detection network for OTDR traces is illustrated in Fig. 2a and reproduces the Faster R-CNN architecture [2]. In particular, the overall detection model is a 1D neural network composed of three major components: *i*) the feature extraction network, *ii*) the RPN, the region proposal network, and *iii*) the detection head.

4.1 Architecture of the proposed Model

Feature Extraction Network. The architecture of the feature extractor network (FEN) was inspired by the family of ResNet [18]. After a few $1D$ convolutional layers, there are four 1D-residual blocks having 7 convolutional layers with a kernel with size 1×13 (see Fig. 2b). The overall architecture has been designed to maximize detection performance and is composed of 22 convolutional layers, featuring a receptive field of 278, stride of 8, resulting in 83.888 parameters.

Region Proposal Network. The RPN is a CNN that takes as input the feature maps from the FEN and outputs a set of *region proposals*. We obtain a region proposal for each spatial location in the input feature map and per each anchor. In our case, anchors are a fixed set of 1D segments (instead of 2D boxes as in [2]) with relatively small scales. This choice results in faster converge and better results, even though our experiments confirm that this parameter does not heavily impact the learning process. In our settings we use 3 different scales, yielding $A = 3$ anchors at each location on the feature maps provided by the FEN. Each region proposal is associated with an *eventness score* p, that encodes the confidence of having an event in that specific location, and an *offset* t, that is meant to be applied to the corresponding anchor placed in that specific location to best match an event annotation. As in [2], offsets $t = (t_x,\, t_w)$ are described by two terms: the scale-invariant translation t_x and the log-space width scaling relative to an anchor a:

$$t_x = \frac{x - x_a}{w_a}; \quad t_w = \log \frac{w}{w_a} \tag{1}$$

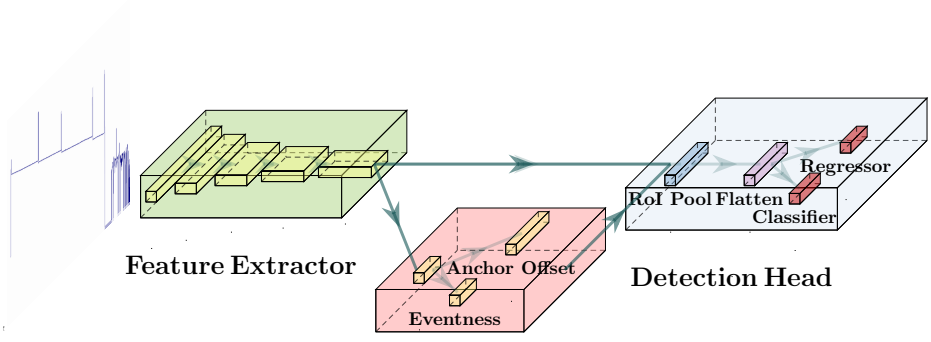

Fig. 2 (a): Architecture of the proposed OTDR Event Detection Network

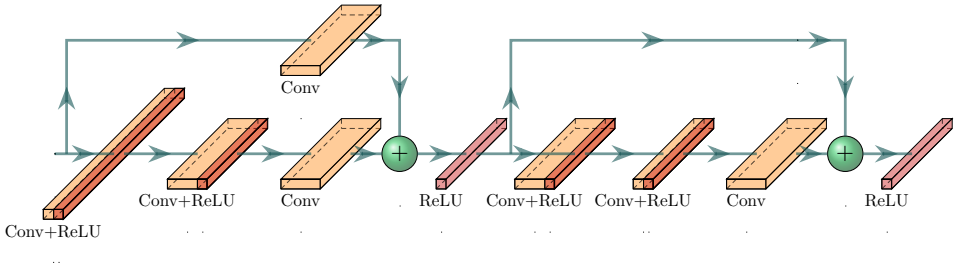

Fig. 2 (b): Architecture of Convolutional Blocks in the Feature Extraction Network

being x_a (x) the location of the corresponding anchor (estimated segment), and w_a (w) the width of the corresponding anchor (estimated segment). In practice RPN is applied in sliding window fashion and the region proposals are provided by two sibling output layers (see Fig. 2a).

