Model and Data-centric Machine Learning Algorithms to Address Data Scarcity for Failure Identification

LAREB ZAR KHAN [1,*], JOÃO PEDRO [2,3], NELSON COSTA [2], ANDREA SGAMBELLURI [1], ANTONIO NAPOLI [4], AND NICOLA SAMBO [1]

[1] Scuola Superiore Sant'Anna, via G. Moruzzi 1, 56124, Pisa, Italy
[2] Infinera Unipessoal Lda, 2790-078 Carnaxide, Portugal
[3] Instituto de Telecomunicações, Instituto Superior Técnico, 1049-001 Lisboa, Portugal
[4] Infinera, Strategy, Architecture, and Engineering, Munich, Germany
[*] Corresponding author: larebzar.khan@santannapisa.it

Compiled May 30, 2024

The uneven occurrence of certain types of failures in optical networks results in a scarcity of data for less frequent failures, leading to imbalanced datasets for training machine learning (ML) models. This poses a significant bottleneck in terms of reliability and practical implementation of ML for failure management. Existing research works often overlook this aspect while demonstrating high accuracies by utilizing sufficiently balanced training datasets collected in controlled laboratory setups and simulations. However, this approach does not reflect a realistic network scenario. To address this issue, different model-centric and data-centric approaches have been investigated in this work to determine their potential in improving the learning of ML models, specifically neural networks (NNs), on less frequent failures with such imbalanced training datasets. For failure identification, the obtained results suggest that data-centric approaches tend to perform better in terms of classification accuracy, with an improvement of up to 5.5% in F1-score observed on less frequent failures compared to baseline NN (i.e., without any model-centric or data-centric treatment). However, some data-centric approaches may also have significant additional computational complexity associated with them and, therefore, a suitable approach should be chosen based on the desired classification performance and available computational resources.

http://dx.doi.org/10.1364/ao.XX.XXXXXX

1. INTRODUCTION

The provision of uninterrupted and high-quality internet services to end-customers is of paramount importance for network operators [1–3]. Timely detection and rectification of soft failures (i.e., slow and slight degradation of performance) can prevent their progressive deterioration into hard failures (i.e., complete service disruption) [4–6]. Conventional threshold-based failure management methods are not very effective for complex and dynamic optical networks [1], and therefore, in the quest of new methods for autonomous and uninterrupted network operation, machine learning (ML) has gained significant attention due to its ability to learn from the massive amount of data generated by optical networks [7–10]. Over the past few years, numerous ML-based techniques targeting various aspects of soft-failure (hereinafter referred to as failure) management, such as detection, identification, and localization, have been investigated. Failure detection and identification have been studied in [11] as a two-stage scheme in which digital spectrum information has been utilized to train one class support vector machine (OC-SVM) and SVM for failure detection and identification, respectively. The authors of [12] have also investigated failure detection and identification, where they analyzed anomalies in continuously monitored bit error rate (BER) using several ML techniques such as neural network (NN), random forest (RF), and SVM. Decision tree-based failure identification and localization framework has been proposed in [13] that utilizes optical spectrum information. In [14], a probabilistic ML algorithm based on Bayesian networks has been investigated to localize and identify the probable causes of failures. A NN-based failure localization framework in a small-scale laboratory setup has been studied in [4]. Many other similar works investigating ML-based failure management have been listed in these [1, 7, 8] review papers. Almost all these existing works used simulated and/or experimental data to train the ML models while demonstrating good performance. However, we have to consider that data in simulations and laboratory setups is generated in a controlled environment, so typically the desired amount of failure data can be collected. In reality, failures seldomly occur in optical networks under operation and, consequently, the amount of

failure data available to train ML models may be scarce.

In the failure detection scenario, which is a binary classification problem, normal operation data is always far greater than failure data because the network has to operate flawlessly most of the time, and it is not viable to intentionally introduce failures in a live network just for the sake of data collection. Similarly, for failure localization, since failures do not occur evenly across the network, we may have sufficient number of failure samples to train our ML model for some locations but not for others. Likewise, in the case of failure identification, some failures occur more frequently than others, resulting in uneven distribution of samples among different failure types in the training dataset. As a result, datasets collected from real networks for training ML models for failure management are fairly imbalanced, and training ML models on such datasets may result in a bias towards sufficiently well-represented scenarios while negatively impacting performance on less frequent or under-represented scenarios. As an example, this can be seen in the results reported in [15], where field data collected for seven months has been utilized for ML-based failure identification. The imbalance in the resultant dataset biased the performance of employed ML model towards the well-represented failure, resulting in lower F1-scores on under-represented failures. Thus, clearly, the problem of imbalanced datasets for failure management is an open issue and needs investigation.

Few works, such as [16], attempted to address this issue for failure detection by treating it as an anomaly detection problem and training ML models like autoencoder (AE) only on normal operation data, with failures detected based on high reconstruction loss of failure data since these samples were not used during the training phase. This approach, however, cannot be extended to address the imbalanced training problem in failure localization and identification because these are inherently multi-class classification problems. Hence, as rightly suggested by [1], the non-consideration of imbalanced training datasets is a limitation of the majority of existing works that needs to be addressed. A few recent studies [17, 18] investigated this issue in the context of failure identification in optical networks, where data augmentation has been used to create synthetic failure data. In [17], synthetic data has been combined with real experimental testbed data and improved classification performance has been demonstrated. This data augmentation approach falls under data-centric ML as it involves a systematic and algorithmic increase in the quantity and/or quality of the training dataset for a given model, yet the underlying process to generate/process data may impose an extra computational cost that has not been discussed in [17, 18]. Moreover, to the best of the authors' knowledge, no comparison with its counterpart model-centric ML (i.e., creating the "best" model for a given dataset) approaches has been reported in these or any other existing works. Therefore, it remains to be seen which approach is superior and in which aspect when applied to failure management in optical networks.

This paper builds on the authors' work reported in [19], and it aims to fill the aforementioned research gap by providing a direct and comprehensive comparison between different model-centric and data-centric ML approaches for dealing with the imbalanced training problem in ML-based failure management, with failure identification as the considered use-case and NN as the employed ML model. More specifically, while in [19] one model- and one data-centric approach was studied, in this paper, a wider set of approaches are presented, analyzed, and discussed. The model-centric approaches investigated in this work focus on modification of the standard loss function of NN, as well as changing the sampling strategy of batches within a training epoch of NN to account for the imbalanced distribution of failure samples in the training dataset. While modifying loss function, the contribution of different failure classes to the overall training loss has been re-weighted based on the extent of their representation in the training dataset. With focal loss, this approach has been extended to the sample level, further reducing the contribution of correct classifications to the overall loss so that NN could focus more on misclassifications, which are typically from less frequent failure classes. The combined performance of re-weighting of failure classes as well as samples has also been investigated for addressing the imbalanced training problem. As for sampling strategy during training, a balanced mini-batch training approach has been studied that samples imbalanced training data in a balanced manner for each batch within a training epoch.

On the data-centric side, SMOTE, SMOTE-TOMEK and generative and adversarial networks (GANs)-based techniques have been used to augment data of under-represented failures by generating synthetic samples to reduce their scarcity. All these techniques have been individually applied to an optimized baseline NN, and the resultant performance has been compared in terms of classification accuracy (i.e., F1-score) and computational complexity (i.e., sum of the training time of the employed NN-based failure identifier and additional time required for generating synthetic data). The obtained results suggest the superior performance of data-centric approaches as compared to other considered approaches in terms of F1-score, with up to 5.5% improvement observed against optimized baseline NN on under-represented failures. However, it has also been observed that some data-centric approaches can be computationally expensive, so the best balance of performance and computational cost should be determined depending on the scenario.

