

Understanding Signal Feature Impact in CNN-Based Modulation Classification with Realistic Datasets

Bruno M. S. Teixeira[1,2], Thalles M. Moreira[1,2], João Rodrigo Faria[1,2], Fábio D. L. Coutinho[1,2], Samuel S. Pereira[1,2], and Arnaldo S. R. Oliveira[1,2(✉)]

[1] Instituto de Telecomunicações, Aveiro, Portugal
[2] Departamento de Eletrónica, Telecomunicações e Informática, Universidade de Aveiro, Aveiro, Portugal
{bmteixeira,thallesmmoreira,jrfaria,fabiocoutinho,samuel.s.pereira, arnaldo.oliveira}@ua.pt

Abstract. Automatic Modulation Classification (AMC) plays a crucial role in modern communication systems by facilitating signal identification before demodulation. In recent years, Convolutional Neural Networks (CNNs) have been increasingly adopted for this task due to their inherent pattern recognition capabilities and robustness against noise, making them particularly suitable for the highly dynamic scenarios envisioned in future wireless communication systems. Regarding the model optimization process, the data characteristics are crucial in ensuring the model's readiness for real-world conditions. In the current literature, most open-access datasets primarily emphasize cumulative channel and hardware impairments introduced during signal transmission and reception. However, the influence of specific signal characteristics on CNN-based modulation classifiers remains underexplored. This paper addresses this gap by presenting an experimental dataset of M-ary Quadrature Amplitude Modulation (QAM) signals with diverse features, aiming to analyze their impact on CNN training performance. The results offer valuable insights into how factors such as the number of input samples, the number of examples per modulation order, and the oversampling ratio (OSR) affect model performance and generalization.

Keywords: automatic modulation classification · convolutional neural networks · experimental datasets · oversampling ratio · data features training impact

1 Introduction

Radio Frequency (RF) signal transmission, identification, and decoding have always played a crucial role in communication systems. Once the signal is received, down-converted, and digitized, the Digital Signal Processor (DSP)

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15886, pp. 124–136, 2026.
https://doi.org/10.1007/978-3-031-97576-9_8

modules play a pivotal role in processing the signal until demodulation. An essential step toward demodulation is the identification of the signal's modulation scheme, which is necessary to reconstruct the original signal and ensure it is ready for decoding and error correction [1].

Automatic Modulation Classification (AMC) enables the receiver to dynamically identify the modulation scheme of heterogeneous signals from diverse scenarios and environments [2]. Traditional AMC techniques often rely on decision theory and feature-based methods. The most commonly used approaches in each of these categories, as discussed in the literature, are maximum likelihood estimation and statistical signal properties, respectively. While these methods are effective in controlled environments, their performance degrades in dynamic and noisy conditions due to their intrinsic iterative nature and limited flexibility, which reduces their applicability in real-world scenarios [3]. To address this, classical machine learning (ML) classifiers like support vector machine (SVM) have been suggested in the literature, which offer a promising alternative by providing a feed-forward strategy and a more flexible approach compared to baseline methods. Despite these advances, classical ML saturates at 90 % accuracy on basic modulation sets (e.g., QPSK and 16-QAM) in AWGN channels at moderate SNR. Still, its performance deteriorates significantly when dealing with higher-order constellations or in the presence of fading and hardware impairments [4]. More recently, deep learning (DL)-based algorithms have garnered significant attention in the literature, particularly with the use of Convolutional Neural Networks (CNNs) [5].

CNN-based approaches have gained significant recognition across various scientific domains due to their inherent advantages, such as effective feature extraction, robustness to noise and input variations, and high classification accuracy. These make them strong candidates for AMC applications. Another crucial feature is the CNN architecture's independence from the input height and width, which is advantageous because this processing function is typically implemented in edge computing environments [6]. In these deployment scenarios, significant challenges arise for the hardware deployment of DL-based algorithms due to limited computational resources, strict latency requirements, and varying wireless conditions. Studies on CNN deployment have shown that, in environments that require a trade-off between energy consumption and computational resources, Field-Programmable Gate Arrays (FPGAs) are more appealing than Graphics Processing Units (GPUs) and Central Processing Units (CPUs). Due to their reconfigurability, FPGAs enable customized parallelism and dataflow architectures, resulting in lower latency and higher energy efficiency, especially in real-time or edge inference scenarios [7].

