IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 20, 2020, accepted June 12, 2020, date of publication June 16, 2020, date of current version June 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3002770

Gated Recurrent Unit Neural Networks for Automatic Modulation Classification With Resource-Constrained End-Devices

RAMIRO UTRILLA[1], ERIKA FONSECA[2], ALVARO ARAUJO[1], AND LUIZ A. DASILVA[2,3], (Fellow, IEEE)

[1]B105 Electronic Systems Lab, ETSI Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain
[2]CONNECT Research Centre for Future Networks and Communications, Trinity College Dublin, Dublin 2, D02 PN40 Ireland
[3]Commonwealth Cyber Initiative, Virginia Tech, Arlington, VA 22203, USA

Corresponding author: Ramiro Utrilla (rutrilla@b105.upm.es)

This work was supported in part by the Science Foundation Ireland under Grant 17/NSFC/5224.

ABSTRACT The continuous increase in the number of mobile and Internet of Things (IoT) devices, as well as in the wireless data traffic they generate, represents an essential challenge in terms of spectral coexistence. As a result, these devices are now expected to make efficient and dynamic use of the spectrum by employing Cognitive Radio (CR) techniques. In this work, we focus on the Automatic Modulation Classification (AMC). AMC is essential to carry out multiple CR techniques, such as dynamic spectrum access, link adaptation and interference detection, aimed at improving communications throughput and reliability and, in turn, spectral efficiency. In recent years, multiple Deep Learning (DL) techniques have been proposed to address the AMC problem. These DL techniques have demonstrated better generalization, scalability and robustness capabilities compared to previous solutions. However, most of these techniques require high processing and storage capabilities that limit their applicability to energy- and computation-constrained end-devices. In this work, we propose a new gated recurrent unit neural network solution for AMC that has been specifically designed for resource-constrained IoT devices. We trained and tested our solution with over-the-air measurements of real radio signals. Our results show that the proposed solution has a memory footprint of 73.5 kBytes, 51.74% less than the reference model, and achieves a classification accuracy of 92.4%.

INDEX TERMS Cognitive radio, spectrum sensing, deep learning, automatic modulation classification, recurrent neural network, gated recurrent unit, IoT, end-device, edge computing, software-defined radio.

I. INTRODUCTION

The number of mobile and Internet of Things (IoT) devices, as well as the wireless data traffic they generate, continues to grow at an unprecedent rate [1]. These devices often operate in the same frequency bands, substantially increasing spectrum occupancy and posing new and essential challenges to overcome. One of these challenges is to achieve highly efficient and reliable communications in increasingly complex, heterogeneous, and dynamic scenarios, where many different radio systems coexist. To this end, multiple Cognitive Radio (CR) techniques have been proposed to sense and analyze the spectrum and, based on that information and on previous experience, make decisions about its use and adapt the communication parameters of the devices accordingly [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Ding Xu.

Performing these CR tasks in a centralized manner would generate latency problems and massive network traffic due to the data distribution between the devices, complicating the growing spectrum scarcity problem. For this reason, and aided by the improved capabilities of end-devices, non-cooperative and edge computing approaches are gaining importance in areas like the IoT [3], [4]. In these approaches, the end-devices are responsible for performing all or at least some of these computation-intensive CR tasks, allowing faster response times and reducing communication overhead [5]. The field that has emerged from the application of CR techniques to the IoT is known as Cognitive IoT (CIoT) [6], [7].

Spectrum sensing is the first step of the cognitive cycle. Within it, in this work, we focus on the Automatic Modulation Classification (AMC), as an integral part of intelligent radio systems. AMC consists of recognizing the modulation

scheme of a sensed signal, which is an essential feature to carry out multiple CR techniques, both in military and civilian applications [8]. As detailed in section II, many AMC methods have been proposed in the literature. However, most of them have not been conceived to be implemented in autonomous IoT end-devices, since they require high energy, processing, and storage resources.

