Data Augmentation to Improve Performance of Neural Networks for Failure Management in Optical Networks

LAREB ZAR KHAN [1,*], JOÃO PEDRO [2,3], NELSON COSTA [2], LORENZO DE MARINIS [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 January 16, 2023

Despite the increased exploration of machine learning (ML) techniques for the realization of autonomous optical networks, less attention has been paid to data quality, which is critical for ML performance. Failure management in optical networks using ML is constrained by the fact that some failures may occur more frequently than others, resulting in highly imbalanced datasets for the training of ML models. To address this limitation, a variational-autoencoder-based data augmentation technique has been investigated in this paper, which can be used during data preprocessing to improve data quality. The synthetic data generated by the variational autoencoder has been utilized to reduce imbalance in an experimental dataset used for training of neural networks (NNs) for failure management in optical networks. Firstly, it has been shown that with a modified training dataset, training time of NNs can be reduced. A reduction of up to 37.1% and 60.6% was achieved for failure detection and cause identification, respectively. Secondly, it has been shown that an improvement in the quality of the training dataset can reduce the computational complexity of NNs during inference phase. As determined analytically, almost 68% reduction in computational complexity has been achieved for the neural network used for failure cause identification. Finally, data augmentation has been shown to achieve an improvement in classification accuracy. This work demonstrates an improvement of up to 7.32%.

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

1. INTRODUCTION

Machine learning (ML) is expected to offer great potential to optimize the management of optical networks [1–4]. Therefore, its applications in optical networks are being extensively studied, including optical network failure management (ONFM) as a key use case [5, 6]. In [7], a ML technique based on neural network (NN) and shape-based clustering was proposed to proactively detect and locate faults along with their probable root causes. Wang et al., in [8], proposed a ML-based performance monitoring and failure prediction technique for optical networks that uses support vector machine and double exponential smoothing. A probabilistic ML algorithm based on Bayesian Networks was investigated in [9] to localize and identify the most probable cause of a failure. The research presented in [10–13] also focuses on ML-based failure detection and/or cause identification. In all of these works, numerous ML techniques have been investigated for ONFM to achieve improved performance; however, the emphasis on data quality, which acts as a fuel for any ML model, appears to be missing.

The good quality of the training data is indispensable for achieving the optimal performance of a ML model [14, 15]. In the case of neural networks (NNs), increasing their complexity to some extent (i.e., adding more hidden layers and/or neurons) improves performance [16], implying that poor data quality can be compensated by increasing the complexity of a NN. However, as data quality can be improved in software, the increased implementation complexity of NNs in hardware can be avoided. One of the ways to improve data quality is to introduce class balance in the dataset. In a balanced dataset, none of the class is underrepresented. But, when dealing with ONFM and, in particular, with failure cause identification, some failures are more common than others. This may result in an unequal distribution of failure observations and, thus, in imbalanced datasets. Training ML models with highly imbalanced datasets may result in a bias toward the majority (more common failure) class [17], affecting overall performance in terms of accuracy and/or training time.

There are two possible approaches to reduce imbalance in a dataset: (i) under-sampling of the majority failure class(es) or (ii)

over-sampling of the minority failure class(es). Under-sampling of majority classes can result in information loss [18] and therefore, it may not necessarily be as effective as oversampling. On the other hand, there are multiple ways to perform oversampling of minority classes. Blind over-sampling (i.e., creating duplicate samples) is one approach that can be utilized, but, as no new information is added, this approach is more likely to cause overfitting because, instead of generalizing the data, the ML model may memorize it. Hence, this oversampling approach is also not very effective. There are other methods, such as the synthetic minority oversampling technique (SMOTE) [19] and its variants [20–22] that generate new samples, but these oversampling approaches have limited performance when samples from the minority class are in extremely small proportion [23]. This process of generating new synthetic samples of data from existing data is commonly referred to as *data augmentation*. To address the issue of class balance and/or insufficient data for training of ML models employed for ONFM, few works utilizing SMOTE [24] and generative and adversarial networks (GANs) [25] for data augmentation already exist in the literature. The training of GANs suffers from instability and modal collapse due to its zero-sum game working principle [26] i.e., improvement in performance of one component of model at the expense of the other. On the contrary, data augmentation techniques based on variational autoencoder (VAE) have been shown to require comparatively less training time [27]. Moreover, VAE generates synthetic samples after determining the underlying patterns of minority classes, so it outperforms techniques like SMOTE that rely on linear interpolation for new samples generation and suffers when there are very few minority class samples in the dataset. In general, data augmentation applied to ONFM still needs to be studied in depth.

This paper is an extension of [28] and presents a comprehensive analysis on exploiting a VAE-based data augmentation technique to reduce the imbalance in actual experimental datasets (hereinafter referred to as *real dataset*) in the context of ONFM. This work aims to provide a new perspective on how improving the quality of training data improves the performance of ML models, specifically NNs for ONFM, in the form of higher classification accuracy/F1-score and/or reduced computational complexity of deployed NNs for inference and/or reduced training time. The benefits for both the training and inference phases have been investigated, and depending on the requirements, any of these benefits can be availed. For our analysis, we considered two typical use-cases within ONFM i.e., soft-failure detection and cause identification, and compared the performance of NNs when trained with the real dataset or with *modified dataset* (i.e., fully or partially balanced dataset containing synthetic and real data). During the training phase, the modified dataset results show a significant performance improvement in terms of training time reduction (i.e., 37.13% for failure detection and 60.66% for cause identification). For the scenarios where long offline NN training is affordable and a reduction in training time is not particularly desired, we demonstrated that for similar training time, a modified dataset can enable a less complex NN (65.89% fewer trainable parameters in the considered case) to achieve similar performance to a complex NN trained with real dataset for failure cause identification. If we deploy this less complex NN for failure cause identification, we can reduce multiplication or bit operations by up to 68% for each inference. We also showed for another scenario that improving training dataset quality can provide a significant improvement in classification accuracy (up to 7.32%) and F1-scores (up to 18%).

The remainder of this paper is structured as follows. Section 2 covers the necessary details about the experimental setup and the data acquired from it. Section 3 focuses on the approach we proposed for data augmentation, followed by Section 4 which presents an analysis of obtained results for soft-failure detection and cause identification. In Section 5, analysis of how data augmentation can reduce the computational complexity of NNs is presented. Section 6 discusses another advantage of data augmentation, which is improved classification accuracy, and Section 7 assesses the performance of a balanced training dataset under different sampling scenarios. Finally, our overall findings are summarized in the conclusion.

2. EXPERIMENTAL SETUP AND DATA ACQUISITION

In our investigation, we considered using the experimental testbed shown in Figure 1 for data collection. To realize the transmitter (Tx) and receiver (Rx), commercial transponders were used. The testbed consisted of four single mode fiber spans, denoted by S_1, S_2, S_3, and S_4, each 80 km long. Erbium doped fiber amplifiers (EDFAs), marked as A_1, A_2, A_3, and A_4, were used at the end of each span to compensate for power attenuation along the fiber. A wavelength selective switch (WSS) was placed at the end of S_2 to artificially produce different soft-failures in the system.