Detection Head. As illustrated in Fig. 2a, the detection head takes as input the feature maps from the FEN and event locations provided by the RPN, which are fed together to a RoI (Region of Interest) pooling layer. RoI pooling layer returns a fixed-length feature vector from each region proposal, which are then fed to the two sibling output layers. The former is a classification layer that is made by a fully connected layer followed by a softmax. This returns – for each RoI – a collection of classification posterior probabilities over $|Y| + 1$ classes, which is augmented by the "no-event" class as required by detection networks. The latter is made of $|Y|$ regressors that provide as output the offsets (with respect to the same anchors used in the RPN) for each of the $|Y|$ event classes. These offsets further refine the location of each event provided by the RPN, also taking into account its type. Further details on the network architecture are described in [2].

4.2 Training

Before training the entire OTDR detection network, we preliminary train the convolutional layers of the FEN by formulating an auxiliary classification problem (see Section 5). This is a required step to train the RPN and the detection head which will be fed with the feature maps extracted from the FEN. Then, we pursue an "alternating training" following the scheme in [2], to train the whole OTDR detection network. The alternating training consists in fours steps:

1. We fix the weights of the pre-trained FEN backbone and fine-tune the RPN layers for solving the region proposal task.
2. We train only the detection head on the proposals provided by the RPN.
3. We fix the weights of the FEN backbone, and fine-tune specifically the layers of the RPN, using the entire network as a detector.
4. Finally, we fine tune only the detection-head layers, keeping all the weights fixed in both RPN and FEN.

Training FEN. As suggested in [4], we take the convolutional layers of the FEN and add a Global Averaging Pooling (GAP) [19] and a Softmax layers at the network top. This modification allows us to preliminary train the feature extraction layers for an event classification task over slices extracted from OTDR traces. Each slice is cropped over a fixed-sized window of 300 points and labeled with its corresponding type (including "no-event"). Once trained, the GAP and softmax layers are removed, and the trained layers are used to initialize the 1D-Faster R-CNN. Features extracted by these layers are then fed to both the RPN and detection head (see Fig. 2a).

Training RPN. RPN training develops similar to [2] and promotes estimated anchors to match the locations of annotated events, ignoring event labels. In order to define the ground truth for the eventness score returned by the RPN, we consider the $i-$th anchor as positive match when it has an IoU higher than 0.5 with the support of at least one annotated event (in this case it is associated to $p_i^* = 1$). Anchors having IoU smaller than 0.5 with all the support of the annotated events are deemed as negative ($p_i^* = 0$). The loss function for training the RPN over a mini-batch combines the classification error for the eventness scores and a regression error for the estimated anchor offsets:

$$\mathcal{L}\left(\boldsymbol{p}, \boldsymbol{t}\right) = \sum_i \mathcal{L}_{cls}\left(p_i, p_i^*\right) + \lambda \cdot \sum_i p_i^* \cdot \mathcal{L}_{reg}\left(t_i, t_i^*\right). \tag{2}$$

Where $\boldsymbol{p}$ and $\boldsymbol{t}$ are the collections of eventness score and offset relative to their corresponding anchors, respectively. The summation is intended over a mini-batch of region proposals. The event loss $\mathcal{L}_{cls}$ is the binary cross-entropy over event/no-event classes (with targets p_i^* defined above). The offset regression loss $\mathcal{L}_{reg}$ is based the smoothed $\mathcal{L}_1$ loss [2], namely $\mathcal{L}_{1,smooth}$, which is a variant of $\mathcal{L}_1$ loss function that is smoothed at the origin:

$$\mathcal{L}_{1,smooth}\left(x\right) = \begin{cases} \frac{1}{2}\, x^2, & \text{if } |x| < 1 \\ |x| - \frac{1}{2}, & \text{otherwise.} \end{cases} \tag{3}$$

Regression loss compares the estimated (t) and ground-truth (t^*) offset relative to the same anchor defined as in (1). Regression loss is multiplied by p_i^* because we want to assess localization errors only for positive anchors. The hyperparameter λ balances the two terms of this multi-task loss function.