The remainder of this paper is organized as follows. Section 2 covers the necessary details about the experimental setup and the data acquired from it. Section 3 explains the considered model-centric approaches in this work, followed by Section 4, which provides an overview of the considered data-centric approaches. Section 5 presents a detailed comparative analysis

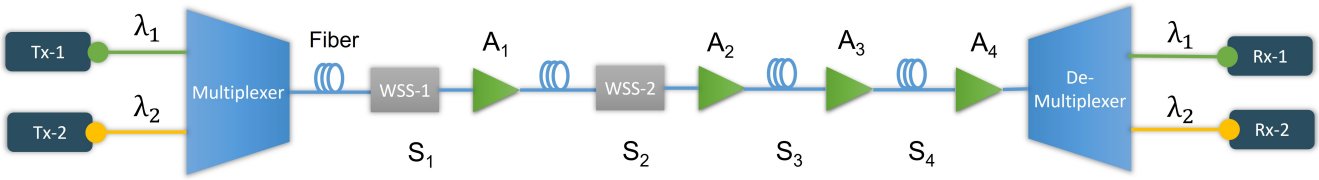

Fig. 1. Experimental testbed setup

Table 1. Considered failures and their corresponding system configuration

Failure type	Channel	Configuration for WSS-1			Configuration for WSS-2		
		Attenuation (dBs)	Central frequency (THz)	Bandwidth (GHz)	Attenuation (dB)	Central frequency (THz)	Bandwidth (GHz)
Filter shift + attenuation (F_1)	1	0	194.12	50	6	194.1	50
	2	5	194.2	50	0	194.24	50
Attenuation (F_2)	1	3	194.1	50	3	194.1	50
	2	4	194.2	50	5	194.2	50
Filter tightening (F_3)	1	0	194.1	26	0	194.1	28
	2	0	194.2	26	0	194.2	50
Filter tightening + attenuation (F_4)	1	0	194.1	26	3	194.1	50
	2	0	194.2	30	4	194.2	50

of the obtained results, and finally, Section 6 summarizes the entire work with some concluding remarks.

2. EXPERIMENTAL SETUP AND DATA ACQUISITION

For this investigation, data has been collected from an experimental testbed shown in Fig. 1. The considered testbed is a part of the ARNO testbed setup at the TECIP Institute of Scuola Superiore Sant'Anna in Pisa, Italy. It includes two Ericsson SPO 1400 optical devices, SPO-1 and SPO-2. Each SPO is equipped with OTN muxponder, exploiting an optical tunable coherent transceiver at 100 Gb/s with dual polarization quadrature phase shift keying (DP-QPSK) modulation format. Each transceiver is composed of a transmitter (Tx) and a receiver (Rx). Two channels i.e., channel-1 and channel-2, with around 37.5 GHz of spectrum width, central frequencies λ_1 = 194.1 THz and λ_2 = 194.2 THz, respectively, have been set up in the testbed for this study.

Two arrayed waveguide gratings (AWGs) have been used at Tx side and Rx side for multiplexing and de-multiplexing the optical signals. The optical link is composed of four spans, labeled S_1, S_2, S_3, and S_4. Each span consists of a single mode fiber spool with length of approximately 80 km, and a fiber attenuation co-efficient of 0.2 dB/km. The attenuation of optical signals along the optical link has been compensated using a series of erbium doped fiber amplifiers (EDFAs) denoted by A_1, A_2, A_3, and A_4 in Fig. 1, configured in a fixed gain mode. In order to artificially generate failures in the network, two wavelength selective switches (WSSs), namely WSS-1 and WSS-2, have been placed at the end of S_1 and S_2, respectively. Four failure conditions on both channels have been considered i.e., filter shift + attenuation (F_1), attenuation (F_2), filter tightening (F_3), and filter tightening + attenuation (F_4). The detailed system configuration for each of these considered failures is given in Table 1.

The bit error rate (BER) and optical signal-to-noise ratio (OSNR) values have been collected from the two receivers using a Kafka-based telemetry system [20] with a sampling interval of approximately 4 seconds. A total of 2000 samples have been collected for each failure per channel and then, with the removal of duplicates followed by the manual removal of samples, an imbalance in the dataset has been created to mimic the real network scenario where higher frequency of occurrence of some failures results in sufficiently well-representation of those failures and under-representation of other failures in the training dataset, as can also be seen in collected field data of [15]. As the considered use-case for our investigation is failure identification rather than failure detection, data is only collected for failures, not for normal operation. Table 2 shows the imbalanced distribution of samples among the considered failures in our case. This particular distribution of failures has been chosen because combinations of different failures are less likely to occur (hence under-represented failures) than individual failures (hence well-represented failures). However, it is worth noting that other experiments highlighted that similar trends and conclusions can be drawn considering different choice of individual failures and other degrees of imbalance among failure classes, indeed as in [19], other failures have also been considered with different degrees of imbalance.

Table 2. Failures' distribution in overall and training dataset

Failure Type	Number of samples in overall dataset		Number of samples in training dataset	
	Channel-1	Channel-2	Channel-1	Channel-2
F_1	360	289	231	184
F_2	1124	1205	717	773
F_3	869	981	552	632
F_4	526	601	328	394

From this imbalanced dataset, 20% of the data has been kept as a test dataset for the final model evaluation after training, and the remaining 80% has been splitted into training and validation datasets using an 80-20% split ratio. That validation dataset has been used during the training phase to ensure that the NN is not overfitting. Table 2 also shows the number of samples in the training dataset for all the considered failures on both channels. It should be noted that the validation and test datasets have not been modified throughout the study, only the training dataset has been modified in the data-centric scenarios. The performance of the NN-based failure identifier after applying all model-centric and data-centric techniques has been evaluated on the same unmodified test dataset.

3. MODEL-CENTRIC MACHINE LEARNING

The primary objective of model-centric ML is to produce the "best" model for a given training dataset. It accomplishes this

by modifying the model to improve its performance on a specific ML task [21], in this case failure identification in optical networks. There are several ways to obtain the best model, including:

i) Changes to the loss function (which captures the difference between the actual and predicted values);

ii) Modification of strategy to sample dataset during training;

iii) Tuning of hyperparameters (which are model parameters that determine its architecture and cannot be changed during training);

iv) Regularization to prevent overfitting, etc.

The problem we are addressing in this work is the scarcity of data for less frequent failures, which impacts the performance of NNs in the identification of such failures. Therefore, the modification of the loss function and sampling strategy are perhaps the most appropriate approaches to tackle this problem in a model-centric manner. As for modification of the loss function, the most commonly used loss function for multi-class classification problems like failure identification is categorical cross entropy (CCE) loss, also known as SoftMax loss. If $\boldsymbol{s}$ denotes a vector containing the linear output (i.e., before applying non-linear activation) of all N neurons in the final layer of a NN, then CCE loss can be computed as

$$CCE\ Loss = -\sum_{i=1}^{N} \boldsymbol{y_i} \log\left(\frac{e^{s_i}}{\sum_{j=1}^{N} e^{s_j}}\right), \tag{1}$$

where $\boldsymbol{y_i}$ represents the one-hot encoded true label of class i (i.e., 1 if the sample belongs to class i, and 0 otherwise). Eq. 1 also suggests that the CCE loss function is basically a SoftMax activation i.e., $p(s)_i = \frac{e^{s_i}}{\sum_{j=1}^{N} e^{s_j}}$ followed by cross entropy (CE) i.e., $CE = -\sum_{i=1}^{N} y_i \log(p(s)_i)$. To further illustrate the computation of loss using the CCE loss function, we provide the following toy example.

Toy Example: Consider a scenario where we have five possible failures (i.e., N = 5), namely sf_1, sf_2, sf_3, sf_4, and sf_5, and the given input $\mathbf{x}$ containing BER and OSNR values at a given time instant corresponds to sf_2, hence, y_i = [0, 1, 0, 0, 0]. If the predicted linear output of the final layer of a NN-based failure identifier is $\boldsymbol{s}$ = [1.23, 3.5, 0.28, 1.7, 0.5], then by applying SoftMax activation, we can determine the likelihood $p(s)_i$ of each failure. For sf_1, $p(s)_1 = \frac{e^{1.23}}{e^{1.23}+e^{3.5}+e^{0.28}+e^{1.7}+e^{0.5}} = 0.076$, and similarly for other failures (sf_2, sf_3, sf_4, and sf_5), likelihood calculates to 0.736, 0.029, 0.121 and 0.036, respectively. Once, likelihoods have been determined, loss can be computed using Eq. 1 as $CCE = -(0 \times log(0.076) + 1 \times log(0.736) + 0 \times log(0.029) + 0 \times log(0.121) + 0 \times log(0.036)) = 0.133$.