Furthermore, the dataset's representativeness in relation to the target scenario is crucial for optimizing a CNN-based classifier with the smallest possible number of learnable parameters. The dataset is used in the offline training process, allowing the CNN model to generalize the characteristics of the environment under consideration. Due to the relevance of this, various studies have

utilized real-world-oriented datasets to demonstrate two key aspects: first, that CNN-based modulation classifiers are feasible even when considering channel and hardware impairments [8]; and second, that the CNN architecture can be optimized for the specific target scenario [5]. Nevertheless, the availability of these datasets obtained through experimental testbeds remains limited, constraining the study and proliferation of CNN-based classification estimators. Moreover, the literature still lacks a mature approach to introducing and adopting real-world scenario datasets in DL-based algorithms. This leads to skepticism regarding these algorithms for practical applications and prototyping wireless networks with such attributes.

With this in mind, this paper follows a complete workflow, from data generation to the prototyping phase of CNN-based classification estimators for edge implementation. An experimental dataset containing Quadrature Amplitude Modulation (QAM) with various signal features is generated to study its impact on CNN model training. The obtained training performance provides insights into the effects of factors such as the number of input samples, the number of examples per modulation order, and the oversampling ratio (OSR). This information, in turn, enables us to draw conclusions that will help optimize the model specifically for edge deployment by identifying trade-offs between these factors. Additionally, the entire dataset is made available to address the need for real-world-oriented implementations within the scientific community, aiming to foster the proliferation of various DL-based algorithms across the network stack[1].

The rest of the paper is structured as follows: Sect. 2 provides an overview of existing related work. Section 3 outlines the groundwork for implementation, while Sect. 4 provides the necessary details on the experimental setup used. Section 5 presents the obtained results and their subsequent discussion. Finally, the future work is explored in Sect. 6, followed by Sect. 7.

2 Related Work

Making Realistic datasets available has been a growing trend in the literature. This section provides an overview of datasets tailored for modulation recognition. With edge environments as the target deployment scenario, it is crucial to consider a key aspect: the complexity of the CNN-based model for modulation recognition. As mentioned earlier, reducing the number of learnable parameters by optimizing the model is essential, and to achieve this, understanding the impact of dataset characteristics is crucial. In addition to the effects of the radio frequency (RF) chain and the resulting RF impairments [3], as well as the channel conditions, we focus on isolating the impact of signal generation parameters on CNN model training, such as the OSR. This is crucial for recognizing any modulation, even if the signal has a different symbol rate or bandwidth. The literature on this matter is limited, and understanding how these parameters affect the performance of CNN-based modulation classification is vital.

[1] https://www.kaggle.com/datasets/brunomst/cabledamcdata.

Table 1. Open access datasets available in the literature.

Dataset	SNR	Size	Type	M-QAM	OSR	Roll-off	Citation
RML 2016.10a	[−20,18]	44,160	Synthetic	4, 16, 64	–	–	[2,9]
RML 2016.01b	[–20,18]	60,000	Synthetic	4, 16, 64	–	–	[2]
RML 2018.01a	[−20,30]	638,976	Synthetic	4, 16, 32, 64, 128, 256	–	(0.1,0.4)	[2,3,10]
HisarMod2019.1	[−20,18]	180,000	OTA	4, 16, 32, 64, 128, 256	2	0.35	[2]
Ours	30	24,576	Cabled	4, 16, 32, 64, 128, 256	6, 8, 10, 12	0.2	–

The available datasets targeting modulation classification with open access in the literature are displayed in Table 1, which includes data generation features such as the size in terms of samples per modulation, the obtained signal-to-noise ratio (SNR), the OSR value, the used roll-off filter, the works where it is used, and the type of dataset. As we are targeting QAM modulation for the envisioned model scenario, due to the verified complexity of achieving good performance with these algorithms in this case, along with the verified variance for this modulation [3], this table also provides information about the modulation order (M).