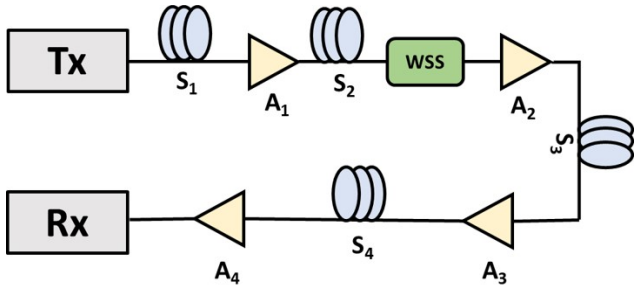

Fig. 1. Experimental testbed setup

The considered soft-failures were filter tightening (i.e., reduction of filter bandwidth), filter shift (i.e., shift in the central frequency of the filter), their combined effect (i.e., filter tightening along with filter shift), unwanted attenuation, and its combined effect with filter tightening. In case of no soft-failure i.e., normal operation, the central frequency (f_c) of the WSS was set to 192.3 THz and the attenuation and bandwidth of the WSS were set to 0 dB and 37.5 GHz, respectively. Table 1 summarizes the configuration details for all five considered soft-failures.

Assigned Label	Soft-Failure	Filter Bandwith (GHz)	Attenuation (dB)	Central Frequency (THz)
F_0	Filter Tightening	26	0	192.3
F_1	Attenuation	37.5	6	192.3
F_2	Filter Tightening + Attenuation	26	6	192.3
F_3	Filter Tightening + Filter Shift	26	0	192.32
F_4	Filter Shift	37.5	0	192.32

Table 1. Considered Soft-Failures

Through the Kafka-based telemetry system [29], input-output power levels at each amplifier, bit error rate (BER) and optical signal-to-noise ratio (OSNR) at Rx, were extracted from the

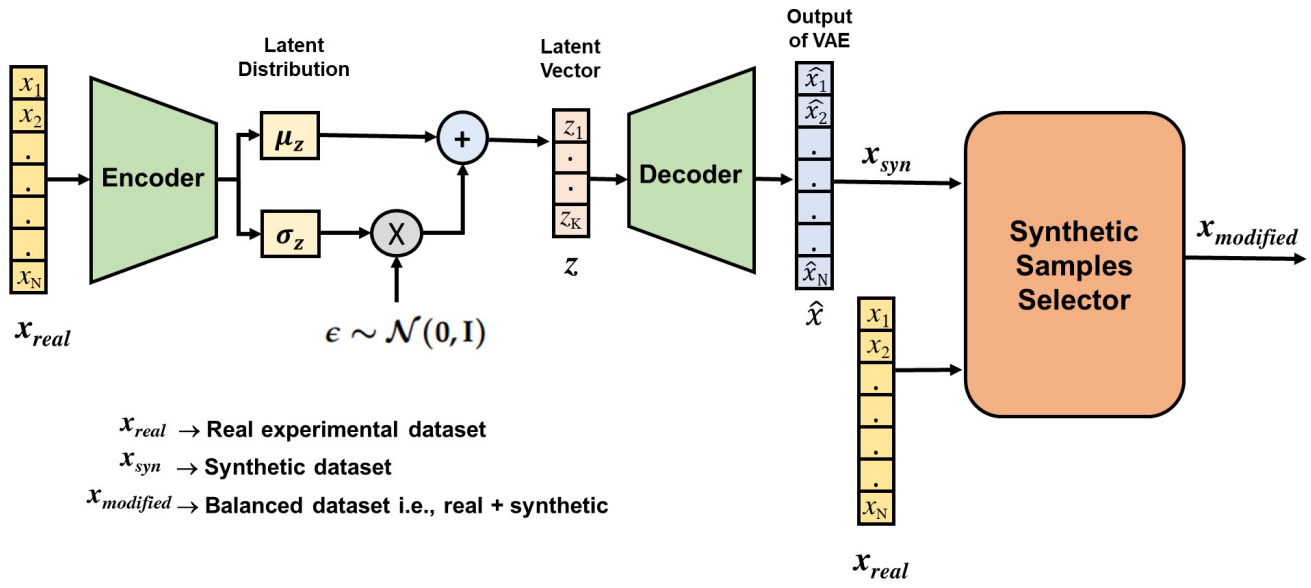

Fig. 2. Proposed approach for data augmentation (variational autoencoder followed by a synthetic samples selector)

testbed. With a sampling interval of approximately 4 seconds, we collected 10000 samples for normal operation and 5000 samples for each soft-failure, and duplicate samples were removed, resulting in an unequal distribution of failure samples within the dataset. The resultant real training dataset had 1232 samples for normal operation and 954 samples for failure operation. The next section provides more details about the distribution of failure samples among soft-failure classes in the training dataset. This acquired dataset from testbed was used in two different scenarios: N = 3, in which we considered BER, OSNR, and the input power at A_2 as features for ML, and the other N = 2, in which we relied only on coherent receiver parameters, namely BER and OSNR. Thus, the N = 2 scenario assumes end-to-end monitoring available only, whereas the N = 3 scenario also considers power monitoring. The consideration of these scenarios allowed us to demonstrate different benefits of using data augmentation.

3. PROPOSED APPROACH FOR DATA AUGMENTATION

For data augmentation, we used a VAE-based approach, illustrated in Figure 2. A VAE was followed by a *synthetic samples selector (SSS)*. The following two subsections will describe the functioning mechanism of the two components of our proposed technique and how they complement each other.

A. Variational Autoencoder (VAE)

A VAE, like a classical autoencoder (AE), contains an encoder and a decoder (both of which are NNs), but a VAE can be considered as the generalization of an AE to a generative model. VAE gets its generative capabilities because its encoder is designed to enforce a normal probability distribution on the attributes of the latent space [30] (i.e., the output of the encoder). During training, VAE determines the optimal distribution parameters (μ_z and σ_z) for each latent space attribute by minimizing the overall VAE loss (l_{VAE}), given by Eq. 1.

$$l_{VAE} = l_{reconstruction} + l_{KL-Divergence} \tag{1}$$

The l_{VAE} has two components, (i) reconstruction loss ($l_{reconstruction}$) and (ii) KL-Divergence loss ($l_{KL-Divergence}$). The minimization of $l_{reconstruction}$ pushes the decoder to approximate the inverse mapping of the encoder so that the reconstructed input ($\hat{x}$) can be very close to the actual input (x). The reconstruction loss is nothing but the mean squared error (MSE) between x and $\hat{x}$. If both x and $\hat{x} \in \mathbb{R}^N$, then

$$l_{reconstruction} = \frac{1}{N}\sum_{i=0}^{N-1}(x_i - \hat{x}_i)^2 \tag{2}$$

On the other hand, minimization of $l_{KL-Divergence}$ enforces a normal distribution $\mathcal{N}(\mathbf{0}, \mathbb{I}_\mathbf{K})$ on the latent space, where K denotes the dimensions of the latent space. The $l_{KL-Divergence}$ is basically the relative entropy (i.e., a measure of distance between two distributions) between the standard normal distribution i.e., $\mathcal{N}(\mathbf{0}, \mathbb{I}_\mathbf{K})$ and $\mathbf{Z}$, which is described by a Gaussian distribution with mean (μ_z) and standard deviation (σ_z). The resultant entropy can be expressed as in Eq. 3 and its detailed derivation can be found in [31].

$$l_{KL-Divergence} = \frac{1}{2}\sum_{j=0}^{K-1}(\sigma_{z_j}^2 - 1 - log[\sigma_{z_j}^2] + \mu_{z_j}^2) \tag{3}$$