The CCE loss function may have a disadvantage of treating all failures equally during the training of a NN, regardless of their distribution. As a result, the total loss minimized during training may not accurately depict actual performance on under-represented failures. Therefore, NN may misclassify the under-represented failures as well-represented failures, limiting the performance of NN-based failure identifiers. The following three subsections will explain some of the ways in which CCE loss function can be modified to address this problem.

A. Weighted Categorical Cross Entropy Loss

One model-centric approach that is commonly used in computer science to deal with imbalanced training datasets is to apply weights to the CCE loss function, resulting in weighted categorical cross entropy (WCCE) loss. Each class is assigned a weight to reflect its importance during training. For class i, corresponding weight can be computed as:

$$W_i = \frac{n_{\text{total\_samples}}}{N \times n_{\text{samples}_i}}, \tag{2}$$

where $n_{total\_samples}$ represents the total number of samples in the training dataset, N the total number of failure classes, and $n_{samples_i}$ the number of samples in the i^{th} failure class. These class weights can then be incorporated in CCE loss function (Eq. 1) as:

$$WCCE\ Loss = -\sum_{i=1}^{N} (\boldsymbol{y_i} \odot \boldsymbol{w}) \log\left(\frac{e^{s_i}}{\sum_{j=1}^{N} e^{s_j}}\right), \tag{3}$$

where $\odot$ indicates element-wise multiplication of one-hot encoded true class label vector ($\boldsymbol{y_i}$) and weight vector ($\boldsymbol{w}$) containing calculated weights for all failure classes. As weights are proportional to the inverse of failure class representation in the training dataset, this results in higher weights for under-represented failures and lower weights for well-represented failures. These weights are then used to adjust the contribution of each failure class to the overall loss during training, as suggested by Eq. 3. This often helps to mitigate the impact of class imbalance and keeps the NN from being biased towards the well-represented class. However, this approach may also have a limitation as it re-weights the contribution of the entire failure class, which may penalize correct predictions as well.

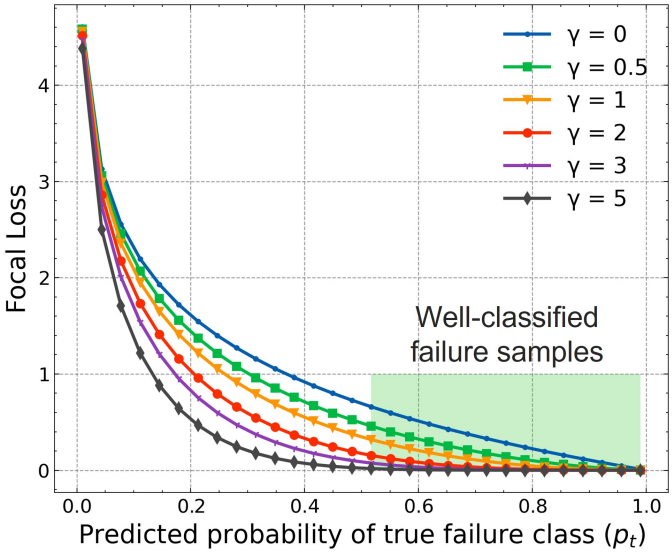

Fig. 2. Focal loss vs. predicted probability of the true failure class for different values of γ

B. Focal Loss

To overcome the potential limitation of WCCE loss, the re-weighting approach can be extended to the sample level with the focal loss function, as proposed in [22], which is a modified version of the CCE loss function and it focuses more on misclassified failure samples rather than the entire failure class. If

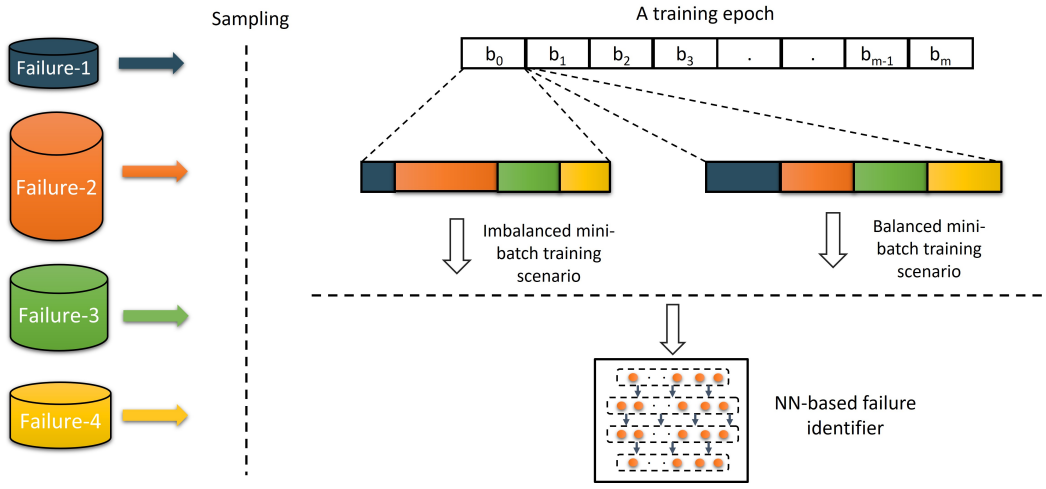

Fig. 3. Illustrative comparison between imbalanced mini-batch training (typical scenario) and balanced mini-batch training

p_t denotes the predicted probability (i.e., after SoftMax activation) for the true/actual failure class t, then the CCE loss (Eq. 1) reduces to:

$$CCE\ Loss_{p_t} = -\log(p_t). \tag{4}$$

The modulation of resultant loss (Eq. 4) with $(1-p_t)^\gamma$ term, where γ ($\gamma \geq 0$) is a focusing parameter, results in the focal loss function, given as:

$$Focal\ Loss = -(1-p_t)^\gamma \log(p_t). \tag{5}$$

Fig. 2 shows the focal loss as a function of p_t for different values of γ , where $\gamma = 0$ represents the CCE loss function (Eq. 4). If $p_t \geq 0.5$, then a sample is considered as well-classified, and all such failure samples fall in the green highlighted region indicated in Fig. 2. Ideally, for a correct classification of a failure sample, loss should be minimal. But, with $\gamma = 0$, loss contribution from well-classified failure samples is not insignificant and thus should be further reduced. With modulating factor i.e., $(1-p_t)^\gamma$ controlled by γ, the contribution from well-classified failure samples to the overall loss can be significantly reduced, as evident from Fig. 2. Eventually, NN focuses more on misclassifications during training, which are typically from under-represented failure classes. As the majority of samples in imbalanced training datasets belong to well-represented failure classes, NN can learn their underlying patterns well, resulting in correct classifications. However, the abundance of samples from well-represented failures tends to overshadow NN's inability to correctly identify less frequent failures, so minimizing the contribution of well-classified samples to overall loss is important.

For a given dataset, a suitable value of γ can be tuned and in our case, $\gamma = 2$ has been used to tackle this issue.

C. Weighted Focal Loss

It is intuitive to investigate the combined performance of re-weighting of failure samples and classes, which led us to combine both previously discussed model-centric techniques. We applied weights to focal loss function to determine if any improvement in performance when dealing with the imbalance training issue can be achieved in this scenario. The resultant weighted focal loss can be expressed as:

$$Weighted\ Focal\ Loss = -W_t(1-p_t)^\gamma \log(p_t), \tag{6}$$

where W_t denotes the weighted contribution of true failure class t, determined using Eq. 2.

In addition to these approaches that modify NN's standard loss function to tackle the issue of imbalanced training, we also investigated another model-centric approach that modifies model's behavior by changing the strategy through which NN samples training data. The following subsection provides an overview of this approach.