In this work, we address AMC from the perspective of resource-constrained devices. This poses multiple challenges. AMC methods require the use of Software-Defined Radios (SDRs) to acquire the raw radio signals they need as input. However, low-power SDR end-devices have certain acquisition limitations compared to Universal Software Radio Peripherals (USRPs) and other widely used traditional SDR systems [9]–[11]. The complexity of AMC methods must therefore be reduced to adapt their requirements to the memory and processing resources of end-devices. This complexity reduction will decrease the classification accuracy of a method to a greater or lesser extent. Considering this, *our main objective is to achieve a highly accurate AMC solution, with a memory footprint that allows its implementation in resource-constrained end-devices, and to validate it with real samples acquired with a platform of these characteristics.* The main contributions of this work include:

1) A publicly available dataset consisting of over-the-air measurements of real radio signals with 11 different modulations [12]. These measurements were acquired with a resource-constrained SDR end-device [11], which is the type of platform targeted by this work. Specifically, the signals were recorded in an office environment at two different distances between the transmitter and the receiver, 1 and 6 meters, so their amplitude, noise level, and propagation conditions are different. Even so, in both cases, the Signal-to-Noise Ratio (SNR) is greater than 20 dB. All the tests carried out in this work were performed with this dataset. In this way, the AMC models studied have been evaluated under realistic conditions, both in the use of signals transmitted over the air and in the type of platform adopted to acquire those signals.
2) An empirical validation, using our dataset, of a reference Gated Recurrent Unit (GRU) neural network model for AMC [13]. This model was originally evaluated with RadioML2016.10a[1], a dataset of synthetically generated radio signals using a channel model that includes the following effects: random center frequency and sample rate offsets, additive white Gaussian noise, multi-path, and fading [14]. Specifically, we selected this model for its low memory footprint. The results obtained training and evaluating this model with our generated dataset are consistent with those of the reference work under equivalent high-SNR conditions, establishing a solid starting point for our subsequent study.
3) A study of how multiple parameters affect the classification accuracy, the complexity, and the memory footprint of the reference GRU neural network model. The parameters evaluated are the size of the training set, the length of the input vector, the number of layers in the model, and the number of cells in each layer. The procedure carried out provides useful insights on how to optimize other models for resource-constrained devices.
4) A new GRU neural network model and a new training set size resulting from the previous study. The memory footprint of our proposed solution is 73.5 kBytes, while that of the reference model is 152.3 kBytes [13]. In addition, a classification accuracy of 92.4% has been achieved by increasing the training set size. This was possible since the potential of the model was not fully exploited with the original size used in the reference work.

The rest of the paper is organized as follows. Section II presents related works in AMC and its implementation in real devices. Section III contains a full description of the resources and methodology used in this work. The results obtained in the different tests are presented and discussed in section IV. Conclusions are offered in section V.

II. RELATED WORK

As already mentioned, AMC is an essential task to carry out multiple CR techniques aimed at improving spectral efficiency, such as dynamic spectrum access, link adaptation and interference detection, among others [8]. In addition, AMC is also useful at a spectrum enforcement level, as it allows devices to better analyze the wireless environment, identify suspicious transmissions, detect abnormal behaviors and localize sources of interference [15].

For this reason, AMC is a long-studied problem in the field of CR. It has been traditionally approached from two major kinds of methods: those based on likelihood theory, and those based on feature extraction and classification techniques. Likelihood-Based (LB) methods suffer from high computational complexity in most practical cases, and in the Feature-Based (FB) approach, the design of feature extractors is a time-consuming task that requires considerable domain expertise [16]. Furthermore, these drawbacks are compounded by the increasing complexity of wireless scenarios. On the other hand, Deep Learning (DL) methods can be directly fed with raw data and automatically discover a suitable internal representation or feature vector from which the learning subsystem, often a classifier, can detect or classify patterns in the input. These methods have dramatically improved the state of the art in many domains, demonstrating better generalization, scalability and robustness capabilities than previous solutions [17]. As a result, in recent years, many researchers are exploring the application of DL methods to the AMC field, obtaining promising results [5], [18].

The generation of radio signal datasets, and their open-access publication and availability to the community,

[1]https://www.deepsig.ai/datasets

have greatly contributed to accelerate this progress. RadioML [14] is a representative example of it. This dataset was generated with the GNU Radio [19] development toolkit, and it contains a set of synthetically generated signals with different SNR levels that correspond to different modulations. Along with the dataset, O'Shea *et al.* published several works where they proposed AMC methods based on Deep Convolutional Neural Networks (DCNNs) [14], [20]. These networks are designed to process data that come in the form of multiple arrays and are the dominant method in the computer vision domain, where they have brought about significant advances [17]. In fact, in a later work [21], these authors achieved accuracy results close to 100% at high SNRs, and they also validated their model with over-the-air measurements of the signals.