The learned latent distribution of each attribute needs to be sampled randomly to obtain a latent vector. However, if this random sampling is performed within the NN, then backpropagation of the error is not possible. This is because the relationship of each parameter in the NN with respect to the final output error cannot be determined because of this randomness, making training of NN impractical. To overcome this limitation, a reparameterization trick [30, 32] is used, given by Eq. 4, which lets randomness be fed as input to the model i.e., using $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, making backpropagation of error possible during training.

$$z = \mu_z + \sigma_z \odot \epsilon \tag{4}$$

Following latent space sampling, the VAE decoder takes the sampled latent vector as input and produces an output that should ideally match the encoder's input. But, because of VAE's imperfect training, the decoder's output differs slightly from the real input; nevertheless, it retains the same underlying pattern

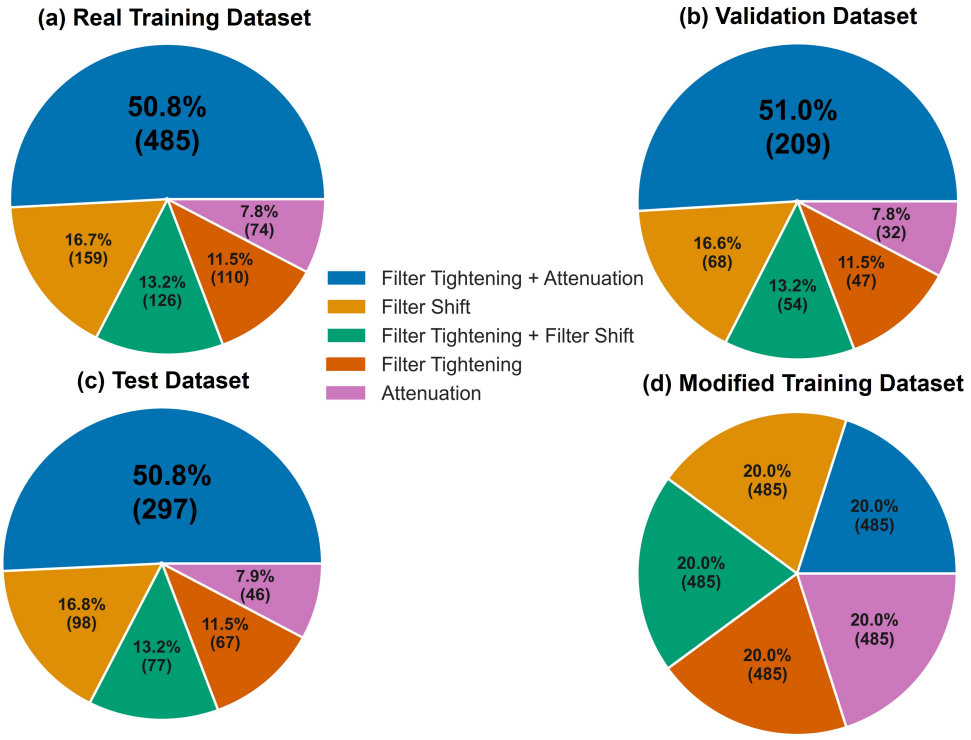

Fig. 3. Distribution of considered soft-failures within (a) real training, (b) validation, (c) test and (d) modified training dataset

as the encoder's input. Thus, the reconstructed input is a synthetic sample generated by VAE, and by randomly sampling the latent space multiple times, we can generate a large number of synthetic samples, hence achieving data augmentation.

For our analysis, the encoder had the same number of inputs as the number of considered features (i.e., N = 2 in one case and 3 in the other), the number of attributes in the latent space (i.e., K) was set to 2, and the encoder had one hidden layer with 4 neurons. The decoder had 2 neurons in its input layer, 4 in its only hidden layer, and the same number of neurons in its output layer as the encoder's input. The sigmoid activation function was used for the output layer of the VAE and where required, ReLU was used as an activation function for the intermediate layers. With Adam as the optimizer and a batch size of 8, the learning rate for VAE training was 0.0079. It should be noted that only VAE is the trainable component of this proposed data augmentation approach, and its encoder and decoder are trained together as a single entity. However, after training, we used it in a block-wise manner, feeding all training data to the encoder, which generated a different latent vector for each observation in the dataset due to random sampling, and then these latent vectors were fed to the decoder, which decoded them. This process was repeated 10 times, increasing the available data by a factor of 10. The synthetic samples selector (SSS) was then used to select the required number of synthetic samples for each soft-failure class in order to obtain a modified dataset.

B. Synthetic Samples Selector (SSS)

We proposed SSS to select "good" synthetic samples from all the samples generated by VAE. The SSS can be considered as a module that selects synthetic samples of any particular under-represented soft-failure class based on their Euclidean distance from the mean of that same class in the real dataset. This selection criteria was used considering that synthetic samples that lie close to any particular under-represented soft-failure class in a real dataset in N-dimensional space are likely to have similar characteristics to that particular class.

To mathematically explain the working principle of SSS for N = 3 scenario, let Γ, Ω and $\Gamma_{majority}$ to be the sets of all soft-failure classes, non-majority soft-failure classes, and majority soft-failure classes in the real dataset, respectively. As we considered five different soft-failures with labels ranging from F_0 to F_4, so $\Gamma = \{F_0, F_1, F_2, F_3, F_4\}$ and in this case, majority soft-failure class is F_2 , therefore $\Gamma_{majority} = \{F_2\}$, resulting in $\Omega = \{F_0, F_1, F_3, F_4\}$. If s denotes each under-represented soft-failure class in Ω, then $\forall s \in \Omega$:

$$\boldsymbol{\varphi}^{(s)}_{\boldsymbol{reference}} = (\varphi^{(s)}_{ref,P_{in\_A2}}, \varphi^{(s)}_{ref,BER}, \varphi^{(s)}_{ref,OSNR}) \tag{5}$$

$$= \frac{1}{N^{(s)}_{real}}\{\sum_{n=0}^{N^{(s)}_{real}-1}(x^{(s),(n)}_{real,P_{in\_A2}}, x^{(s),(n)}_{real,BER}, x^{(s),(n)}_{real,OSNR})\}$$

where $\boldsymbol{\varphi}^{(s)}_{\boldsymbol{reference}}$ represents the reference/mean and $N^{(s)}_{real}$ denotes the number of samples of soft-failure class s in the real dataset. The Euclidean distance of each synthetic sample belonging to class s is computed from the $\boldsymbol{\varphi}^{(s)}_{\boldsymbol{reference}}$, which can be expressed as $\forall k$ where $k = 0, 1, \ldots, N^{(s)}_{syn} - 1$, compute

$$d^{(s),(k)} = \sqrt{\sum_{j=0}^{N-1}(\boldsymbol{x}^{(s),(k)}_{\boldsymbol{syn},(j)} - \boldsymbol{\varphi}^{(s)}_{\boldsymbol{reference},(j)})^2} \tag{6}$$

Once the Euclidean distances are computed for all the synthetic samples of soft-failure class s, then first $N^{(s)}_{syn,selected}$ samples corresponding to the ascending order of the computed distances are selected where