D. Balanced mini-batch training

Balanced mini-batch training (BmBT) modifies the mini-batch gradient descent-based training process of a NN by sampling batches from training data in a slightly different manner. In principle, a NN is trained in terms of epochs, with an epoch being one forward pass and one backward pass (i.e., error backpropagation) of all the samples in the training dataset. Mini-batch gradient-based training, which is commonly used to overcome the limitations of batch gradient descent (where NN weights are adjusted after passing all samples) and stochastic gradient descent (where NN weights are adjusted after passing every single randomly selected sample), passes all training samples in batches, with batch size being a hyperparameter. For a given training dataset, the total number of batches (m) for each epoch is given as:

$$m = \frac{Total\ number\ of\ samples\ in\ training\ dataset}{batch\ size}. \tag{7}$$

In a typical epoch of mini-batch gradient descent-based training of NN, each batch maintains the actual distribution of all classes, which means that if the given training dataset is imbalanced, each sampled batch will be imbalanced as well. However, in BmBT, the random sampling from the training dataset for each batch is performed in such a way that an equal number of samples from each class are drawn, resulting in an equal representation of all failure classes in each batch. Fig. 3 illustrates the difference between balanced and imbalanced mini-batch training scenarios.

4. DATA-CENTRIC MACHINE LEARNING

The data-centric approaches focus on the algorithmic increase in the quantity and/or quality of the training dataset for a given ML

model. For addressing the imbalanced training problem, data-augmentation is the most commonly used data-centric approach. Data augmentation systematically increases the amount of training data by generating new synthetic samples from already existing data [23]. However, data augmentation may present limited performance if the amount of real (i.e., already existing) data is extremely low. Data augmentation can be broadly classified into two categories: (i) ML-based and (ii) non-ML-based techniques to generate synthetic samples. In ML-based techniques, deep generative modelling techniques – like variational autoencoders (VAEs) and variants, Generative and adversarial networks (GANs) and variants, Normalizing Flows, Energy-Based and Autoregressive Models – generate synthetic samples (a review on these techniques can be found in [24]), with typically requiring a relevant computational complexity. On the other hand, non-ML-based methods, mainly SMOTE-based techniques, are simpler and typically based on linear interpolation, therefore there is no need to train any model unlike ML-based methods to produce new synthetic samples. In general, data augmentation helps to increase the diversity in the training dataset, which often improves the performance and robustness of ML models. Among many existing data augmentation approaches in the literature, we investigated three in the scope of this work: SMOTE, its variant i.e., SMOTE-TOMEK and a GAN-based approach. These methods have been chosen because they represent two computational cost extremes. SMOTE-based approaches are computationally very cheap, while the GAN-based approaches are highly computationally intensive. Other data augmentation techniques usually fall somewhere in between these two extremes in terms of computational cost. It is worth noting again here that in this work, only training data has been augmented with data-centric ML techniques, while the test dataset has been kept unmodified for the final performance evaluation.

The three considered data augmentation techniques are discussed in the following subsections.

A. SMOTE

Synthetic minority oversampling technique (SMOTE) is an oversampling technique proposed in [25]. The main idea behind SMOTE is to perform oversampling of under-represented classses by interpolation of their samples within a defined neighborhood. The K-nearest neighbors method [26] is used to determine the nearest samples of a randomly selected under-represented class sample. Here, K specifies the number of nearest samples to be selected from the same under-represented class.

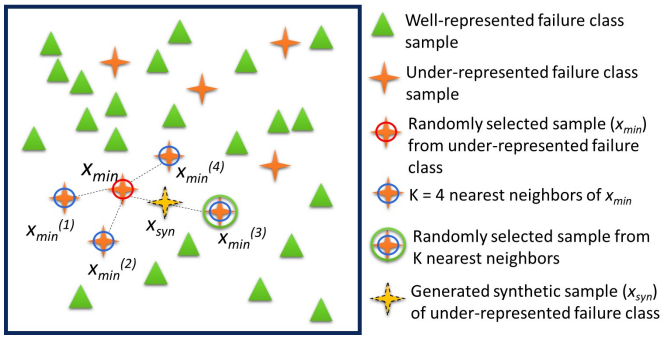

Fig. 4. Synthetic minority oversampling technique (SMOTE)

Fig. 4 illustrates how SMOTE generates synthetic samples for a two-class scenario, which can simply be extended to multiple classes. First, a random sample (x_{min}) is selected from the under-represented class. Then its K-nearest neighbors ($x_{min}^{(1)}$, $x_{min}^{(2)}$, $x_{min}^{(3)}$, ... , $x_{min}^{(K)}$) are determined; in this example, $K = 4$. One of those nearest neighbors is then chosen at random, $x_{min}^{(3)}$ in this case, and a synthetic sample is generated. This synthetic sample is placed at a position determined by multiplying a random number (between 0 and 1) with the Euclidean distance between the original sample (x_{min}) and its chosen neighbor ($x_{min}^{(3)}$). This procedure is repeated until the desired number of samples are obtained for that under-represented class.

B. SMOTE-TOMEK

SMOTE-TOMEK [27] is a variant of SMOTE in which oversampling of under-represented class using SMOTE is followed by the slight downsampling of well-represented class using TOMEK links [28] approach, with an objective to clarify the boundary between the under-represented and well-represented classes, making the under-represented region(s) more distinct. A TOMEK link is formed when two samples are nearest neighbors to each other but belong to two different classes. Fig. 5 illustrates how TOMEK links are identified, and samples from the well-represented class are removed to break the TOMEK links, which is likely to reduce the number of misclassifications.

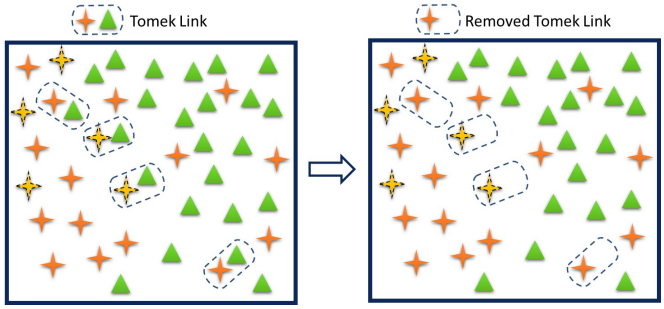

Fig. 5. Illustration of identification and removal of TOMEK links

C. Conditional Tabular Generative and Adversarial Network

Conditional Tabular Generative and Adversarial Network (CT-GAN) is a variant of a standard Generative and Adversarial Network (GAN) [29], and it is designed to perform augmentation of the tabular data. This subsection aims to provide an overview on the working of GAN, its limitations while handling tabular data and how CT-GAN overcomes those limitations.

Fig. 6 shows the working of GAN. A GAN consists of two NNs, a generator (G) and a discriminator (D). A generator takes a random noise vector (z), also known as a latent vector, as input and outputs a synthetic data sample G(z). The discriminator, on the other hand, is a binary classifier whose goal is to distinguish synthetic data from real data. Both generator and discriminator compete in an adversarial manner to achieve their respective goals. The training goal of the generator is to maximize the likelihood of the discriminator making a mistake, whereas the training goal of the discriminator is to maximize its probability of correctly identifying real and synthetic data.