Many works have also used DCNN structures to explore the AMC problem considering other aspects, such as the input wireless signals representation [22], the effect of SNR variations between the training and testing phases on model robustness [23], and the efficiency of training and retraining processes [24]. However, all of them are complex solutions with a memory footprint of the order of hundreds of megabytes or higher [25], which makes their implementation in autonomous resource-constrained devices unfeasible.

Model quantization is an enabling step to implement this kind of solutions in real embedded systems with less computational, storage and energy resources. However, direct quantization usually leads to accuracy losses. Therefore, new methods are being proposed to avoid this degradation. Nagel *et al.* [26] have recently introduced a data-free FP32 to INT8 quantization method that achieves state-of-the-art results. Additionally, because of its great flexibility, the use of special purpose hardware, such as Field Programmable Gate Arrays (FPGAs), is being widely explored to bring DL techniques to end-devices [27], [28]. Consequently, during the last years, multiple tools for mapping neural networks on FPGAs have appeared [29], [30].

However, even with quantification techniques, the memory required by DCNN-based methods is still high for end-devices, which usually have between 128 and 512 kBytes of non-volatile memory [31]–[35]. In addition, these methods would involve a high computational load on these devices, which would result in high latencies and could adversely affect other system tasks that share the processor [36].

For this reason, other works have proposed the use of less complex Recurrent Neural Networks (RNNs) [13], [25]. These networks have feedback connections, so they retain an internal hidden state that implicitly contains information about all the past elements in a data sequence [37]. In this way, the network can discover correlations between them. This feature makes RNNs prime candidates for learning problems that involve sequential data, such as handwriting and speech recognition, or the classification of time-series and medical signals [17], [38], [39].

Conventional RNNs often experience vanishing/exploding gradient problems during their training [17]. For this reason, multiple variants have been proposed, such as the GRU [40] and Long Short-Term Memory (LSTM) [41] neural networks. These networks introduce gating mechanisms to regulate the input, output, and feedback information in the cells, and have proved to be more robust against these training problems.

Hong *et al.* [13] explored the use of GRU neural networks for the AMC problem. Specifically, they proposed to directly feed these networks with the In-phase and Quadrature (I/Q) samples of the baseband radio signals. They evaluated the impact of varying the number of layers in their proposed structure and the number of cells in each layer. This work shows that with a much less complex model than the one used by O'Shea *et al.* [20] accuracy results above 90% can be achieved. Similarly, Rajendran *et al.* [25] also used this type of simpler RNN structures but, instead of GRUs, they employed LSTM cells, which are slightly more complex [42]. Besides, they fed their model with amplitude and phase information, since they obtained poor results by simply providing raw I/Q samples. In this work, they also studied the impact of quantifying the weights and activation functions of the model with the aim of reducing its computational load and memory footprint requirements. With this approach, they observed a 10% accuracy loss with respect to their full precision version.

The solutions proposed in these two works achieve memory footprints of 152.3 and 400.1 kBytes, respectively. These are in the order of the memory available in typical IoT end-devices, which is hundreds of kilobytes. However, devices cannot allocate most of their memory only to the AMC task. For this reason, in this work, we study these proposals in depth, and we evaluate the parameters that affect the memory footprint and the classification accuracy to finally propose an AMC model optimized for resource-constrained devices.

III. METHODOLOGY

A. PROBLEM STATEMENT

Modulation classification is generally addressed as an N-class classification problem, in which each of the N classes corresponds to a different modulation. The received signal $r(t)$ can be expressed as follows:

$$r(t) = s(t) * c(t) + n(t), \tag{1}$$

where $s(t)$ is the modulated signal from the transmitter, $c(t)$ is the impulse response of the wireless communication channel, and $n(t)$ is additive noise. Thus, given $r(t)$ as the only input signal, the modulation classifier must determine the probability that $s(t)$ is modulated with each of the N possible modulations. This can be expressed as follows:

$$P_i = P(s(t) \in N_i | r(t)), \tag{2}$$

where P_i is the probability that $s(t)$ is modulated with the modulation corresponding to the ith class.