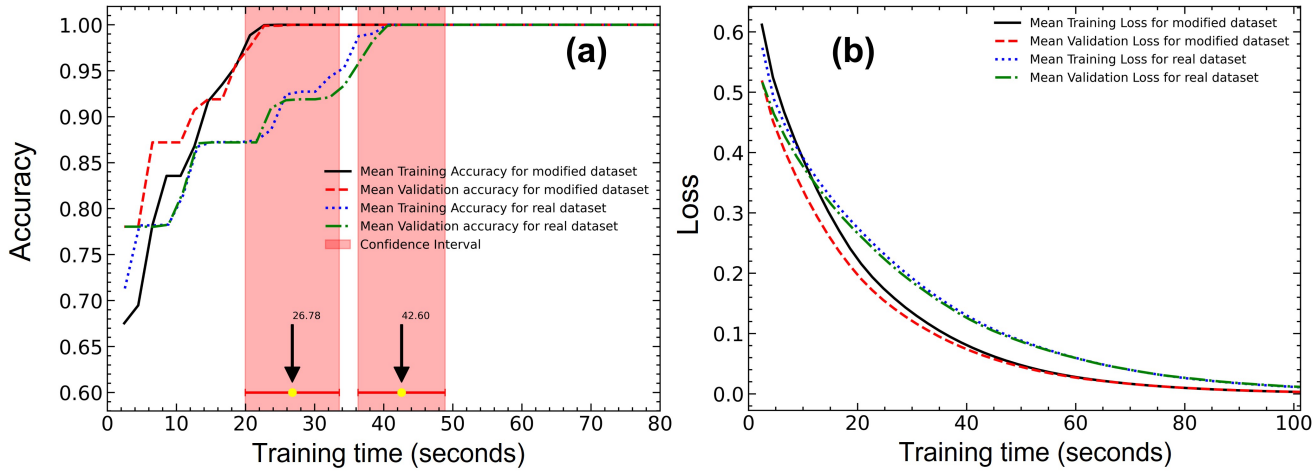

Fig. 4. Soft-failure detection performance evaluation vs. training time (a) model accuracy, (b) model loss

$$N_{syn,selected}^{(s)} = \frac{N_{real}^{(\Gamma_{majority})}}{m} - N_{real}^{(s)} \quad (7)$$

m controls the number of synthetic samples that are required to be added in the imbalanced dataset to achieve balance, with m = 1 corresponding to a perfect balance in all soft-failure classes and m = 2 corresponding to oversampling of minority classes to the point where they can have a sample count equal to 50% of the samples in the majority class. m can have other values as well, but for this work, we only considered m = 1 and 2. After selecting the required number of synthetic samples of soft-failure class s, selected synthetic samples ($\mathbf{\Psi}_{syn,selected}^{(s)}$) are combined with the real data samples ($\boldsymbol{x}_{real}^{(s)}$) of that particular class, resulting in

$$\boldsymbol{x}_{modified}^{(s)} = \mathbf{\Psi}_{syn,selected}^{(s)} \cup \boldsymbol{x}_{real}^{(s)} \quad (8)$$

The resultant modified dataset with equal representation of all soft-failure classes can be represented as

$$\boldsymbol{x}_{modified} = \boldsymbol{x}_{modified}^{(F_0)} \cup \boldsymbol{x}_{modified}^{(F_1)} \cup \boldsymbol{x}_{real}^{(\Gamma_{majority})} \cup \boldsymbol{x}_{modified}^{(F_3)} \cup \boldsymbol{x}_{modified}^{(F_4)} \quad (9)$$

It should be noted that for our analysis, only the training dataset was augmented. Throughout our analysis, performance is compared between the real and modified training datasets on the NNs employed for ONFM. The validation and test datasets were kept the same for the evaluation of trained NNs. As the validation dataset is used during training to tune the NN's hyperparameters so that it can make accurate predictions in a real-world setting, the use of synthetic data for validation may not necessarily depict the actual performance of the NNs. The same is true for the test dataset as well, because it is used to provide an unbiased estimate of the final trained NN performance. The split between the train and test datasets was 70-30%, and the same split was used for the train and validation datasets.

Figure 3 illustrates how different soft-failures are represented within (a) real training, (b) validation, (c) test, and (d) modified (m = 1 scenario) training datasets. The unequal distribution of soft-failure data samples in the test and validation datasets also suggests that these datasets were not modified. Moreover, it is worth-mentioning again that this particular distribution within the real dataset was achieved after removing duplicates from the data acquired from the testbed. The resulting dataset was appropriate for our analysis because different soft-failures were represented unequally, which is a typical case in optical networks since some failures are more common than others. However, this does not necessarily mean that the most-represented failures for our analysis are common in large-scale optical networks as well.

4. RESULTS

Within ONFM, we compared the performance of real and modified training datasets on NNs for soft-failure detection and cause identification. The metrics used for this performance comparison are loss and accuracy on training, validation, and test datasets, as well as training time to achieve maximum accuracy on validation dataset. In general, the loss value (determined by a cost function) indicates how a model performs after each epoch (i.e., one complete pass through the whole training dataset), and the model's performance improves by minimizing this loss. On the other hand, accuracy is a more interpretable metric as it provides the ratio of correct classifications to the total number of classifications made by the ML model. The training time to achieve specific validation accuracy has been preferred here as a metric over the number of epochs because the time taken for an epoch depends on the size of the training dataset. Since a modified dataset is achieved by oversampling, utilizing synthetic samples generated by VAE, its size is greater than the size of the real dataset. Hence, training time as a metric depicts actual performance, which is not the case with only the number of epochs as a performance metric. Furthermore, since ML training time is a random variable, we trained NNs for soft-failure detection and cause identification 100 times (with each training iteration itself comprised of 100 epochs) and considered their averaged/mean results with 95% confidence interval. It should be noted that the results presented in this section correspond to m = 1 (i.e., perfect class balance) and N = 3 (i.e., three input features: BER, OSNR, and input power at A_2 for the NN) scenario.

A. Soft-Failure Detection

For failure detection, we used a NN (size: input layer (I) × hidden layer-1 (h_1) × output layer (O) = 3 × 2 × 1), hereinafter referred to as NN_0. Adam was the optimizer used for training, with a learning rate of 0.0008 and a batch size of 8. The activation function for hidden layer neurons was ReLU, while the

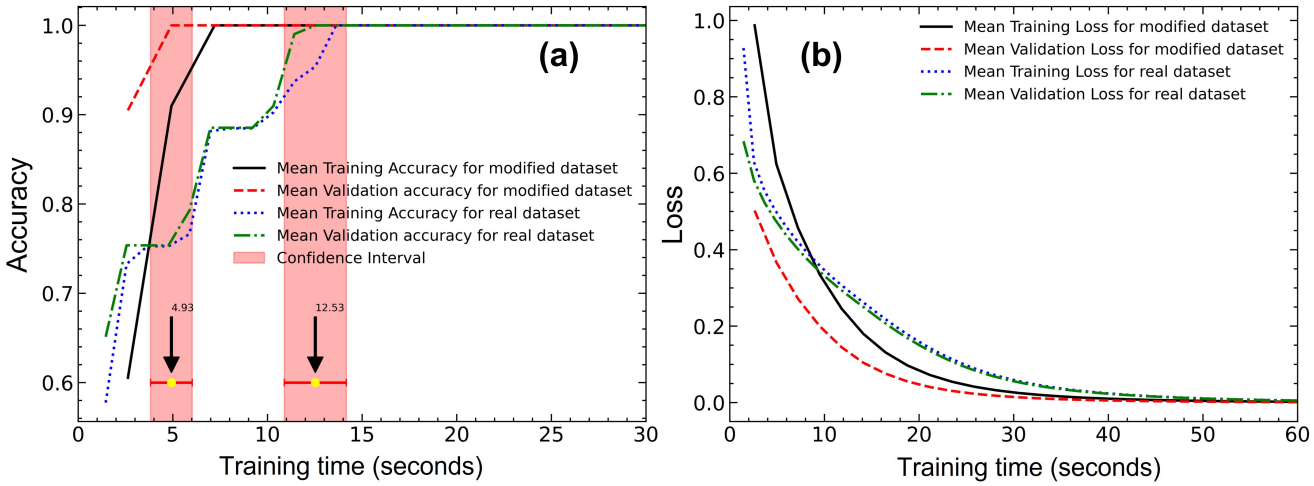

Fig. 5. Soft-failure cause identification performance evaluation vs. training time (a) model accuracy (b) model loss