Training Fast R-CNN. The multi-task loss computed on each RoI follows from [2] and is defined as:

$$\mathcal{L}\left(y_i, y_i^*, t_i, t_i^*\right) = \mathcal{L}_{cls}\left(y_i, y_i^*\right) + \lambda \cdot \left[y_i^* \geq 1\right] \cdot \mathcal{L}_{reg}\left(t_i, t_i^*\right) \tag{4}$$

where y_i (y_i^*) denotes the class prediction (ground truth) for each RoI, while t_i (t_i^*) is the offset parameterized as in (1) and represent the shift of the predicted (true) event with respect to a generic anchor. As in (2), 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 classification loss $\mathcal{L}_{cls}$ simply consists in the categorical-cross entropy for the predicted event types, while $\mathcal{L}_{reg}$ is defined as for the RPN.

5 Experiments

In this section we first describe the dataset preparation procedures and the employed figures of merit, then illustrate the experimental results for both event detection and for the auxiliary classification task to train the FEN. This latter indicates how good features are at distinguishing optical events.

5.1 Dataset of OTDR Traces

We first define a range of real fiber span setups, which include different event's types placed in different locations of the fiber link. OTDR recordings are stored in "SOR" file format [20], which includes – together with all the raw measurement – several information about the OTDR module and the tested link. To obtain the OTDR traces we extract only the raw measurements and the location along the fiber for each measurement. As a pre-processing 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, see Section 3. A specific annotation tool has been developed for this purpose. Overall, we have collected 628 traces with 1674 labeled events (excluding no events). Even if this dataset have much fewer examples than image classification test-beds, our event-detection network that is designed for $1D$ signals has overall 103.108 parameters, which is much less than $2D$ object detection networks.

5.2 Experimental Setup

Due to the relatively small size of the dataset, both approaches have been evaluated using $K-$Fold Cross-Validation. We split our dataset in $K = 5$ different

folds and performance are estimated over the union of all the test folds. During training we use Adam optimizer with learning rate 0.001, small batch sizes (8 for the pre-training of the layers of the FEN, and 2 for training the event-detection network), 200 epochs on each fold and set $\lambda = 5$ in (2) and (4). In addition, to reduce the risk of overfitting, we adopt the early-stopping criteria that the validation error should decrease in 25 epochs.

Data Augmentation. Given the class imbalance and the limited amount of annotated events, when training the FEN for the auxiliary classification task, we resort to several data-augmentation techniques to improve generalization capabilities of our network. We apply, to 10% of randomly selected data in each batch, a right/left translation of the OTDR trace by a random amount between 5% and 25% of the original size. Furthermore, we also shift the power amplitude by adding a random value within $[-8, +8]$. This latter augmentation is applied randomly to 5% of each training batch. On top of these "standard" augmentation transformations, we also adopted mix-up data augmentation [21]. Mix-up has been shown to be very beneficial when training deep CNNs for solving several tasks both in image and time series domains. Indeed, we found this technique is also very effective to improve generalization capabilities of our OTDR event detection network.

5.3 Event Classification Performance

To evaluate performance of the FEN on the auxiliary event classification task mentioned in Section 4.2, we resort to common classification metrics, namely the accuracy, precision, recall and F_1 score. Results are computed averaging performances from all the test folds and are reported in Table 1a. The layers of the FEN are very good at identifying specific events such as no-event and face-plate, while for fiber-end and fiber-cut we found slightly worse performance, as reported in Fig. 3b. This can be 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. 3a).