Radio receivers used in CR usually provide the received signals in I/Q format. The in-phase and quadrature components of a received signal $r(t)$ can be expressed as

$I = A\cos(\varphi)$ and $Q = A\sin(\varphi)$, where A and φ are the instantaneous amplitude and phase of $r(t)$.

B. BASELINE METHOD

The gated RNN structures proposed in [13], [25], and further discussed at the end of section II, are similar. In this work, we take the solution proposed by Hong *et al.* [13] as the baseline method for two main reasons:

- Whereas the classification accuracy results are similar with GRU and LSTM cells, GRUs have fewer trainable parameters [43], so they require less memory, and they train and execute faster than LSTMs.
- As the transceiver directly provides the I/Q samples of the received signal, the use of this format at the model input is more efficient than the use of amplitude and phase information, the calculation of which involves additional computational overhead for the device.

As previously mentioned, Hong *et al.* evaluated different versions of a model, varying the number of GRU layers and the number of cells per layer. From all their evaluated versions, in this work, we take as our reference model the structure shown in Fig.1, which is formed by one GRU layer with 100 cells followed by two dense fully connected layers with 64 and 11 units respectively. This decision involves a trade-off between classification accuracy and memory and processing requirements. Specifically, using the Keras library, the RadioML2016.10a dataset and for high SNRs, this model achieves a classification accuracy of 85%, compared to 90% of the version with two GRU layers, which requires more than double the memory. On the other hand, the accuracy improvement between the version with 100 cells and those with 150 and 200 cells is negligible.

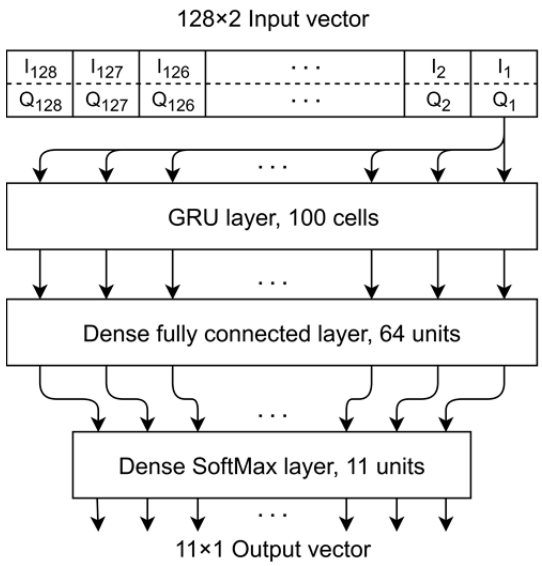

FIGURE 1. Baseline GRU network model for AMC.

In this model, the GRU layer applies the Rectified Linear Unit (ReLU) function, while the final dense layer uses the SoftMax function. In addition, the RMSprop optimizer and the categorical cross entropy loss function were adopted for the network training. Finally, to perform a classification, this model requires an input of 128 I/Q samples, which are shaped as a 128×2 vector. These samples are fed to the model sequentially, i.e., each of these 128 I/Q samples is simultaneously fed to all GRU cells in the first layer at its corresponding time step.

C. DATASET

Addressing the AMC problem from the perspective of end-devices poses a challenge in terms of resource utilization, but also because of their signal acquisition capabilities, which are limited compared to traditional SDR systems. For this reason, in this work we have generated a new dataset with wireless signals recorded in a real environment directly by a resource-constrained SDR end-device. Specifically, we have used MIGOU [11], a low-power experimental platform with SDR capabilities that has been specifically designed to address the hardware architectural constraints that limit CR research and experimentation with low-power end-devices. This platform was configured to sense a communication channel and send the raw I/Q samples (without timing recovery) to a computer that properly stores them.

A USRP B210 connected to another computer running GNU Radio was used on the transmitter side. In order to implement the different transmitters with the same set of 11 modulations adopted by the baseline work, we used the source code[2] and same data sources with which RadioML2016.10a was generated. Figure 2 shows the complete dataset generation set up, where d is the distance between both devices.