The loss function of this standard GAN is given as

$$\min_G \max_D \mathcal{L}(G,D) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))], \quad (8)$$

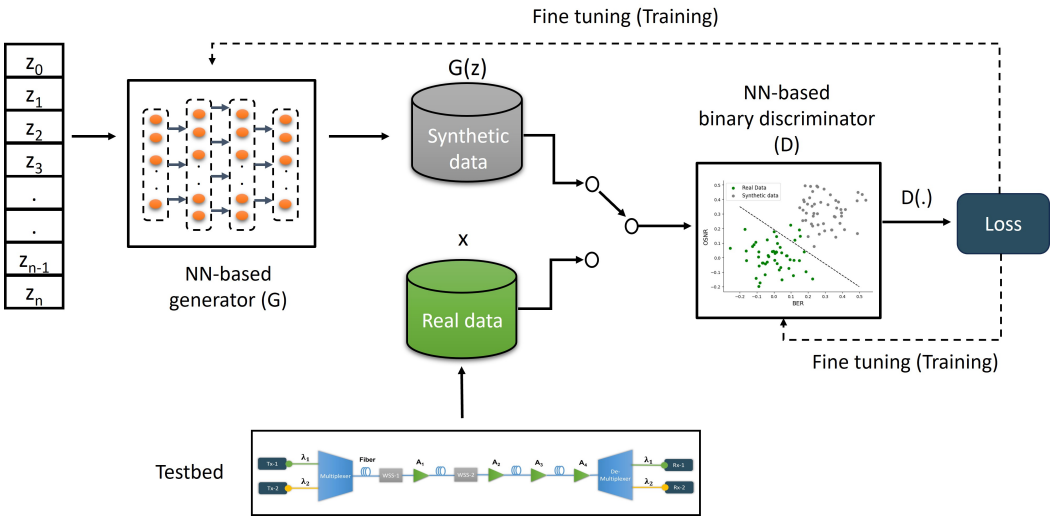

Fig. 6. Generative and adversarial network

where the first term is the discriminator's prediction on real data and the second term is the discriminator's prediction on generated synthetic data. The goal of discriminator is to have a large D(x) value, which indicates high confidence that the real sample is classified as real. Moreover, the discriminator wants D(G(z)) to be as small as possible in order to ensure that synthetic samples are classified as non-real samples, whereas the generator wants D(G(z)) to be as large as possible. Hence, both generator and discriminator compete to achieve their conflicting goals. Overall, the generator tends to minimize Eq. 8, whereas the discriminator tends to maximize it.

A standard GAN struggles with the augmentation of tabular data because such data typically contains mixed datatypes (i.e., integers, floats, and categorical) with different distributions, but it is good at handling a single data type, such as in images. Moreover, standard GAN suffers from issues like mode collapse [30], which occurs when the generator produces a limited set of outputs, collapsing multiple modes or patterns into a single or smaller subset to easily fool the discriminator, but this eventually results in synthetic data that is not diverse and thus undesirable. Another issue is unstable training [30], as generator and discriminator compete in an adversarial manner, the loss function of a standard GAN (given in Eq. 8) often leads to non-convergent training. In addition, standard GAN also struggles to maintain the same distribution in generated synthetic data as in real data specially when dealing with highly imbalanced categorical features.

CT-GAN has been proposed in [31] to address these issues and demonstrated that high quality augmentation of tabular data can be performed. It overcomes the mixed data type problem by using both SoftMax and tanh activation functions at the output layer. The Wasserstein loss [32] function with gradient penalty addresses the problem of mode collapse and unstable training. CT-GAN's conditional generation allows the generator to consider additional information, such as class labels or other relevant attributes, during the data generation process. Conditioning on this information ensures that the generated data aligns with the distributions of the real data. Based on all these improvements and the results demonstrated in [31], it can be inferred that the CT-GAN is one of the most effective techniques for tabular data augmentation. However, it is important to note that a very scarce representation of a class can lead to inadequate learning of CT-GAN which consequently impacts the model's performance.

For our investigation, the noise vector has a length of 100 in the CT-GAN employed for augmentation of data for the two under-represented failure classes, implying that the input layer of the generator NN also has 100 neurons. In addition, the generator also has four hidden layers with 270, 256, 129, and 75 neurons, as well as an output layer with 4 neurons. Discriminator, on the other hand, has four neurons in the input layer, three hidden layers with 151, 256, and 36 neurons, respectively, and a single neuron in the output layer as it acts as a binary classifier. The generator and discriminator has a learning rate of 2×10^{-4}, with batch size of 10. After the training of CT-GAN, 10,000 samples in total have been generated for the two least represented failure classes i.e., F_1 and F_4, and then the required number of samples for each channel have been mixed with the real data to increase the representation of these failures in the training dataset and the resultant modified dataset has been used to train NN for failure identification. It is worth noting that for the modified training dataset, the representation of these under-represented failures has not been increased to the level of the most prevalent failure class, as this is not always required [17], but rather the number of samples for the two under-represented failure classes have been upscaled according to W_i (i.e., corresponding weights for i^{th} failure class, determined using Eq. 2).

5. RESULTS AND COMPARATIVE ANALYSIS

The performance of considered model-centric and data-centric approaches has been compared with each other against a baseline NN. Optuna, a Bayesian optimization-based tool, has been used to tune the hyperparameters of baseline NN. The resultant baseline NN (i.e., with tuned hyperparameters) contained 3 neurons in the input layer, 189 neurons in the first hidden layer, 220 neurons in the second, and 4 neurons in the output layer. The dropout regularization rate has been 0.159 to prevent overfitting, with the learning rate of 8.25×10^{-4} and batch size of 64. Tanh has been used as the activation function for the hidden layers, while SoftMax has been used for the output layer. All the aforedescribed techniques have been applied to this optimized baseline NN, and the performance has been evaluated in

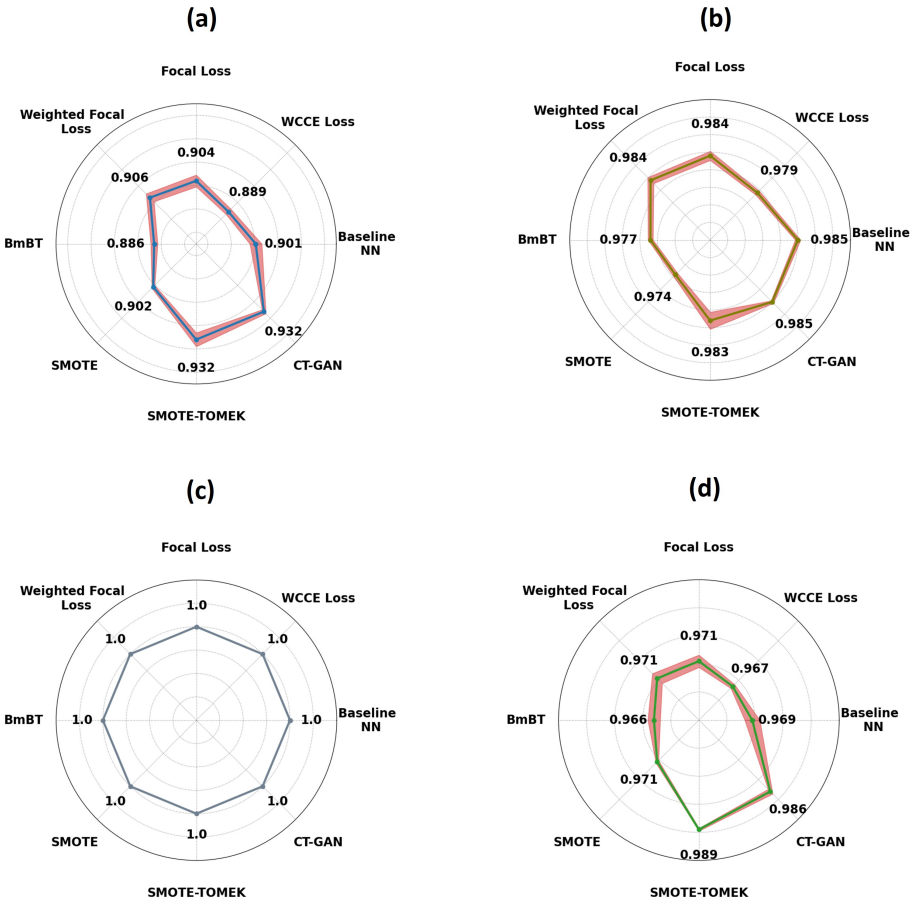

Fig. 7. F1-score comparison of all the considered techniques on channel-1 for (a) F_1, (b) F_2, (c) F_3, and (d) F_4

terms of computational cost and F1-score for both the channels. F1-score is one of the appropriate metrics to evaluate classification performance in case of class imbalance because it takes into account both precision (i.e., the ratio of true positives to the sum of true positives and false positives, measuring the model's ability to avoid false positive errors) and recall (i.e., the ratio of