activation function for output layer neurons was sigmoid. As shown in Figure 4(a), NN_0 achieved a training and validation accuracy of 1.0 (i.e., 100%) with the real and modified training datasets. This is because our dataset was separable (i.e., non-overlapping) in the considered scenario, allowing NN_0 to accurately detect failures. However, it should be noted that with the modified training dataset, on average 100% validation accuracy was achieved only in 26.78 seconds using our system Intel Xeon W-2255 CPU @ 3.70GHz with NVIDIA RTX A5000 GPU. But, with the real dataset, the same accuracy was achieved after 42.60 seconds, implying a 37.13% reduction in training time with the modified dataset. This reduction in training time is related to the degree of imbalance between normal observations (1232) and failure observations (954), and in this case, the imbalance was removed by adding the required number of randomly selected synthetic failure samples picked by SSS. As shown, the use of a balanced training dataset significantly improved performance.

If we evaluate the loss performance, we can observe that with modified dataset, the decrease in loss was sharper as compared to the real dataset, as showed in Figure 4(b) which again suggests the superior performance of modified dataset. It should be noted that no regularization was applied to NN_0, which allowed us to observe that the modified dataset did not result in any overfitting of NN_0, as indicated by Figure 4 as well. Moreover, all the hyperparameters were kept the same for the NN_0. The only difference was in the training dataset. Following this training, both cases attained 100% accuracy on the same test dataset, indicating that the model was perfectly trained using both datasets but this training was achieved in a shorter time with a modified dataset.

B. Soft-Failure Cause Identification

After the failure detection, the next step is to identify the cause of that failure which is a more complex problem than failure detection. This multi-class classification was accomplished using a NN (hereinafter referred to as NN_1) with two hidden layers (one with 10 neurons and the other with 8 neurons), both of which used *tanh* as a non-linear activation function. There were 5 neurons in the output layer because we considered five different soft-failures. *Softmax* was used as an activation function for output layer neurons, which provides the probability of each considered soft-failure corresponding to the given input. The optimizer used for the training was Adam with learning rate of 0.0005 and batch size of 8.

Figure 5 shows a comparison of the results obtained for soft-failure cause identification using the real and modified training datasets. Similar performance as for failure detection was achieved for soft-failure cause identification, but this time modified dataset led to a further improvement: a reduction of the training time by 60.6% when compared to using the real dataset. This is due to a greater class imbalance in this case than in the failure detection case. This improvement in performance with modified training dataset is intuitive because to achieve 100% accuracy, a NN must see enough samples from each soft-failure class during training to determine the underlying pattern of that class. But, with an imbalanced training dataset, NN sees an unequal number of samples in each epoch, requiring more epochs/time to see enough samples from the under-represented soft-failure classes. A balanced training dataset, on the other hand, provides the NN with enough samples of all soft-failure classes in fewer epochs, allowing it to attain 100% accuracy on unaugmented and imbalanced validation dataset.

5. ANALYSIS OF NEURAL NETWORK COMPLEXITY REDUCTION DUE TO DATA AUGMENTATION

We extended our investigation to explore how an improvement in training data quality can provide some gains in terms of NN computational complexity reduction during inference phase. Since failure cause identification is a relatively complex problem than failure detection, we considered failure cause identification case only. We compared the performance of a NN (NN_1 with size $I \times h_1 \times h_2 \times O = 3 \times 10 \times 8 \times 5$, the one used in Section 4B) trained with real dataset and a lower-complexity NN (NN_2, with size $I \times h_1 \times O = 3 \times 6 \times 5$) trained with modified dataset. Except for the number of hidden layers and neurons in each layer, NN_2 had the same set of hyperparameters (i.e., learning rate of 0.0005 with Adam as an optimizer and batch size of 8) as NN_1. In NN_1 and NN_2, there were 173 and 59 trainable parameters, respectively.

As can be seen in Figure 6, NN_1 took 12.53 seconds whereas NN_2 took 12.44 seconds to achieve 100% accuracy on validation dataset. Although the training time is similar, the gain is achieved in terms of complexity reduction since NN_2 has 65.89% less parameters as compared to NN_1. To translate

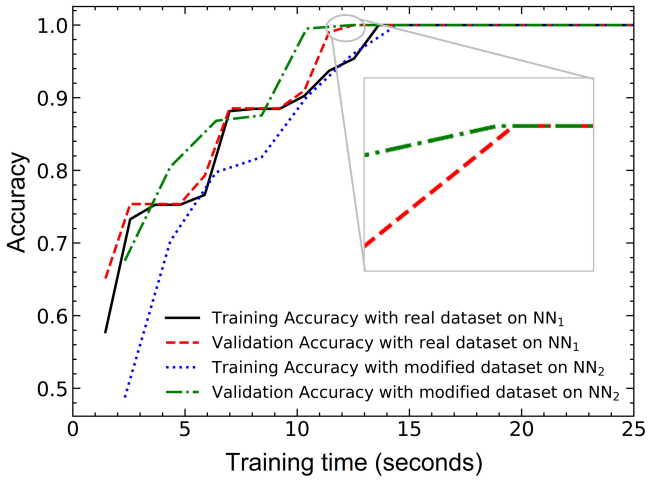

Fig. 6. Training phase performance of NN_1 and NN_2 trained on real and modified datasets, respectively

this complexity reduction in terms of computational cost on hardware for each inference of trained NN, we considered following two metrics.

(i) Number of real multiplications (NRM)
(ii) Number of bit-operations (NBOP)

NRM estimates the computational complexity in terms of the number of multipliers required, while neglecting adders. It is because the addition is low-cost in terms of hardware resource utilization, while the multiplier is typically the slowest element in the system [33] and consumes a significant chip area [34]. NBOP metric, on the other hand, is more comprehensive as it takes into account multiplication and addition operations while estimating computational complexity. It also tells us how changing bitwidth (i.e., precision/number of bits required to represent a number) impacts the computational complexity of NNs.

For estimating NRM and NBOP, consider that the output ($\boldsymbol{y}$) of a dense (i.e., fully-connected) layer containing n_n number of neurons, each with n_i inputs, can be given as

$$\boldsymbol{y} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{10}$$

where $\boldsymbol{W}$ is the weight matrix with dimensions $\mathbb{R}^{n_n \times n_i}$, $\boldsymbol{x}$ is input vector of length n_i, $\boldsymbol{b}$ is a bias vector of length n_n and $\sigma(.)$ denotes the non-linear activation function. For this analysis, it has been assumed that the non-linear operation is performed using a look-up table (LUT), which is a common practice [35]. Regarding multipliers, from Eq. 10, it can be inferred that total $n_n \times n_i$ number of multiplications are performed within each dense layer, hence

$$NRM = n_n n_i \tag{11}$$

As all layers in NN_1 and NN_2 are dense, so using Eq. 11 we can estimate NRM for the whole NN_1 and NN_2. Note that the input layer (I) simply passes the given input to the intermediate layers for further processing, so no multiplication is performed. Thus, NN_1 requires a total of 150 (i.e., 30 at h_1, 80 at h_2 and 40 at O) multiplications for a given input. On the other hand, NN_2 requires 48 multiplications in total, i.e., 18 at h_1 and 30 at O. These numbers suggest that 68% lesser multiplication operations need to be performed on hardware, which is a huge gain considering that training data quality has been improved in software.