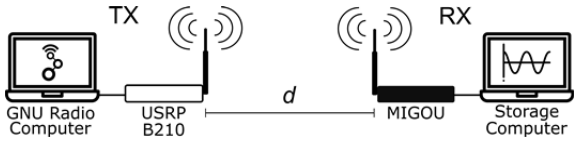

FIGURE 2. Dataset generation scenario set up.

The configuration parameters of the different elements used to generate the dataset are shown in Table 1. In addition, the parameters related to the content and format of the generated dataset are shown in Table 2.

At this initial stage, the purpose of this work is to determine the maximum classification accuracy that can be achieved in real conditions with an AMC model specifically designed for resource-constrained devices. As shown in previous works [13], [20]–[25], the best results are always obtained in low-noise conditions, i.e. no significant improvements are seen for SNR values greater than 15 dB. Therefore, the basis for selecting the distances from the transmitter at which to make the recordings was to ensure a SNR greater than this value.

[2] An error with AM modulations, solved in later versions of the RadioML dataset, was corrected for our dataset generation.

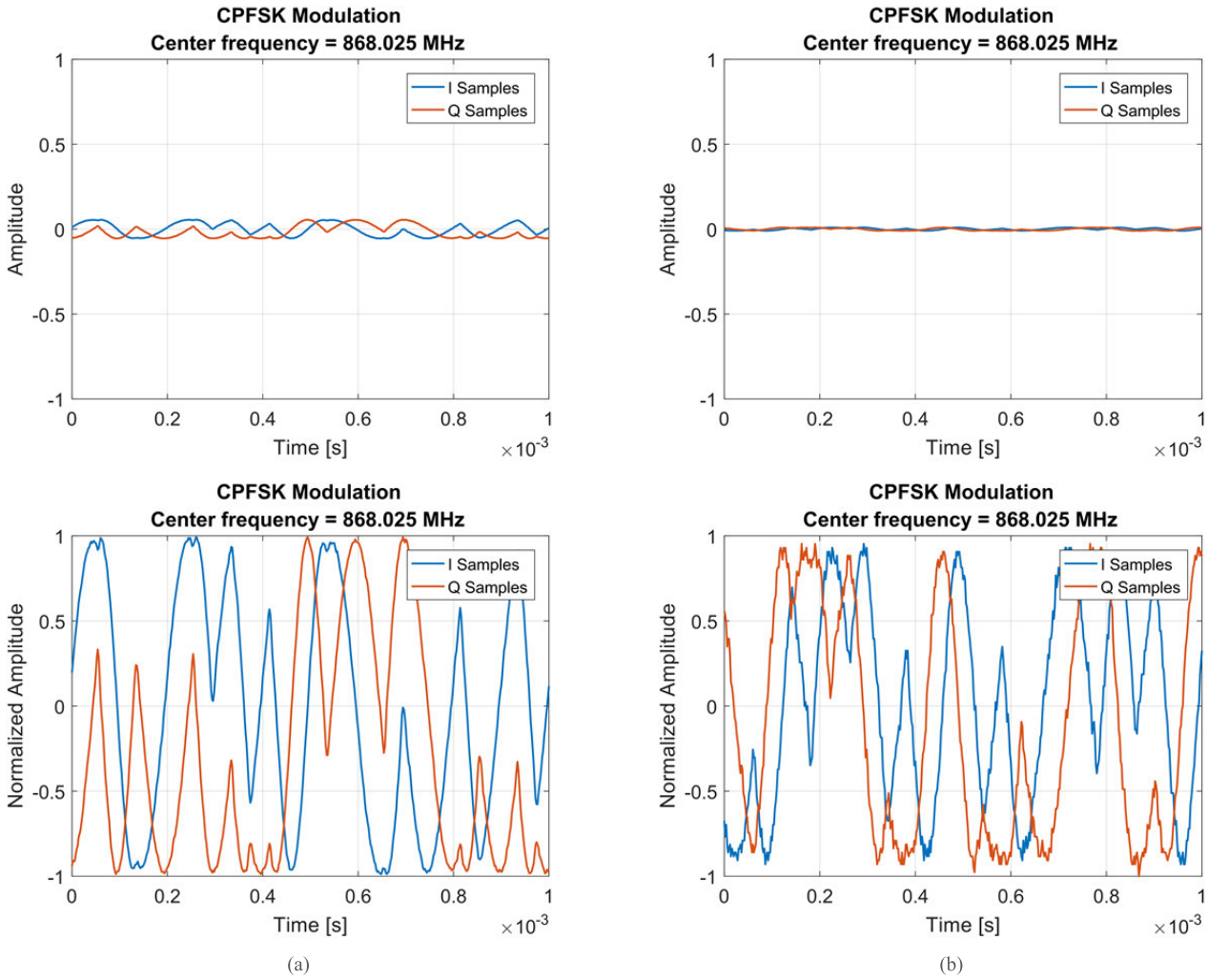

FIGURE 3. Comparison of signals recorded at (a) 1 and (b) 6 meters. The signals in the bottom row are the normalized version of those in the top row.

TABLE 1. Configuration parameters of the dataset elements.

Element	Parameter	Value
Transmitter (TX)	Sample Rate	200 kHz
	Samples per Symbol	8
	USRP B210 TX Gain	40
Receiver (RX)	Sample rate	500 kHz
	Bandwidth	500 kHz
	Intermediate Freq.	500 kHz
	MIGOU RX Gain	12
Common	Operating Frequency	868.025 MHz
	Antennas	VERT900 (Ettus Research)

TABLE 2. Dataset parameters.

Parameter	Value
Modulations	BPSK, QPSK, 8PSK, PAM4, QAM16, QAM64, GFSK, CPFSK, WBFM, AM-DSB, AM-SSB
Average SNRs	37 dB (at 1 m) and 22 dB (at 6 m)
Number of vectors per modulation-SNR pair	400,000
Vector shape	128×2

Specifically, measurements were taken at two distances from the transmitter, 1 and 6 meters, and were carried out indoors, in an office environment. Naturally, changes in the distance between devices, and in relation to other elements in their surroundings, affect the amplitude, noise level and propagation conditions of the recorded radio signals. Therefore, we also evaluated the robustness of the AMC methods to these factors. The average SNRs of the signals recorded at 1 and 