From Table 1, three different types of datasets are reported: synthetic, OTA, and cabled. Given the need to individualize the data generation features due to the gap found in the literature, the impact of channel impairments is minimized in the open-access dataset we provide. This helps create a dataset with a distinct nature, focusing more on data generation aspects rather than channel impairments. The inherent RF impairments from the transmission and reception RF chains are also present.

Typical datasets, such as Radio ML 2016.10a [9], primarily generated synthetic signals with simulated channel impairments, including multi-path fading, Doppler shift, symbol timing offset, and carrier frequency offset. A more detailed version, Radio ML 2018.01a [3], expanded the modulation variety and updated the channel impairments to include in-phase and quadrature component (IQ) imbalance, phase noise, and enhanced multi-path fading models. These improvements help reduce the gap between synthetic signals and real-world over-the-air (OTA) conditions. Regarding OTA datasets, the one provided in [2] is unique and stands as the most documented open-access dataset in the literature.

Nevertheless, the current open-access datasets in the literature exhibit a gap in dataset transparency or the range of data feature generation. Therefore, our provided open-access dataset aims to contribute by offering an alternative that enhances the existing datasets, improving transparency and expanding the variety of data features.

3 Workflow

This section outlines the workflow followed in this study, covering dataset organization, CNN architecture, and the training procedure. Figure 1 illustrates the

overall workflow, which consists of three main steps: experimental data generation, training the CNN model, and prototyping the envisioned CNN for edge deployment.

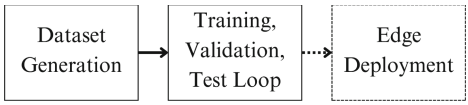

Fig. 1. Design workflow utilized in this work considering future edge deployment.

3.1 Dataset Organization

The different datasets adopted in this work contain various features with distinct value ranges. To establish a baseline comparison with our experimentally obtained dataset, we adopt the Radio ML 2018.01a dataset. This serves both as a validation for our code and a means to identify any missing data features in the existing dataset.

Figure 2 illustrates the various dimensions of the provided open-access dataset, including different OSR values, the number of samples per example, and the number of examples per modulation. The displayed organization mirrors the database structure to facilitate its usability.

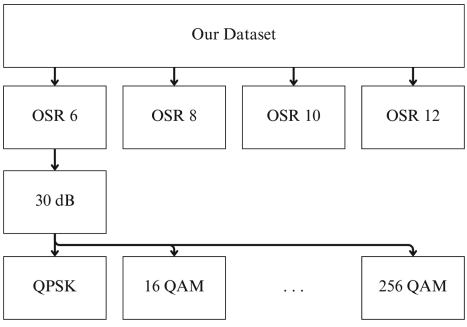

Fig. 2. Structure of the created dataset through the laboratory-acquired signals used in this study.

3.2 CNN Architecture

The CNN used in this study is based on the Deep Residual Network (ResNet) [11], well-known for its residual blocks. These blocks link the input of each block to its output, allowing the construction of deeper architectures while mitigating the vanishing gradient problem. Each residual block

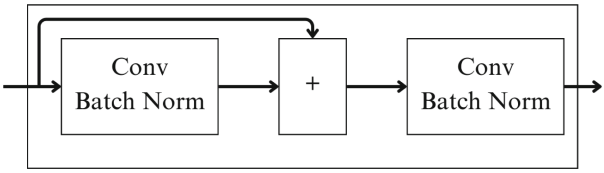

Fig. 3. Residual block structure.

typically consists of a convolutional layer (Conv), followed by batch normalization (BatchNorm), and finalized with a rectified linear unit (ReLU) activation layer as displayed in Fig. 3.