5.4 Event Detection Performance Metric

In computer vision, object-detection performance are typically assessed via the mean average precision (mAP) score, which is a global measure of classification and localization accuracy. This metric has been introduced in the PASCAL VOC challenge [22]. We also adopt the COCO [23] metric, denoted as mAP@ $[.5, .95]$: which evaluates mAP at 10 different intersection over union thresholds. As both metrics are specifically designed for $2D$ boxes, they have been adapted to cope with our predictions that are instead $1D$ line segments.

5.5 Event detection performance

Our model achieves very satisfactory detection performance with an average precision score (AP@0.5) exceeding 77% in each class, and a mean average pre-

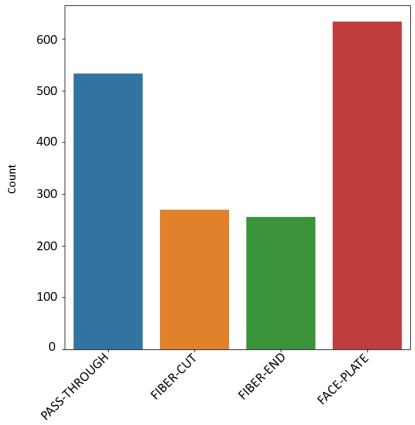

Fig. 3 (a): Class distribution

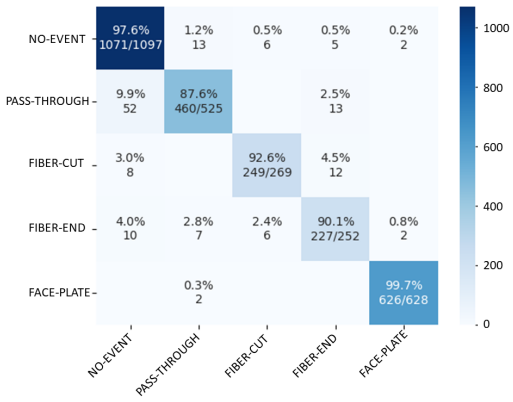

Fig. 3 (b): Confusion Matrix

Metric	**NE**	**FP**	**PT**	**FE**	**FC**	**AVG**
Accuracy	0.9760	0.9970	0.8760	0.9010	0.9260	0.9502
Precision	0.9387	0.9937	0.9544	0.8833	0.9540	0.9448
Recall	0.9763	0.9968	0.8762	0.9008	0.9396	0.9351
F_1 Score	0.9571	0.9952	0.9136	0.8919	0.9396	0.9395
ROC-AUC	0.9872	0.9995	0.9914	0.9842	0.9895	–

Table 1 (a): Classification results over Cross-Validation[2]

Event Type	PASCAL-VOC	COCO
Face-Plate	0.86	–
Pass-Through	0.89	–
Fiber-End	0.77	–
Fiber-Cut	0.88	–
mAP	0.85	0.49

Table 1 (b): Mean AP Scores

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 1 (c): NCS-1001 Comparison

cision equal to 85% among all the classes (see Table 1b). We also achieve 49% of MS-COCO mean average precision.

We have also compared our approach with existing solutions currently embedded in Cisco NCS-1001 (or shortly NCS-1K) for OTDR events detection. To enable a fair comparison, 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 1c show that the proposed detection network substantially outperforms existing solutions on NCS-1K devices, which

[2] No-Event (NE), Face-Plate (FP), Pass-Through (PT), Fiber-End (FE), Fiber-Cut (FC), average among classes (AVG)

implement hand-crafted detectors characterized by thresholds to be tuned, and that operate under strict assumptions on the event position and size. Moreover, the proposed OTDR event-detection network can successfully identify events of different types, and can provide a more accurate localization since it does not process the trace on a fixed-size window basis.