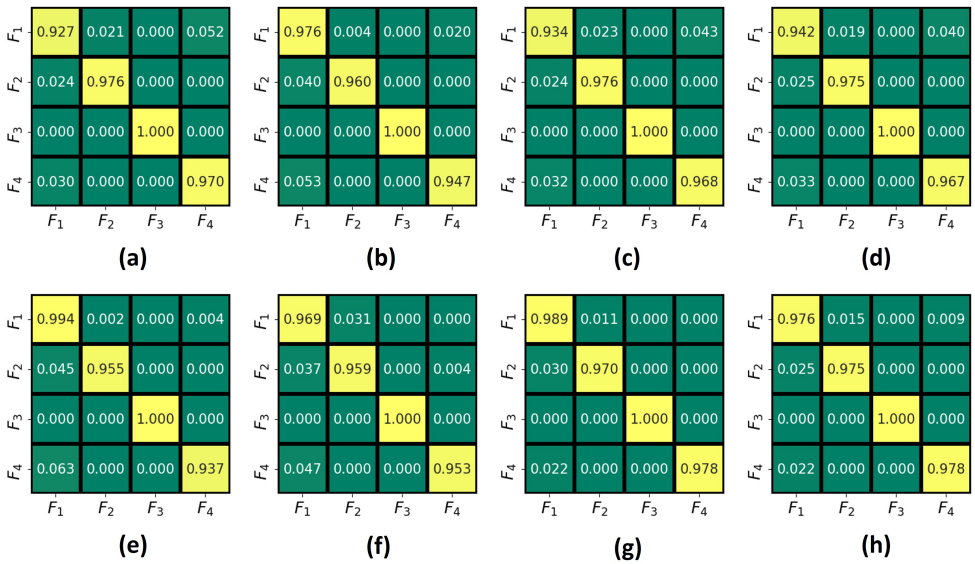

Fig. 8. Normalized confusion matrix on channel-1 for (a) Baseline NN, (b) WCCE loss, (c) Focal loss, (d) Weighted Focal loss, (e) BmBT, (f) SMOTE, (g) SMOTE-TOMEK, and (h) CT-GAN

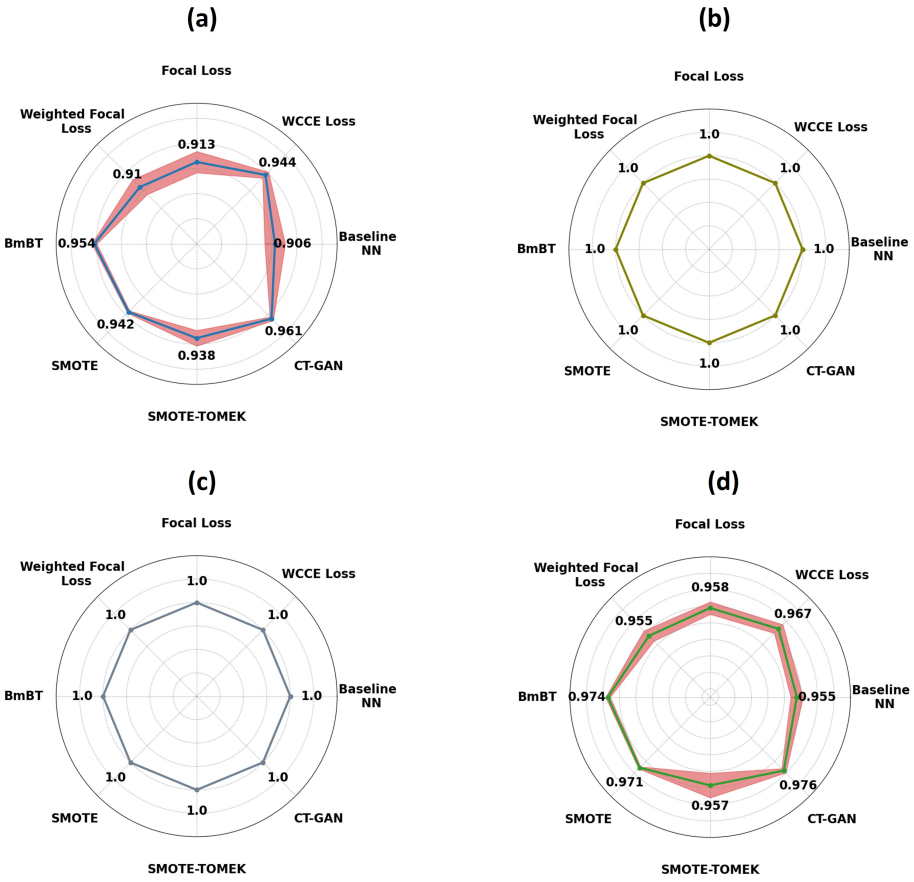

Fig. 9. F1-score comparison of all the considered techniques on channel-2 for (a) F_1, (b) F_2, (c) F_3, and (d) F_4

true positives to the sum of true positives and false negatives, quantifying the model's ability to avoid false negative errors). We also conducted a statistical test i.e., Cochran's Q-test [33] with the "mlxtend" [34] python package to ensure that the difference in performance of the NN-based failure identifier after incorporating the considered model and data-centric techniques is statistically significant.

Fig. 7 shows the results on channel-1 in terms of the obtained

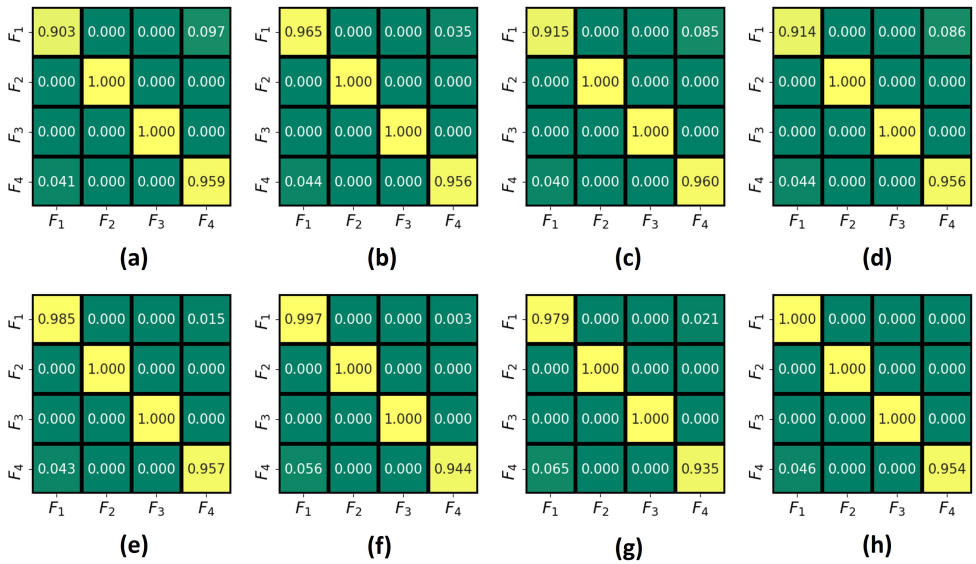

Fig. 10. Normalized confusion matrix on channel-2 for (a) Baseline NN, (b) WCCE loss, (c) Focal loss, (d) Weighted Focal loss, (e) BmBT, (f) SMOTE, (g) SMOTE-TOMEK, and (h) CT-GAN