Table 2 presents the CNN architecture adopted in this work for $N_{classes} = 6$, corresponding to the number of modulation orders. This CNN consists of five residual blocks and is referred to as ResNet5. Note that $N_{samples}$ is integrated into the CNN structure to analyze its impact on model performance.

Table 2. CNN architecture details.

Layer	Output Dimensions
Input	$2 \times N_{samples}$
Conv	$16 \times N_{samples}$
Residual Block	$32 \times N_{samples}$
Residual Block	$64 \times N_{samples}$
Residual Block	$128 \times N_{samples}$
Residual Block	$256 \times N_{samples}$
Residual Block	$N_{classes} \times N_{samples}$
Flatten	$N_{classes} \times N_{samples}$
Fully Connected	$N_{classes}$
ReLU	$N_{classes}$

3.3 Training, Validation and Test Procedure

The CNN model optimization process is conducted on a machine equipped with an AMD Ryzen 9 7950X3D 16-Core Processor and an NVIDIA GeForce RTX 4080 SUPER GPU. Training is performed over 10 realizations to account for variations in weight initialization, with each realization consisting of 100 epochs. The PyTorch framework is used for model implementation, with the dataset divided into three subsets: 80% for training, 10% for validation, and 10% for testing.

Regarding the hyperparameter training, Table 3 provides a summary of them. The cross-entropy loss function is employed to compute training and validation losses. CNN weights are optimized using the Adam optimizer with a learning

rate of 6 $times 10^{-5}$, ensuring efficient minimization of the training loss. The batch size is set to 256.

Table 3. Training hyper-parameters.

Processes	Num. of Epochs	Dataset Partition	Loss Function	Learning Rate	Optimizer	Batch Size
10	100	Train 80%, Validation 10%, Test 10%	Cross-Entropy	6e-5	Adam	256

4 Experimental Dataset Generation

This section provides an overview of the entire process, including data generation, transmission, reception, and pre-processing steps.

4.1 Signal Generation

The general process for signal generation is illustrated in Fig. 4. In the first step, a sufficient number of signals is generated to ensure that every sequence of bits in the transmitted signal is unique.

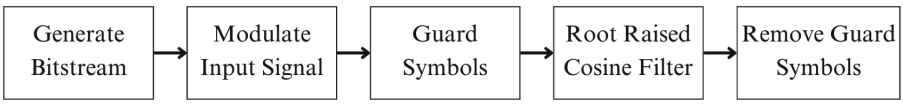

Fig. 4. Signal generation sequence.

A sampling frequency of 1.02 MHz is selected, as this value is evenly divisible by all considered OSR values. The Root Raised Cosine (RRC) filter is applied with a roll-off factor of 0.2, and 200 guard symbols are introduced to mitigate inter-symbol interference. During modulation, the adopted modulation orders (M) are 4, 16, 32, 64, 128, and 256, while the considered OSR values are 6, 8, 10, and 12. Consequently, the output signal for each configuration has a structure of 43 lines by 200,000 rows.

Table 4 provides a comprehensive description of the parameters used for each generation signal process.

4.2 Laboratorial Setup

The experimental setup for the signal acquisition process is shown in Fig. 5. As previously mentioned, the transmitted signal consists of 200,000 samples and is loaded into the Vector Signal Generator (VSG), specifically the SMA100J model.

Table 4. Dataset parameters.

Parameters	Values
Roll-off Factor	0.2
SNR	30 dB
Guard Symbols	200
OSR	6, 8, 10, 12
Bit Precision	32-bit floating-point
Number of samples	25,165,824 complex samples
Modulations	4QAM, 16QAM, 32QAM,64QAM, 128QAM, 256QAM

The VSG is programmed to operate at a center frequency of 500 MHz with a transmission power of -66 dBm, resulting in an approximate SNR of 30 dB. The VSG is directly connected to the Vector Signal Analyzer (VSA) via a coaxial cable. The VSA is configured to operate at the same center frequency, with a sampling rate of 1.02 MHz and a capture length of 100,000 samples.