6 Conclusion and future work

In this paper we presented a deep learning model that detects events in OTDR traces. This is a very promising alternative to existing solutions, which are based on simple expert-driven rules and are not flexible enough to identify different types of events along the fiber. We show that the proposed approach is not only able to recognize more event types than existing algorithms, but it is also accurate in localizing them. Remarkably, the proposed solution can be easily extended to detect a larger set of event types, including rare events or events that are less common than those considered in this work.
Finally, our experiments show that the proposed approach can effectively solve the optical event-detection problem, and that can be used in a real-world environment, saving time to optic engineers and providing detailed analysis of OTDR traces without requiring special expertise. Our event-detection network has been already deployed onto Cisco NCS-1001 optical platform, offering an on-demand feature for automatic OTDR trace analysis.

References

[1] M.K. Barnoski et al. “Optical time domain reflectometer”. In: *Applied optics* 16.9 (1977).
[2] Shaoqing Ren et al. “Faster R-CNN: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pp. 91–99.
[3] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587.
[4] Ross Girshick. “Fast R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1440–1448.
[5] Joseph Redmon et al. “You Only Look Once: Unified, real-time object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788.
[6] Kathan Kashiparekh et al. “Convtimenet: A pre-trained deep convolutional neural network for time series classification”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8.
[7] Hassan Ismail Fawaz et al. “Inceptiontime: Finding alexnet for time series classification”. In: *Data Mining and Knowledge Discovery* 34.6 (2020), pp. 1936–1962.

[8] John Cristian Borges Gamboa. "Deep learning for time-series analysis". In: *arXiv preprint arXiv:1701.01887* (2017).

[9] Hassan Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (2019), pp. 917–963.

[10] Özal Yıldırım et al. "Arrhythmia detection using deep convolutional neural network with long duration ECG signals". In: *Computers in Biology and Medicine* 102 (2018), pp. 411–420. ISSN: 0010-4825.

[11] Yue Wu et al. "DeepDetect: A cascaded region-based densely connected network for seismic event detection". In: *IEEE Transactions on Geoscience and Remote Sensing* 57.1 (2018), pp. 62–75.

[12] Dimitri Palaz, Gabriel Synnaeve, and Ronan Collobert. "Jointly Learning to Locate and Classify Words Using Convolutional Networks." In: *INTERSPEECH*. 2016, pp. 2741–2745.

[13] Metin Aktas et al. "Deep learning based multi-threat classification for phase-OTDR fiber optic distributed acoustic sensing applications". In: *Fiber Optic Sensors and Applications XIV*. Vol. 10208. International Society for Optics and Photonics. 2017, 102080G.

[14] Lihi Shiloh, Avishay Eyal, and Raja Giryes. "Deep learning approach for processing fiber-optic DAS seismic data". In: *Optical Fiber Sensors*. Optical Society of America. 2018, ThE22.

[15] Sascha Liehr et al. "Real-time dynamic strain sensing in optical fibers using artificial neural networks". In: *Optics express* 27.5 (2019), pp. 7405–7425.

[16] Huijuan Wu et al. "One-dimensional CNN-based intelligent recognition of vibrations in pipeline monitoring with DAS". In: *Journal of Lightwave Technology* 37.17 (2019), pp. 4359–4366.

[17] Yi Shi et al. "An event recognition method for Φ-OTDR sensing system based on deep learning". In: *Sensors* 19.15 (2019), p. 3421.

[18] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[19] Min Lin, Qiang Chen, and Shuicheng Yan. "Network In Network". In: *2nd International Conference on Learning Representations, ICLR*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

[20] *Optical Time Domain Reflectometer (OTDR) Data Format*. SR-4731. Telcordia Technologies. 2011.

[21] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *6th International Conference on Learning Representations, ICLR*. 2018.

[22] Mark Everingham et al. "The PASCAL visual object classes challenge 2007 (VOC2007) results". In: (2007).

[23] Tsung-Yi Lin et al. "Microsoft COCO: Common objects in context". In: *European Conference on Computer Vision*. Springer. 2014, pp. 740–755.