Data Augmentation to Reduce Computational Complexity of Neural-Network-Based Soft-Failure Cause Identifier

Lareb Zar Khan[1,*], Pedro J. Freire[2], João Pedro[3,4], Nelson Costa[3], Antonio Napoli[5], and Nicola Sambo[1]

[1]Scuola Superiore Sant'Anna, Italy, [2]Aston University, United Kingdom, [3]Infinera Unipessoal Lda, Portugal, [4]Instituto de Telecomunicações, IST, Lisboa, Portugal, [5]Infinera, Munich, Germany

[*]larebzar.khan@santannapisa.it

Abstract: We investigated data augmentation to train neural networks (NNs) for soft-failure cause identification, demonstrating its impacts on NN complexity. Results indicate up to 68% reduction in the computational complexity of NN for each inference.

1. Introduction

In recent years, machine learning (ML) techniques have been extensively investigated for the realization of autonomous optical networks [1], with optical network failure management (ONFM) as a promising use case [2]. However, despite this extensive research, there is little emphasis on improving the quality of training data, which is critical in determining the performance of ML models. As a result, the cost of poor quality training data is usually paid in the form of increased (hidden) complexity of the ML model. One method of improving data quality is to eliminate the class imbalance, and in ONFM, class imbalance is common because some types of failures occur more frequently than others, resulting in an unequal distribution of observations within the dataset. Training ML models, in particular neural networks (NNs), with such imbalanced datasets, may result in a bias toward the majority class (i.e., more common failure) [3] that affects the performance of NNs in terms of accuracy and/or training time. In [4], data augmentation has been shown to provide benefits in reducing training time. However, there is a lack of research that analyzes the complexity of the NN, especially in the context of ONFM.

This paper focuses on the gains of data augmentation during the inference phase of the deployed NNs for soft-failure cause identification. The obtained results suggest that a good quality (i.e., balanced, hereinafter referred to as *modified*) training dataset can enable a less complex NN (NN_{LC}) to achieve similar performance as a more complex NN (NN_{MC}) trained with an imbalanced experimental dataset. Results also show that if we deploy NN_{LC} trained with the modified dataset for the identification of soft failures, we can reduce the multiplication or bit operations by up to 68% for each inference, implying a significant reduction in the complexity of NNs.

2. Experimental Testbed Setup for Data Acquisition

For this investigation, we collected data from the experimental testbed shown in Fig. 1. Commercial transponders were used on the transmitter (Tx) and receiver (Rx) sides. The testbed consisted of 4×80-km single-mode-fiber spans, denoted by S_1, S_2, S_3, and S_4. Erbium-doped fiber amplifiers (EDFAs), denoted as A_1, A_2, A_3, and A_4, were used at the end of each span. We artificially introduced five different soft-failures in the system relying on a wavelength selective switch (WSS) placed at the end of S_2. Table 1 summarizes the configuration details for all five considered soft-failures. In the case of normal operation (i.e., no soft-failure), the WSS's central frequency was set at 192.3 THz, and its attenuation and bandwidth were set at 0 dB and 37.5 GHz, respectively.

The input (P_{in}) and output (P_{out}) power levels in each amplifier, as well as the bit error rate (BER) and the optical signal-to-noise ratio (OSNR) in the coherent receiver, were extracted from the testbed. Two imbalanced datasets were considered: UD_1, where only BER and OSNR were taken into account; UD_2, where we included also P_{in} at A_2 in addition to BER and OSNR. UD_1 was inseparable, whereas UD_2 was separable in N-dimensional space, where N is the number of input features considered (i.e., 2 for UD_1 and 3 for UD_2).

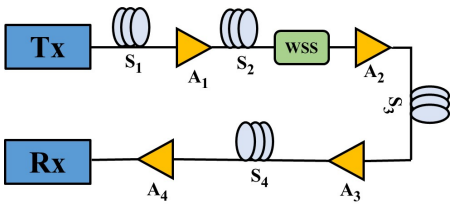

Fig. 1. Experimental testbed setup.

Soft-Failure	Filter Bandwith (GHz)	Attenuation (dB)	Central Frequency (THz)
Filter Tightening	26	0	192.3
Attenuation	37.5	6	192.3
Filter Tightening + Attenuation	26	6	192.3
Filter Tightening + Filter Shift	26	0	192.32
Filter Shift	37.5	0	192.32

Table 1. Considered soft-failures.

3. Proposed Approach for Data Augmentation

Data augmentation has been achieved using a variational-autoencoder (VAE) followed by a synthetic samples selector (SSS) as in Fig. 2. A VAE is a generative ML model with an encoder and a decoder (both of which are NNs). The generative capabilities of VAE stem from its encoder, which encodes input as a distribution over latent space (i.e., encoder output). The latent space distributions are then randomly sampled multiple times, and the resulting latent vectors are fed to the VAE decoder, whose task is to reconstruct the encoder's actual input. However, due to the imperfect training, the reconstructed input is not the same as the actual input. But, it can be used as synthetic data because it retains the underlying pattern of the actual input data.