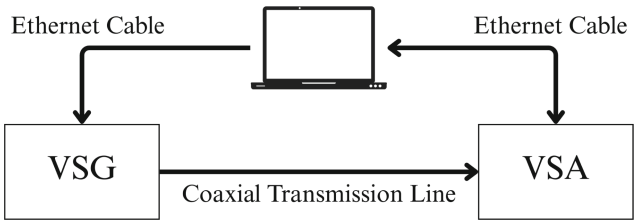

Fig. 5. Connection diagram between VSG, VSA and the computer.

After confirming the correct setup of both devices, a query is executed to retrieve samples from the VSA buffer. The received signal is then tested through a demodulation process. A new query is performed if the signal fails any of the tests. Otherwise, the next signal is loaded into the VSG for transmission. The query buffer size required careful consideration, as longer queries are more prone to errors. To accommodate larger sample buffers, the device reduces precision per sample. Given this, a query length of 100,000 samples is selected to ensure that each query remains smaller than the loaded signal, maintains the uniqueness feature, and prevents any loss of precision in the retrieved data. A picture of the laboratory setup is shown in Fig. 6.

4.3 Data Pre-processing

To feed the experimental data into the ResNet5 model for optimization, we adopt the same dataset organization as in the Radio ML 2018.01a. In our provided

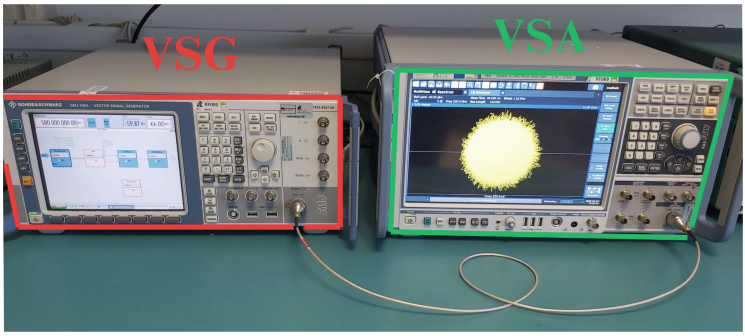

Fig. 6. Picture of the laboratory dataset acquisition setup.

dataset, the data in the .h5 file format is divided into the /X group, which contains the input data with a structure of ($N_{examples} \times N_{classes}$, 2, 1, $N_{samples}$), and the target output /Y with a structure of ($N_{examples} \times N_{classes}$, $N_{samples}$).

Therefore, it is necessary to prepare the received signal in the proper dataset structure. First, signal normalization is performed, followed by restructuring the data into variable dimensions, ensuring each example corresponds to a one-shot solution.

Radio ML 2018.01a Comparison. To perform the baseline comparison, it is necessary to individualize the 30 dB QAM modulations from the Radio ML 2018.01a dataset. It is important to note that this dataset is not normalized. However, we cannot perform normalization due to the lack of information relative to the signal generation, the maximum value in each example may not correspond to a constellation maximum, and normalizing it will introduce error.

5 Results and Discussion

In this section, we present and discuss the results of the modulation classification using the ResNet5 model with a batch size of 256, considering the training process over 100 epochs with 10 realizations to account for variations in weight initialization. The primary metric used for optimization is test accuracy, as it directly reflects the model's performance. In our experimental dataset, OSR values of 6, 8, 10, and 12 are considered. Additionally, to analyze the impact of data features on the ResNet5 model, different values for the number of samples per example, $N_{samples}$ equal to 512, 1024, 2048, and 4096, as well as the number of examples per modulation order, $N_{examples}$ of 512, 1024, 2048, and 4096, are considered. Note that, as a result of shortening or increasing $N_{samples}$, an inverse adjustment is applied to $N_{examples}$ to preserve the overall amount of data.