To analytically evaluate the computational complexity reduction in terms of other considered metric, i.e., NBOP, we used Eq. 12, proposed in [36] which is adapted from [37] to estimate the number of bit operations required to be performed in a single dense layer of a NN.

$$NBOP \approx n_n n_i [b_w b_i + b_w + b_i + log_2(n_i)] \tag{12}$$

where b_w denotes the bitwidth of weights ($\boldsymbol{W}$) and b_i denotes the bitwidth of input ($\boldsymbol{x}$). If we fix b_w and b_i = 8, then NN_1 needs to perform around 12434 (i.e., 2448 at h_1, 6666 at h_2 and 3320 at O) bit operations per inference. On the other hand, for NN_2, around 3947 (i.e., 1469 at h_1 and 2478 at O) bit operations are required to be performed for a given input, suggesting 68.2% reduction in total number of bit operations. For $b_w = b_i$ = 16, NN_1 needs to perform around 43634 (i.e., 8688 at h_1, 23306 at h_2 and 11640 at O) bit operations, while NN_2 needs to perform 13931 (i.e., 5213 at h_1 and 8718 at O) bit operations. In this case, 68.07% reduction in bit operations can be achieved for each inference. Similarly, for $b_w = b_i$ = 32, it can be shown using Eq. 12 that complexity reduction of NN achieved using data augmentation corresponds to 68.02% less bit operations. Therefore, it is clear that improving the quality of training data can reduce the computational complexity of NNs during inference phase.

6. DATA AUGMENTATION FOR THE IMPROVEMENT OF CLASSIFICATION ACCURACY

In order to further investigate the advantages of data augmentation in ONFM, we considered N = 2 scenario where we relied only on coherent receiver parameters i.e., BER and OSNR. Figure 7 shows the resultant dataset, and as can be seen that filter tightening (F_0), attenuation (F_1), and filter tightening + filter shift (F_3) all have OSNR and BER values in the similar range, making the dataset inseparable. It should be noted that only one feature (i.e., input power at A_2) from the previously considered dataset was removed, which had no effect on the number of samples for each soft-failure class.

This inseparable dataset was processed in the same way as the separable dataset discussed in previous sections of this paper. Only the training dataset was augmented with VAE, and the

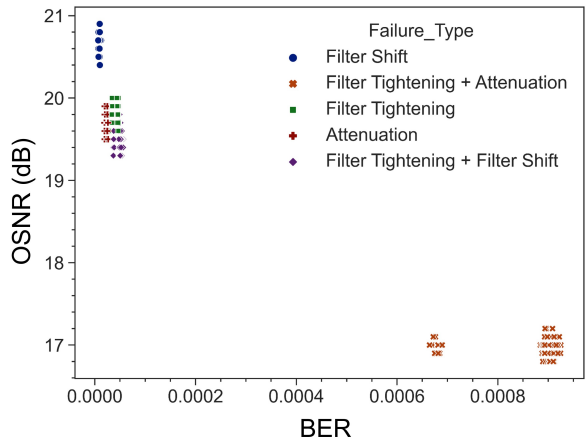

Fig. 7. Dataset without input power at A_2 feature (N = 2 scenario)

required number of synthetic samples were selected with SSS to create a modified dataset. The VAE encoder had 2 neurons in the input layer, 4 in the only hidden layer, and 2 latent dimensions; the decoder was a mirror copy of the encoder. With Adam as the optimizer, the learning rate for VAE was 10^{-4} and batch size was 8. To compare the performance of these inseparable real and modified training datasets, we only considered a soft-failure cause identification case, since for failure detection (being a binary classification), dataset is usually separable because OSNR and BER values are considerably different during failure and normal operation. In this case, the NN used for soft-failure cause identification (hereinafter referred to as NN_3) had the size $I \times h_1 \times h_2 \times O = 2 \times 20 \times 10 \times 5$ with Adam as the optimizer during training, a learning rate of 10^{-4}, and a batch size of 8. Figure 8(a) and 8(c) show the averaged results from 100 iterations with each of 100 epochs. Some observations that can be made from the obtained results are as follows:

i) With the modified training dataset, as suggested by Figure 8(a), the NN can be trained better to provide higher accuracy (89.27% in this case) on the validation dataset as compared to when trained with the real dataset (81.95%). Considering that the classification accuracy in this case is not 100% and that it is a multi-class classification problem, F1-Score (i.e., harmonic mean of precision and recall) is also an appropriate metric for determining whether or not the modified training dataset provided any gains. Figure 8(b) evaluates the performance of NN_3 when trained with real and modified training datasets in terms of weighted-average F1-Score, which is calculated by taking the mean of all per-class F1-Scores while accounting for each class's support, and the macro-average F1-Score, which is calculated by taking the arithmetic mean of all per-class F1-Scores. These F1-Scores show how well trained NNs with real and modified training datasets performed on an unseen and unmodified test dataset. And, as indicated by the obtained results in Figure 8(a) and (b), classification performance has been improved using data augmentation. NN_3 when trained with the modified dataset attained the macro-average F1-score of 0.75, and when it was trained with the real dataset, the F1-score of 0.57 was achieved on test dataset. Similarly, a significant improvement in the weighted-average F1-score can be seen, indicating that data augmentation has the potential to improve classification accuracy for soft-failure cause identification.

ii) The highest accuracy on validation dataset that was achieved in the case of real training dataset i.e., 81.95% in 55.1 seconds was achieved in a much shorter time (29.3 seconds) with the modified training dataset (indicating 46.82% reduction in this case): which supports the claim made in Section 4 that reducing class imbalance can reduce the training time for attaining same accuracy on validation dataset.