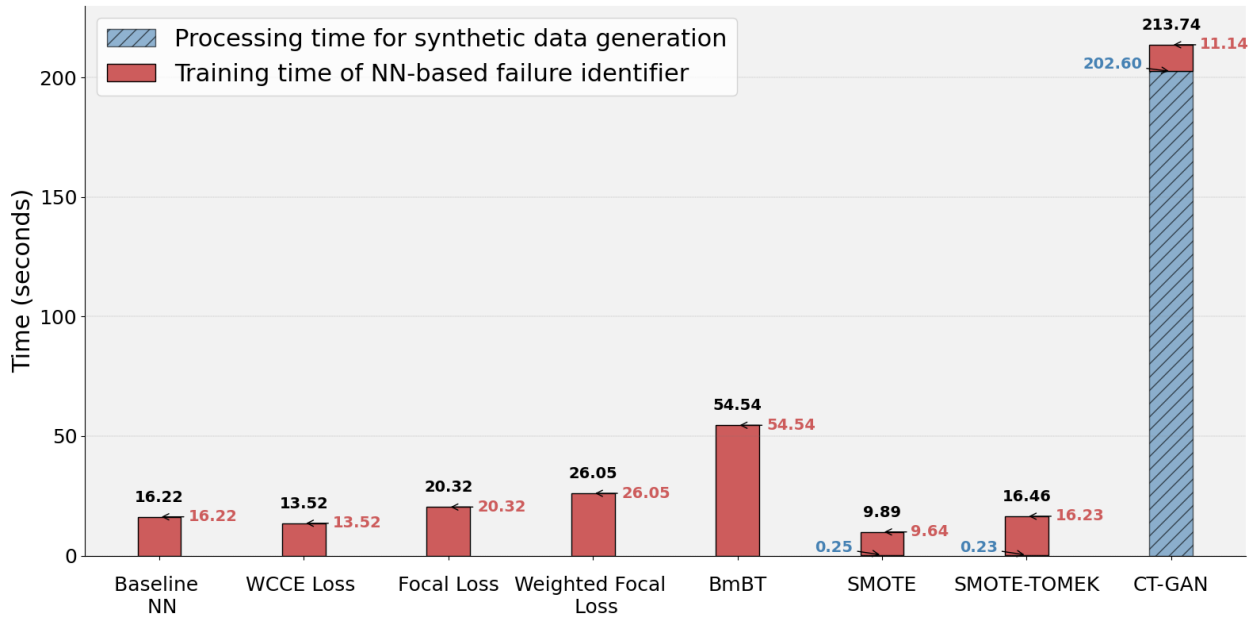

Fig. 11. Computational cost comparison of all the considered techniques

F1-score on the unmodified and imbalanced test dataset for all the considered failures. The reported results here are the averaged results of 100 experiments (i.e., NN training and testing), with the 95% confidence interval indicated in light red color around the mean values. Since F_1 has been the failure with the fewest samples in the training dataset (refer to Table 2), we can see that the F1-scores obtained on it are on the lower side as compared to other failures with more samples. As shown in Fig. 7(a), data-centric approaches, in general, performed better than model-centric approaches for this under-represented failure, with an improvement of 3.1% being observed in this case. Few of the model-centric approaches (i.e., focal loss and weighted focal loss) achieved similar performance as data-centric approaches like SMOTE-TOMEK and CT-GAN for F_2 (Fig. 7(b)), i.e., one of the well-represented failure classes in the training dataset, however, none of the approaches could outperform baseline NN, which classified this failure extremely well with an F1-score of 0.985. For F_3 (Fig. 7(c)), the other well-represented failure class, the employed NN-based failure identifier learnt the underlying pattern extremely well in all scenarios, resulting in its perfect classification. For the second least represented failure class in the training dataset i.e., F_4 (Fig. 7(d)), we observed a similar tendency as F_1 that data-centric techniques perform better, with up to 2% improvement observed. For deeper insights on performance in terms of true positives, true negatives, false positives and false negatives, the corresponding normalized confusion matrices indicating obtained results on test dataset for all the investigated techniques are also reported in Fig. 8.

Similarly, the obtained F1-score results for channel-2 are shown in Fig. 9 and the corresponding normalized confusion matrices for all the considered techniques are reported in Fig. 10. On channel-2, CT-GAN, a NN-based data-centric approach, has showed the highest improvement (5.5%) for the least represented failure class. However, the other considered data-centric approaches have been unable to provide as significant improvement as CT-GAN. Nevertheless, it is interesting to note here that the downsampling (through TOMEK link removal) after oversampling (using SMOTE) may not always be required. For some datasets, only oversampling is sufficient to achieve an already significant/relevant gain, as suggested by the results on channel-2. The BmBT i.e., a model-centric approach, also performed well on this channel, providing F1-score improvements comparable to CT-GAN. Among other model-centric approaches, only WCCE loss has shown some gains in comparison to focal loss and weighted focal loss, which is opposite to the trend observed on channel-1. It is most likely due to different failure distributions on both channels, as indicated by Table 1, because WSS configurations were not identical for both channels for a given failure. It is important to note that the absolute improvements achieved in comparison to the baseline NN for all approaches on both channels may not appear very significant at first. But, we should also consider that it is because the NN used in all scenarios has hyperparameters that have been optimized specifically for the baseline NN using Bayesian optimization. In comparison to other approaches such as grid search or random search, Bayesian optimization is an effective method for tuning hyperparameters. Hence, the tuned hyperparameters favor the baseline scenario. Nevertheless, the objective of this work is to compare model-centric and data-centric approaches using NNs with exactly the same hyperparameters. If we tune the hyperparameters of the NN for either of the model-centric or data-centric approaches, it will favor that particular approach. Alternatively, if we optimize hyperparameters for NN in all the considered scenarios separately, then it will result in different NN architectures which may not lead to a fair and insightful comparison. However, it will be interesting to see how performance (F1-score) degrades on completely out-of-distribution data with new and unseen failure modes, as well as how frequently periodic re-training/fine-tuning may be required to avoid drift in the deployed model's performance for failure identification. This remains as a topic of future research.

As of this point, we have discussed performance in terms of the F1-score, which presents only one aspect of overall performance. As the primary goal of this study is to provide a

comprehensive comparative analysis of the model-centric and data-centric approaches, we also considered the computational cost associated with each approach. Fig. 11 shows the averaged computational time for each approach in terms of training time of the employed NN for failure identification after the application of each technique, as well as the processing time (applicable only to data-centric techniques) for synthetic data generation. The processing time includes training time of ML models, i.e., in case of CT-GAN, and/or time taken to generate synthetic samples for training dataset. These times have been monitored on an Intel(R) Core(TM) i7-12700H @ 2.30 GHz with NVIDIA GeForce RTX 3060 Laptop GPU.

As can be seen, with BmBT approach, the NN-based failure identifier took the longest training time. The possible reason for this is that in our implementation of this technique, random sampling of training data has been performed for each batch within an epoch to select an equal number of samples from each failure class, which prolonged the training process. On the other hand, training time of NN-based failure identifier, with additional synthetic samples in training dataset generated with data-centric treatment , has been comparatively shorter because the addition of synthetic samples to the under-represented failure classes increased their overall representation in the training dataset which resulted in faster NN convergence. However, for ML-based data-centric approaches like CT-GAN, the quick convergence and improved classification performance has been achieved at the cost of high additional processing time i.e., time used to train CT-GAN for generation of synthetic data. This high training time of CT-GAN is because of its inherent adversarial nature, as explained in Section 4C. SMOTE and SMOTE-TOMEK, the other considered data-centric approaches, generate synthetic samples through linear interpolation, so there is no ML model involved for data augmentation and thus no significant additional processing time, as evident from the Fig. 11 as well.

It should also be noted that the combined performance of data-centric and model-centric approaches is not considered in this manuscript because combining both approaches may not necessarily improve performance but may actually even worsen the performance. This is because data-centric, as part of data-preprocessing, will always come first in the implementation pipeline, as compared to model-centric. And, once the data-centric treatment has balanced the training dataset, assigning different priorities to failure classes is likely to degrade performance. Nevertheless, based on the obtained results, it can be inferred that the data-centric approaches tend to be more effective in handling imbalanced scenarios because the addition of synthetic samples diversifies the training dataset, allowing the model to better learn and recognize different patterns in the dataset and, as a result, improve performance on the test dataset. However, some data-centric approaches may require significant additional processing time. When training of the employed NN for failure identification has to be performed offline and the availability of computational resources such as GPUs is not an issue, GAN-based approaches such as CT-GAN can be used. However, if computational resources are very limited then SMOTE-based data-centric approaches can be used to augment training data. Hence, there is always a trade-off between computational cost and classification performance.