In Fig. 7, the test accuracy results are presented to compare the performance between the synthetic Radio ML 2018.01a dataset-denoted as *Baseline* in the figure legend-and the dataset proposed in this work. The primary objective is

to observe the general training behavior and assess the potential impact of the missing OSR value. However, it is essential to note that the Radio ML 2018.01a dataset includes a variety of hardware and channel impairments, which necessitates caution when concluding the influence of OSR. From the results, the OSR value of the *Baseline* dataset appears to lie between OSR = 10 and OSR = 12, based on the observed accuracy trends. Additionally, it is noticeable that for every first epoch, the accuracy of each OSR configuration in our dataset is significantly higher than that of the *Baseline*, with all OSR values showing relatively close performance. This behavior may be attributed to the lack of normalization in the *Baseline* dataset and the presence of hardware and channel impairments included in its generation process.

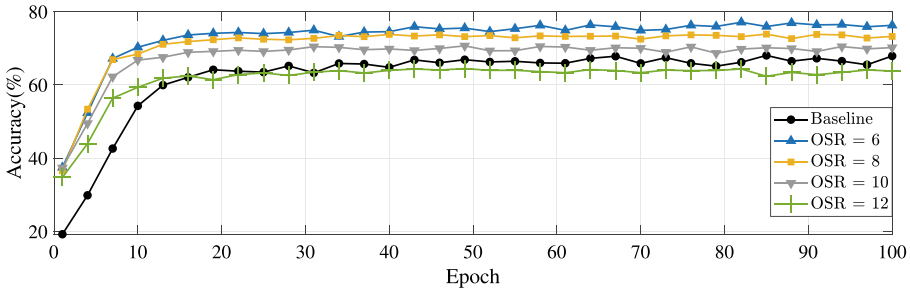

Fig. 7. Comparison of test accuracy across various OSR values from our experimental dataset and the RadioML 2018.01a dataset.

Figure 8 shows the impact of the sample size ($N_{samples}$) in the classification test accuracy for (a) $OSR = 6$ and (b) $OSR = 12$. It is noticeable that in the $OSR = 6$ scenario, the final accuracy improves, particularly for lower $N_{samples}$ values, which correspond to higher $N_{examples}$. This suggests that reducing the sample length improves performance, as the higher density of complementary data points between signal samples, due to the lower OSR, leads to a reduction in helpful information per sample by a factor of the OSR.

In Fig. 9, the impact of the number of examples per modulation ($N_{examples}$) on classification test accuracy is presented for (a) $OSR = 6$ and (b) $OSR = 12$. For a small number of examples, both OSR configurations yield similar accuracy. However, as the number of examples increases, accuracy improves for both, but not at the same rate. For $OSR = 6$, the accuracy difference between the lowest and highest number of examples is approximately 20%, while for $OSR = 12$, the improvement is only around 10%. Interestingly, despite $OSR = 12$ having double the oversampling of $OSR = 6$, it does not translate into higher classification accuracy, revealing an inverse relationship between OSR and classification performance in this scenario.

Figure 10 illustrates the impact of (a) $N_{examples}$ and (b) $N_{samples}$ on the obtained test accuracy for the different OSR values considered in this study. Given the significance of data volume in the training process, increasing the