iii) Higher validation accuracy than the training accuracy,

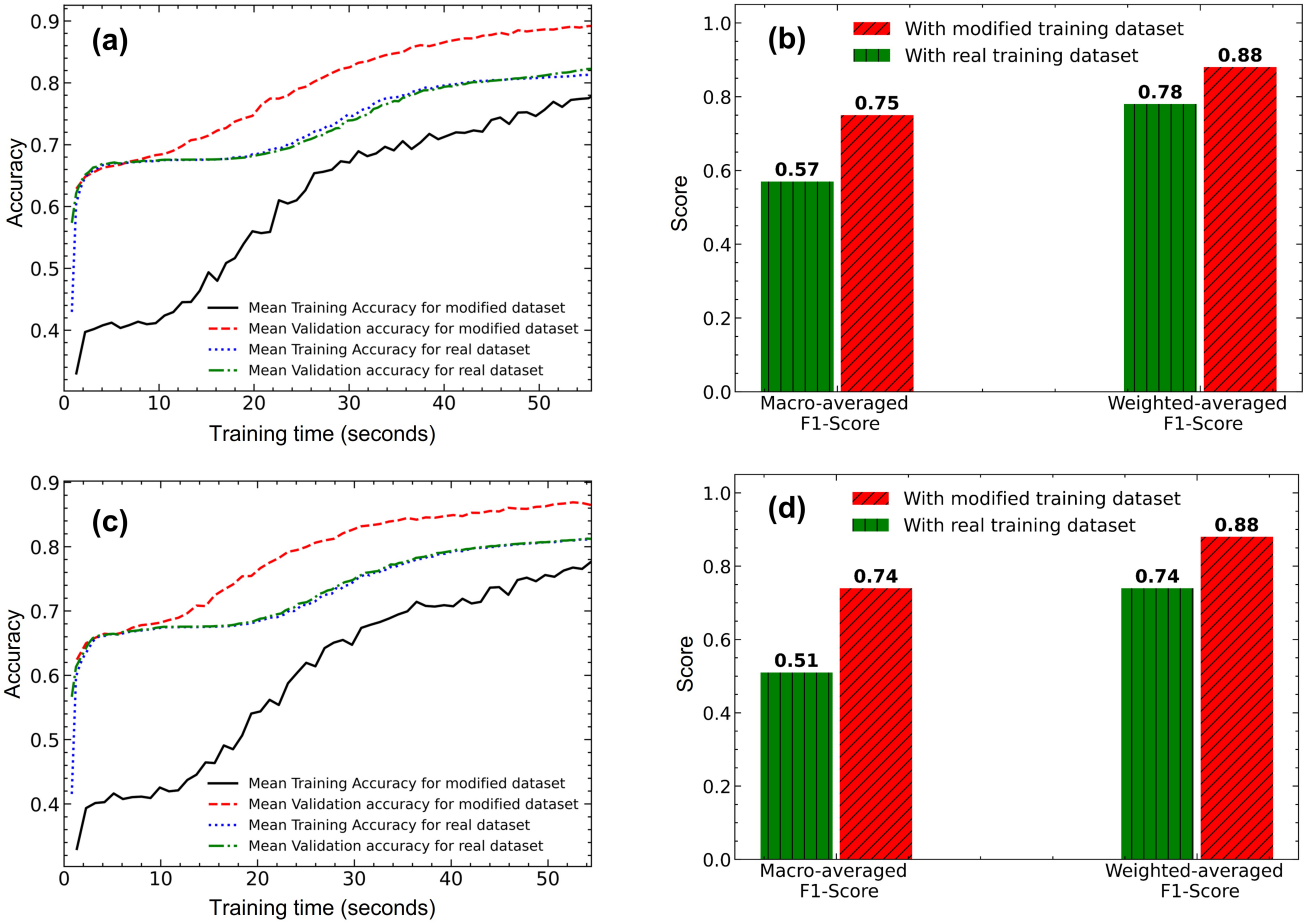

Fig. 8. Performance comparison (m = 1 scenario): For train-test-validation split corresponding to seed = 1 (a and b) and 42 (c and d)—(a and c) training and validation accuracy with real and modified training datasets, (b and d) macro- and weighted-averaged-F1-Scores on test dataset achieved by NN_3 trained with real and modified datatsets

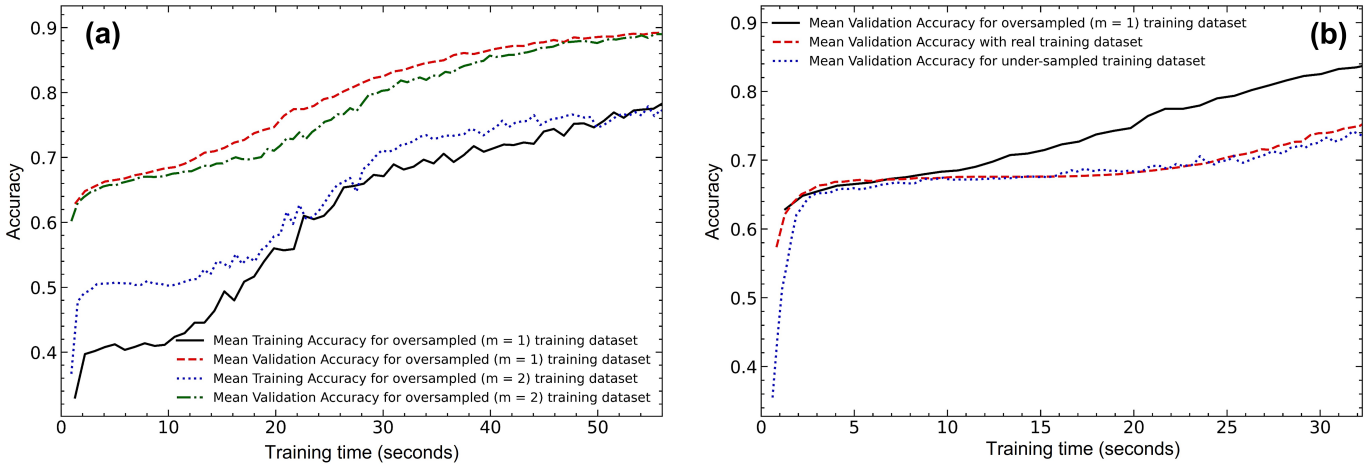

Fig. 9. Performance comparison: (a) complete balance (m = 1) and partial balance (m = 2) oversampling , (b) complete balance oversampling and under-sampling in comparison with real training dataset

which is usually observed when complications are added during training phase enabling NN (ML model in general) to obtain a better generalization of the dataset. This is typically accomplished in two ways: (1) (Dropout) Regularization, which is commonly used and thus discussed in the literature, and (2) Data Augmentation, which is rarely used and thus rarely discussed. In this work, we added samples (that are not exactly same as the real dataset but they have the same underlying pattern) to modify the experimental training dataset with data augmentation. As a result, during training, the NN also sees samples that are relatively new and different from those in the validation dataset. However, during validation, it finds simple and easy-to-classify samples, resulting in higher accuracy than on the training dataset. We believe that having higher accuracy on the validation dataset than on the training dataset has no negative effect on the performance of the NN (ML model in general) because it demonstrates that the NN is more robust and has good generalizability, and thus performs well on validation and test (i.e., unmodified/real experimental) datasets.

To ascertain the consistency of these results, we also considered a different seed for train-test-validation split. Figure 8(c) and (d) depicts the results for seed = 42, and as can be seen, the obtained results are consistent with the previous results (Figure 8(a) and (b)). In this case, the corresponding macro-average F1-score on test dataset increased from 0.51 to 0.74, and the weighted-average F1-score increased from 0.74 to 0.88, indicating again a significant improvement.

7. ANALYSIS UNDER DIFFERENT SAMPLING SCENARIOS

Until this point, the results presented assumed a complete balance (i.e., m = 1 in Eq. 7) among all soft-failure classes in the modified training dataset, which may or may not always be required. In order to investigate this, we considered m = 2 scenario as well in which under-represented soft-failure classes were over-sampled to the point where each of them had half of the total number of majority class samples i.e., $\frac{N_{real}^{(\Gamma_{majority})}}{2}$. Figure 9(a) shows the obtained results in comparison with m=1 scenario, and as can be seen, even with m=2, the similar accuracy on the validation dataset can be achieved, suggesting that perfect balance among the soft-failure classes is not always required. It also implies that, depending on the scenario and dataset, an optimal oversampling threshold for each class needs to be determined, which remains as a topic of our future research.