6. CONCLUSION

We investigated and compared the effectiveness of model-centric and data-centric ML techniques in addressing the issue of limited data for some failures in the training dataset for ML-based failure identification in optical networks. The considered model-centric approaches involved the modification of NN's loss function to prioritize under-represented failures while reducing the dominance of sufficiently well-represented failures in the overall loss. Moreover, we extended this approach to the sample level, reducing the contribution of correctly predicted failure samples to the overall loss, allowing the NN to focus more on misclassifications, which are usually more from the under-represented failure classes. We also combined these two approaches to see if any performance improvement on under-represented failures could be achieved. In addition, we considered another slightly different model-centric approach i.e., balanced mini-batch training in which training process of NN was modified by sampling the imbalanced dataset during training epochs in a balanced manner. On the data-centric side, we performed data augmentation of under-represented failures using SMOTE, SMOTE-TOMEK and CT-GAN, and then mixed generated synthetic samples with experimental data to increase the representation of under-represented failures in the training dataset. All these approaches were applied to a NN with the same hyperparameters tuned for baseline NN, and the performance was evaluated in terms of F1-score and computational cost. The results obtained on the unmodified test dataset suggest that data-centric approaches tend to perform better in terms of F1-score, with an improvement of up to 5.5% being observed on under-represented failures when compared to baseline NN; however, it has also been observed that some data-centric approaches may have a significant computational cost associated with them. Therefore, based on the scenario and acceptable trade-offs, the most suitable approach can be chosen.

FUNDING

This work has been funded by EU H2020 Marie Skłodowska-Curie Actions ITN project MENTOR (GA 956713) and the Horizon Europe SEASON project (GA 101096120).

REFERENCES

1. D. Wang, C. Zhang, W. Chen, H. Yang, M. Zhang, and A. P. T. Lau, "A review of machine learning-based failure management in optical networks," Sci. China Inf. Sci. **65**, 211302 (2022).
2. A. P. Vela, M. Ruiz, F. Fresi, N. Sambo, F. Cugini, G. Meloni, L. Potì, L. Velasco, and P. Castoldi, "BER degradation detection and failure identification in elastic optical networks," J. Light. Technol. **35**, 4595–4604 (2017).
3. L. Z. Khan, A. Triki, M. Laye, and N. Sambo, "Optical network alarms classification using unsupervised machine learning," in *2022 27th OptoElectronics and Communications Conference (OECC) and 2022 International Conference on Photonics in Switching and Computing (PSC)*, (2022), pp. 1–3.
4. K. S. Mayer, R. P. Pinto, J. A. Soares, D. S. Arantes, C. E. Rothenberg, V. Cavalcante, L. L. Santos, F. D. Moraes, and D. A. A. Mello, "Demonstration of ML-assisted soft-failure localization based on network digital twins," J. Light. Technol. **40**, 4514–4520 (2022).
5. J. Babbar, A. Triki, R. Ayassi, and M. Laye, "Machine learning models for alarm classification and failure localization in optical transport networks," J. Opt. Commun. Netw. **14**, 621–628 (2022).
6. B. Shariati, M. Ruiz, J. Comellas, and L. Velasco, "Learning from the optical spectrum: Failure detection and identification," J. Light. Technol. **37**, 433–440 (2019).
7. F. N. Khan, Q. Fan, C. Lu, and A. P. T. Lau, "An optical communication's perspective on machine learning and its applications," J. Light. Technol. **37**, 493–516 (2019).

8. F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, and M. Tornatore, "An overview on application of machine learning techniques in optical networks," IEEE Commun. Surv. & Tutorials **21**, 1383–1408 (2019).
9. D. Rafique and L. Velasco, "Machine learning for network automation: overview, architecture, and applications [invited tutorial]," J. Opt. Commun. Netw. **10**, D126–D143 (2018).
10. C. Tremblay, "Towards cognitive management and performance monitoring in coherent optical networks," in *2020 Conference on Lasers and Electro-Optics (CLEO),* (2020), pp. 1–2.
11. L. Shu, Z. Yu, Z. Wan, J. Zhang, S. Hu, and K. Xu, "Dual-stage soft failure detection and identification for low-margin elastic optical network by exploiting digital spectrum information," J. Light. Technol. **38**, 2669–2679 (2020).
12. S. Shahkarami, F. Musumeci, F. Cugini, and M. Tornatore, "Machine-learning-based soft-failure detection and identification in optical networks," in *2018 Optical Fiber Communications Conference and Exposition (OFC),* (2018), pp. 1–3.
13. A. P. Vela, B. Shariati, M. Ruiz, F. Cugini, A. Castro, H. Lu, R. Proietti, J. Comellas, P. Castoldi, S. J. B. Yoo, and L. Velasco, "Soft failure localization during commissioning testing and lightpath operation," J. Opt. Commun. Netw. **10**, A27–A36 (2018).
14. M. Ruiz, F. Fresi, A. P. Vela, G. Meloni, N. Sambo, F. Cugini, L. Poti, L. Velasco, and P. Castoldi, "Service-triggered failure identification/localization through monitoring of multiple parameters," in *ECOC 2016; 42nd European Conference on Optical Communication,* (2016), pp. 1–3.
15. C. Tremblay, A. Mahmoudialami, P. Alain, N. Sigue, Y. Pei, D. Côté, D. Boertjes, D. Demeter, and C. Desrosiers, "Detection and root cause analysis of performance degradation in optical networks using machine learning," in *2023 European Conference on Optical Communication (ECOC),* (2023), pp. 1–4.
16. S. Liu, D. Wang, C. Zhang, L. Wang, and M. Zhang, "Semi-supervised anomaly detection with imbalanced data for failure detection in optical networks," in *2021 Optical Fiber Communications Conference and Exhibition (OFC),* (2021), pp. 1–3.
17. L. Z. Khan, J. Pedro, N. Costa, L. De Marinis, A. Napoli, and N. Sambo, "Data augmentation to improve performance of neural networks for failure management in optical networks," J. Opt. Commun. Netw. **15**, 57–67 (2023).
18. C. Xing, C. Zhang, B. Ye, D. Wang, Y. Jia, J. Li, and M. Zhang, "Failure data augmentation for optical network equipment using time-series generative adversarial networks," in *2023 Optical Fiber Communications Conference and Exhibition (OFC),* (2023), pp. 1–3.
19. L. Z. Khan, J. Pedro, N. Costa, A. Napoli, and N. Sambo, "Model-centric versus data-centric machine learning for soft-failure cause identification in optical networks," in *2023 European Conference on Optical Communication (ECOC),* (2023), pp. 1–4.
20. A. Sgambelluri, A. Pacini, F. Paolucci, P. Castoldi, and L. Valcarenghi, "Reliable and scalable kafka-based framework for optical network telemetry," J. Opt. Commun. Netw. **13**, E42–E52 (2021).
21. D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu, "Data-centric artificial intelligence: A survey," (2023).
22. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV),* (2017), pp. 2999–3007.
23. S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, "A survey of data augmentation approaches for NLP," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021,* (Association for Computational Linguistics, Online, 2021), pp. 968–988.
24. S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models," IEEE Transactions on Pattern Analysis & Mach. Intell. **44**, 7327–7347 (2022).
25. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res. **16**, 321–357 (2002).
26. P. Cunningham and S. J. Delany, "K-nearest neighbour classifiers - a tutorial," ACM Comput. Surv. **54** (2021).
27. E. F. Swana, W. Doorsamy, and P. Bokoro, "Tomek link and smote approaches for machine fault classification with an imbalanced dataset," Sensors. **22** (2022).
28. I. Tomek, "Two modifications of cnn," IEEE Transactions on Syst. Man, Cybern. **SMC-6**, 769–772 (1976).
29. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems,* vol. 27 Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds. (Curran Associates, Inc., 2014).
30. D. Saxena and J. Cao, "Generative adversarial networks (gans): Challenges, solutions, and future directions," CoRR. **abs/2005.00065** (2020).
31. L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, *Modeling Tabular Data Using Conditional GAN* (Curran Associates Inc., Red Hook, NY, USA, 2019).
32. M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning,* vol. 70 of *Proceedings of Machine Learning Research* D. Precup and Y. W. Teh, eds. (PMLR, 2017), pp. 214–223.
33. M. Aslam, "Cochran's q test for analyzing categorical data under uncertainty," J. Big Data **10**, 147 (2023).
34. S. Raschka, "mlxtend - cochrans q," https://rasbt.github.io/mlxtend/user_guide/evaluate/cochrans_q/. (Accessed: 12/01/2024).