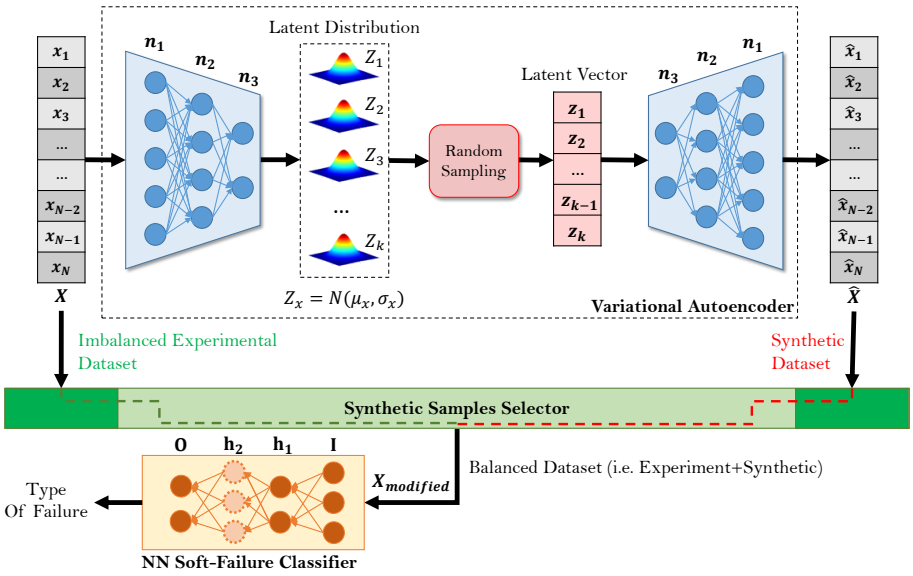

Fig. 2. Proposed approach: variational-autoencoder followed by a synthetic samples selector.

Once synthetic data ($\hat{X}$) are generated, its subset is selected by SSS. In particular, among ($\hat{X}$), for each under-represented soft-failure class, SSS selects data by minimizing their Euclidean distance from the mean of that same class in the actual imbalanced experimental dataset (X). The required number of selected synthetic samples is added to each under-represented soft-failure class in order to obtain the modified dataset ($X_{modified}$, i.e., balanced). It should be noted that only the training dataset was modified, while the validation and test datasets remained unchanged for an unbiased evaluation of NNs trained with modified and unmodified training datasets. Hereinafter, modified (i.e., balanced) datasets obtained from UD_1 and UD_2 are referred to as MD_1 and MD_2, respectively.

4. Results and Complexity Analysis

For soft-failure cause identification, we used a NN (NN_{MC1} with 325 trainable parameters and size: input layer (I) × hidden layer-1 (h_1) × hidden layer-2 (h_2) × output layer (O) = 2 × 20 × 10 × 5) and a comparatively less complex NN (NN_{LC1} with 125 trainable parameters and size: I × h_1 × O = 2 × 15 × 5). Both NN_{MC1} and NN_{LC1} were trained using UD_1 and MD_1, and different aspects of their performance were analyzed.

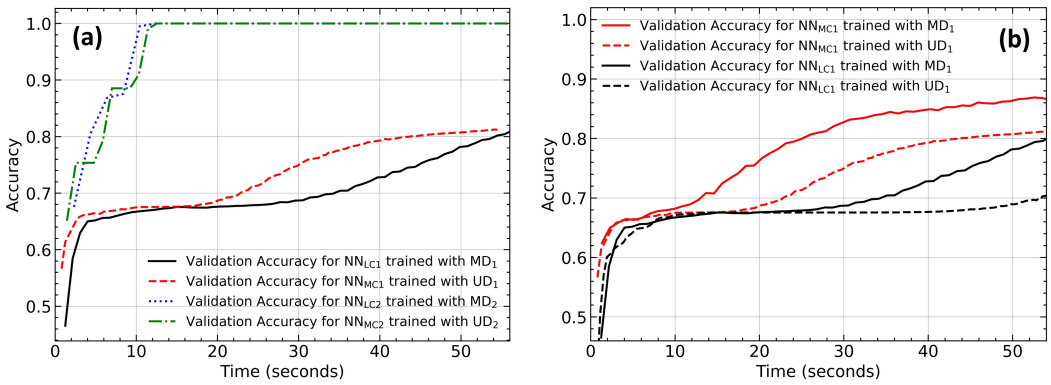

Fig. 3. Performance comparison: (a) NN_{LC1}, NN_{MC1}, NN_{LC2}, and NN_{MC2} trained with MD_1, UD_1, MD_2 and UD_2, respectively (b) NN_{LC1} and NN_{MC1} trained with both MD_1 and UD_1