We also investigated how undersampling to achieve complete class balance compares to the m=1 scenario of oversampling, and the results are shown in Figure 9(b). As evident from the figure, in the case of soft-failure cause identification, undersampling was unable to achieve any gains in validation accuracy, possibly due to information loss during the undersampling. However, this does not necerssarily mean that undersampling is always ineffective. It may perform well in combination with oversampling, which we intend to investigate in the future.

8. CONCLUSION

We investigated a variational-autoencoder-based data augmentation technique to optimize the training and inference phases of neural networks (NNs) for failure management in optical networks. We analyzed the training time, computational complexity, and the classification accuracy of NNs when exploiting augmented data in comparison with real dataset. First, our analysis has shown that the training time for NNs can be significantly reduced using modified datasets (i.e., real plus augmented data). We obtained a 37.13% and a 60.6% reduction in the training time of NNs used for failure detection and cause identification, respectively. Then, our analysis suggests that good quality training dataset obtained using data augmentation can save computational resources during inference phase. In particular, for the soft-failure cause identification, similar performance in terms of training time and accuracy has been achieved using a lower-complexity NN trained with a modified dataset and a more complex NN trained with a real dataset. The difference in terms of computational complexity on hardware amounts to 68% fewer multiplication or bit operations, implying a significant complexity reduction that can be achieved if we deploy less complex NN for soft-failure cause identification. Finally, we showed that improving training dataset quality using our proposed approach can provide gains in terms of classification accuracy as we achieved up to 7.32% improvement.

FUNDING

This work has been partially funded by EU H2020 Marie Skłodowska-Curie Actions ITN project MENTOR (GA 956713) and H2020 B5G-OPEN (GA 101016663).

REFERENCES

1. R. Gu, Z. Yang, and Y. Ji, "Machine learning for intelligent optical networks: A comprehensive survey," J. Netw. Comput. Appl. **157**, 102576 (2020).
2. 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).
3. 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).
4. Y. Pointurier, "Machine learning techniques for quality of transmission estimation in optical networks," J. Opt. Commun. Netw. **13**, B60–B71 (2021).
5. F. Musumeci, C. Rottondi, G. Corani, S. Shahkarami, F. Cugini, and M. Tornatore, "A tutorial on machine learning for failure management in optical networks," J. Light. Technol. **37**, 4125–4139 (2019).
6. 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.
7. D. Rafique, T. Szyrkowiec, A. Autenrieth, and J.-P. Elbers, "Analytics-driven fault discovery and diagnosis for cognitive root cause analysis," in *2018 Optical Fiber Communications Conference and Exposition (OFC),* (2018), pp. 1–3.
8. Z. Wang, M. Zhang, D. Wang, C. Song, M. Liu, J. Li, L. Lou, and Z. Liu, "Failure prediction using machine learning and time series in optical network," Opt. Express **25**, 18553–18565 (2017).
9. 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.
10. 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).
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. F. Musumeci, V. G. Venkata, Y. Hirota, Y. Awaji, S. Xu, M. Shiraiwa, B. Mukherjee, and M. Tornatore, "Transfer learning across different lightpaths for failure-cause identification in optical networks," in *2020 European Conference on Optical Communications (ECOC),* (2020), pp. 1–4.
13. 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.
14. F. Chollet, *Deep Learning with Python* (Manning, 2017).
15. K. Maharana, S. Mondal, and B. Nemade, "A review: Data preprocessing and data augmentation techniques," Glob. Transitions Proc. **3**, 91–99 (2022). International Conference on Intelligent Engineering Approach(ICIEA-2022).
16. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016). http://www.deeplearningbook.org.
17. V. A. Fajardo, D. Findlay, C. Jaiswal, X. Yin, R. Houmanfar, H. Xie, J. Liang, X. She, and D. Emerson, "On oversampling imbalanced data with deep conditional generative models," Expert. Syst. with Appl. **169**, 114463 (2021).
18. N. Chawla, N. Japkowicz, and A. Kołcz, "Editorial: Special issue on learning from imbalanced data sets," SIGKDD Explor. **6**, 1–6 (2004).
19. K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," CoRR. **abs/1106.1813** (2011).
20. H. Han, W. Wang, and B. Mao, "Borderline-smote:a new over-sampling method in imbalanced data sets learning," (2005), pp. 878–887.
21. J. Mathew, M. Luo, C. K. Pang, and H. L. Chan, "Kernel-based smote for svm classification of imbalanced datasets," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society,* (2015), pp. 001127–001132.
22. G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," Inf. Sci. **465**, 1–20 (2018).
23. B. Yan, Y. Zhao, S. Rahman, Y. Li, X. Yu, D. Liu, Y. He, and J. Zhang, "Dirty-data-based alarm prediction in self-optimizing large-scale optical networks," Opt. Express **27**, 10631–10643 (2019).
24. B. Zhang, Y. Zhao, S. Rahman, Y. Li, and J. Zhang, "Alarm classification prediction based on cross-layer artificial intelligence interaction in self-optimized optical networks (soon)," Opt. Fiber Technol. **57**, 102251 (2020).
25. J. Li, D. Wang, S. Li, M. Zhang, C. Song, and X. Chen, "Deep learning based adaptive sequential data augmentation technique for the optical network traffic synthesis," Opt. Express **27**, 18831–18847 (2019).
26. D. Saxena and J. Cao, "Generative adversarial networks (gans): Challenges, solutions, and future directions," CoRR **abs/2005.00065** (2020).
27. M. El-Kaddoury, A. Mahmoudi, and M. M. Himmi, "Deep generative models for image generation: A practical comparison between variational autoencoders and generative adversarial networks," in *Mobile, Secure, and Programmable Networking,* É. Renault, S. Boumerdassi, C. Leghris, and S. Bouzefrane, eds. (Springer International Publishing, Cham, 2019), pp. 1–8.
28. L. Z. Khan, J. Pedro, N. Costa, A. Napoli, and N. Sambo, "Data augmentation to improve machine learning for optical network failure management," in *2022 European Conference on Optical Communication (ECOC),* (2022).
29. 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).
30. D. P. Kingma and M. Welling, "Auto-encoding variational bayes," (2013).
31. S. G. Odaibo, "Tutorial: Deriving the standard variational autoencoder (VAE) loss function," CoRR. **abs/1907.08956** (2019).
32. D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," (2014).
33. E. Jacobsen and P. Kootsookos, "Fast, accurate frequency estimators [dsp tips & tricks]," IEEE Signal Process. Mag. **24**, 123–125 (2007).
34. S. Mirzaei, A. Hosangadi, and R. Kastner, "Fpga implementation of high speed fir filters using add and shift method," in *2006 International Conference on Computer Design,* (2006), pp. 308–313.
35. T. Yang, Y. Wei, Z. Tu, H. Zeng, M. A. Kinsy, N. Zheng, and P. Ren, "Design space exploration of neural network activation function circuits," IEEE Transactions on Comput. Des. Integr. Circuits Syst. **38**, 1974–1978 (2019).
36. P. J. Freire, A. Napoli, B. Spinnler, N. Costa, S. K. Turitsyn, and J. E. Prilepsky, , "Neural networks-based equalizers for coherent optical transmission: Caveats and pitfalls," IEEE J. Sel. Top. Quantum Electron. **28**, 1–23 (2022).
37. C. Baskin, N. Liss, E. Schwartz, E. Zheltonozhskii, R. Giryes, A. M. Bronstein, and A. Mendelson, "UNIQ: Uniform Noise Injection for Non-Uniform Quantization of Neural Networks," ACM Transactions on Comput. Syst. **37**, 1–15 (2019).