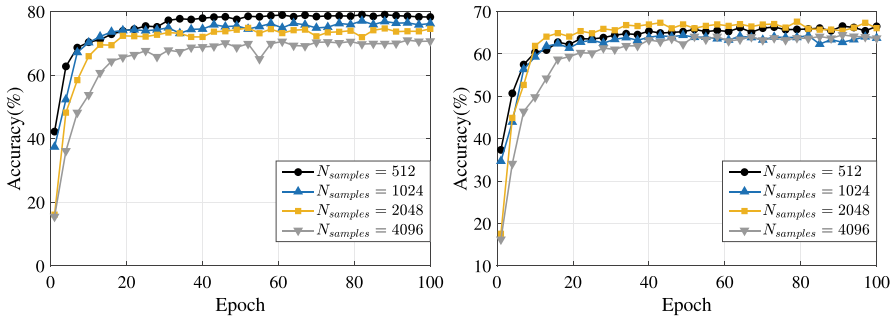

Fig. 8. Impact of the $N_{samples}$ in the obtained test accuracy considering (a) $OSR = 6$ and (b) $OSR = 12$.

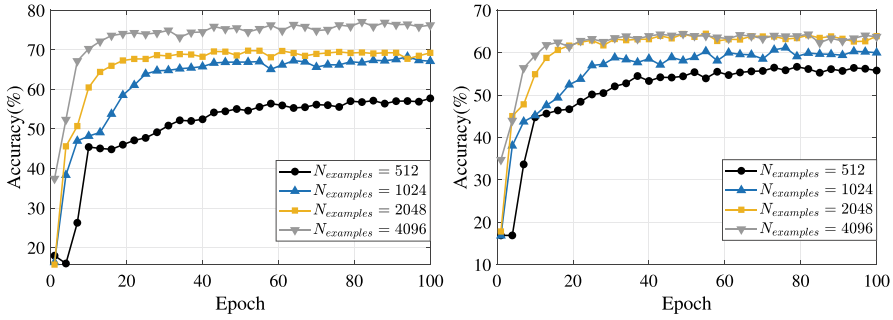

Fig. 9. Impact of the $N_{examples}$ in the obtained test accuracy considering (a) $OSR = 6$ and (b) $OSR = 12$.

number of examples would generally lead to improved model accuracy. However, when focusing on the effect of $N_{examples}$, it is observed that for higher OSR values, the accuracy does not increase linearly with the number of examples. On the other hand, increasing $N_{samples}$ leads to a consistent decrease in test accuracy across all OSR values. This suggests that while larger input sequences may carry more data, they do not necessarily contribute positively to the learning process, possibly due to redundancy or overfitting effects. Consequently, a trade-off between $N_{examples}$ and $N_{samples}$ must be considered when targeting a specific OSR value for edge deployment.

6 Way Forward

This paper aims to fill the gap in publicly available datasets of telecommunication modulations, the variation of OSR with real measurements. This is necessary to recognize any real-world signal with precision clearly. However, a fully

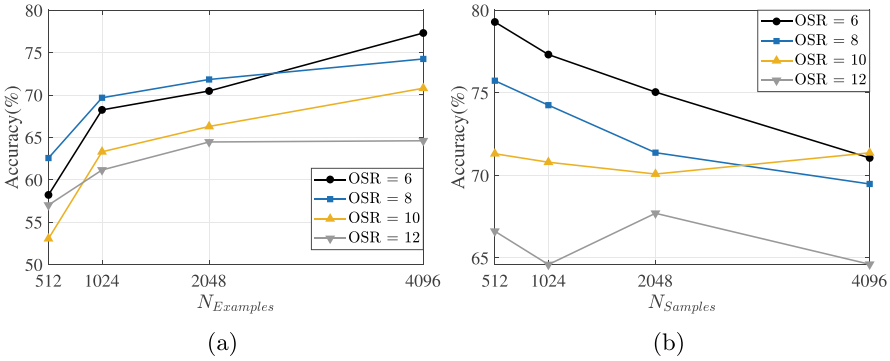

Fig. 10. Impact of the (a) $N_{examples}$ and (b) $N_{samples}$ in the obtained test accuracy for different OSR values.

autonomous signal recognition system needs more than just knowing the modulation. For that, carrier frequency, bandwidth, or symbol rate are also important. The inclusion of multiple ORS attempts to equip our ML algorithm with the ability to recognize any modulation, even if the signal has a different symbol rate or bandwidth; however, it does not enable us to determine the values for these parameters.