We also compared the performance of another NN (NN_{MC2} with 173 trainable parameters and size: $I \times h_1 \times h_2 \times O = 3 \times 10 \times 8 \times 5$) trained with UD_2, to a comparatively less complex NN (NN_{LC2} with 59 trainable parameters and size: $I \times h_1 \times O = 3 \times 6 \times 5$) trained with MD_2. The results in Fig. 3(a) show that NN_{LC2} achieved 100% validation accuracy in 12.44 seconds and NN_{MC2} achieved the same accuracy in 12.53 seconds. Since UD_2 was separable, attaining 100% validation accuracy was possible but for UD_1, validation accuracy of only up to 81.6% was achieved because of its inseparable nature. NN_{LC1} achieved 81.6% validation accuracy in approximately 56 seconds, while NN_{MC1} achieved the same validation accuracy in 54.3 seconds. From both the considered cases, it can be inferred that a complex NN trained with a low-quality dataset (i.e., unmodified in this case) and a comparatively less complex NN trained with a good-quality dataset can result in similar performance. This is possible because, as suggested in Fig. 3(b), improving the quality of the training dataset improves the classification accuracy of a NN. The corresponding F1-score on test dataset also improves as for NN_{MC1}, it improved from 0.57 to 0.75 by changing the training dataset from UD_1 to MD_1, and a similar improvement was observed for NN_{LC1}.

As NN_{LC} trained with a modified dataset performed similarly to NN_{MC} trained with the unmodified dataset, using NN_{LC} for the soft-failure cause identification can save computational resources during the inference phase. To quantify this saving, we considered two commonly used metrics: the number of (i) real multiplications (RMs) and (ii) bit operations (BOPs). RMs provide an estimate of computational complexity (CC) in terms of the number of multipliers required while ignoring adders. This is because adders, unlike multipliers, are typically simpler to implement in hardware or software [5]. For a dense (i.e., fully-connected) layer of a NN, RMs = $n_n \times n_i$ [6], where n_n is the number of neurons in the given layer and n_i is the number of inputs to each neuron in that layer. On the other hand, the BOP is a more comprehensive metric because it takes into account both multiplication and addition operations. If b_w and b_i represent the bit-lengths of each weight and input, respectively, then Eq. 1 gives an estimate of BOPs for a dense layer [6].

$$BOPs \approx n_n n_i [b_w b_i + b_w + b_i + log_2(n_i)] \tag{1}$$

The quantitative analysis in Table. 2 shows how the computational complexity of a NN deployed for soft-failure cause identification reduces in terms of RMs and BOPs. NN_{LC1} requires 63.79% less multipliers and up to 64.06% fewer BOPs than NN_{MC1} for each inference. Similarly, deploying NN_{LC2} in place of NN_{MC2} to achieve similar performance for soft-failure cause identification can reduce the RMs and BOPs by up to 68%.

CC Metric	b_i	b_w	NN_{MC1}	NN_{LC1}	%Reduction	NN_{MC2}	NN_{LC2}	%Reduction
RMs	-	-	290	105	63.79	150	48	68.00
BOPs	8	8	24272	8724	64.06	12434	3947	68.26
BOPs	16	16	84592	30564	63.87	43634	13931	68.07
BOPs	32	32	316592	114564	63.81	163634	52331	68.02
BOPs	64	64	1226032	443844	63.80	634034	202859	68.01

Table 2. Computation complexity (CC) comparison in terms of BOPs and RMs: NN_{MC1} vs. NN_{LC1} and NN_{MC2} vs. NN_{LC2}.

5. Conclusions

We investigated a variational-autoencoder-based data augmentation technique for pre-processing of training datasets for neural networks (NNs) used for soft-failure cause identification. It has been shown that adding augmented data to an experimental training dataset can enable a less complex NN (NN_{LC}) to achieve classification accuracy comparable to a complex NN (NN_{MC}) trained with non-augmented dataset. Besides, NN_{LC} saves up to 68% computational resources for each inference as compared to NN_{MC}.

Acknowledgements: This work has been funded by EU H2020 Marie Skłodowska-Curie Actions ITN projects: MENTOR (GA 956713) and REAL-NET (GA 813144). J. Pedro, N. Costa and A. Napoli thank the European Commission for funding their activities through the H2020 B5G-OPEN (GA 101016663).

References

1. R. Gu *et al.*, "Machine learning for intelligent optical networks: A comprehensive survey," J. Netw. Comput. Appl. **157**, 102576 (2020).
2. F. Musumeci *et al.*, "A tutorial on machine learning for failure management in optical networks," J. Light. Technol. **37**, 4125–4139 (2019).
3. V. A. Fajardo *et al.*, "On oversampling imbalanced data with deep conditional generative models," Expert Syst. with Appl. **169**, 114463 (2021).
4. L. Z. Khan *et al.*, "Data augmentation to improve machine learning for optical network failure management," in *2022 European Conference on Optical Communication (ECOC),* (2022).
5. E. Jacobsen *et al.*, "Fast, accurate frequency estimators [dsp tips & tricks]," IEEE Signal Process. Mag. (2007).
6. P. J. Freire *et al.*, "Neural networks-based equalizers for coherent optical transmission: Caveats and pitfalls," IEEE J. Sel. Top. Quantum Electron. **28**, 1–23 (2022).