In our vision, this dataset must be extended to accommodate the integration of these parameters, carrier frequency, and bandwidth. Furthermore, since classical methods already exist to find such parameters, a comparison will be needed to have a fully autonomous signal recognition system that can operate at the edge using the minimum resource consumption.

7 Conclusion

This paper investigates the importance of data generation characteristics in the design of Convolutional Neural Network (CNN)-based classification estimators, with a focus on edge deployment. An experimental dataset comprising M-ary Quadrature Amplitude Modulation (QAM) signals with diverse characteristics is presented to evaluate the influence of specific signal features on CNN training performance. The results reveal that overlooked factors, such as the oversampling ratio (OSR), can significantly impact the inference capability of CNN-based models. Comprehensive evaluations considering both the number of samples and the number of examples per modulation order demonstrate that achieving optimal performance requires careful consideration of these data characteristics. Consequently, real-world-oriented datasets should explicitly document signal generation parameters and support trade-off analyses aligned with the design choices explored in this work.

Acknowledgments. This work was funded by the PRR - Plano de Recuperação e Resiliência and by the NextGenerationEU funds at Universidade de Aveiro, through the scope of the Agenda for Business Innovation "Agenda Microeletránica" (Project no.14 with the application C644916358-00000028). We acknowledge that ChatGPT was used to proofread the final draft of the article's text.

References

1. Pihlajasalo, J., et al.: Deep learning based OFDM physical-layer receiver for extreme mobility. In: 2021 55th Asilomar Conference on Signals, Systems, and Computers, pp. 395–399 (2021)
2. Zhang, F., Luo, C., Jialang, X., Luo, Y., Zheng, F.-C.: Deep learning based automatic modulation recognition: models, datasets, and challenges. Digital Signal Process. **129**, 103650 (2022)
3. Shea, T.J.O., Roy, T., Charles Clancy, T.: Over-the-air deep learning based radio signal classification. IEEE J. Sel. Topics Signal Process. **12**(1), 168–179 (2018)
4. Orlic, V.D., Dukic, M.L.: Automatic modulation classification algorithm using higher-order cumulants under real-world channel conditions. IEEE Commun. Lett. **13**(12), 917–919 (2009)
5. Sathyanarayanan, V., Wagner, M., Gerstoft, P.: Over the air performance of deep learning for modulation classification across channel conditions. In: 2020 54th Asilomar Conference on Signals, Systems, and Computers, pp. 157–161 (2020)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
7. de Lima Luiz, A., Zimmer, A., Brandmeier, T., Assump o Lolis, L.H.: Performance analysis of CNN speed and power consumption among CPUs, GPUs and an FPGA for EU GDPR 2016/679 compliant automotive applications. Transp. Res. Procedia **72**, 3046–3053 (2023). TRA Lisbon 2022 Conference Proceedings Transport Research Arena (TRA Lisbon 2022), 14th-17th November 2022. Lisboa, Portugal
8. Ya, T., Lin, Y., Haoran Zha, J., Wang, Z.Y., Gui, G., Mao, S.: Large-scale real-world radio signal recognition with deep learning. Chin. J. Aeronaut. **35**(9), 35–48 (2022)
9. Jiang, J., Wang, Z., Zhao, H., Qiu, S., Li, J.: Modulation recognition method of satellite communication based on CLDNN model. In: 2021 IEEE 30th International Symposium on Industrial Electronics (ISIE), pp. 1–6 (2021)
10. Lin, C., Yan, W., Zhang, L., Wang, W.: A real-time modulation recognition system based on software-defined radio and multi-skip residual neural network. IEEE Access **8**, 221235–221245 (2020)
11. Rosa, J., et al.: BACALHAUNET: a tiny CNN for lightning-fast modulation classification. ITU J. Future Evol. Technol. **3**, 252–260 (2022)