6 meters are 37 dB and 22 dB, respectively. To calculate these values, the background noise in the channel was recorded under the same conditions, but without transmitting any signal. With that noise signal, the SNR of each modulated signal was calculated separately for both distances. Recordings of 2 seconds of the signals were used for this calculation. Finally, the SNRs of the 11 modulated signals were averaged for each distance. The difference in amplitude of raw signals recorded at both distances and normalized to the transceiver reception range can be seen in the top row of Fig.3.

All the recorded I/Q signals were divided and formatted into 128×2 vectors, which consist of 128 consecutive samples where the I/Q components of the samples have been separated. These vectors were individually normalized according to the following expression:

$$S_{i,j}^{N} = \frac{S_{i,j}}{\underset{\substack{k\in\{0,\ldots,127\}\\ l\in\{0,1\}}}{max}\left(|S_{k,l}|\right)} \quad \begin{matrix}\forall i \in \{0,\ldots,127\}\\ \forall j \in \{0,1\}\end{matrix}, \tag{3}$$

where S is the original vector, and S^N is the normalized vector. It should be noted that in a real implementation of an AMC method, the signal normalization process to prepare the input vector will be performed by the end-device. The result of applying this normalization to the raw I/Q signals can be seen in the bottom row of Fig.3. It can also be seen that the SNR of the signals recorded at 6 meters is lower.

Finally, 400,000 normalized vectors were included for each modulation-SNR (MOD-SNR) pair in the dataset, resulting in a total of 8.8 million vectors.

D. PROPOSED MODEL

In this work, we propose a model for AMC with two layers, as shown in Fig.4. The first one is a GRU layer with N cells. The input format of this layer is a $M \times 2$ vector of I/Q samples, where M is the number of samples and 2 corresponds to each of the I/Q components of a sample. These vectors are normalized to the range $[-1, 1]$, as explained in subsection III-C. The GRU layer applies the ReLU activation function and its output is a one-dimensional vector of length N formed by the last output of the output sequence of each cell. This vector is fed to the next layer.

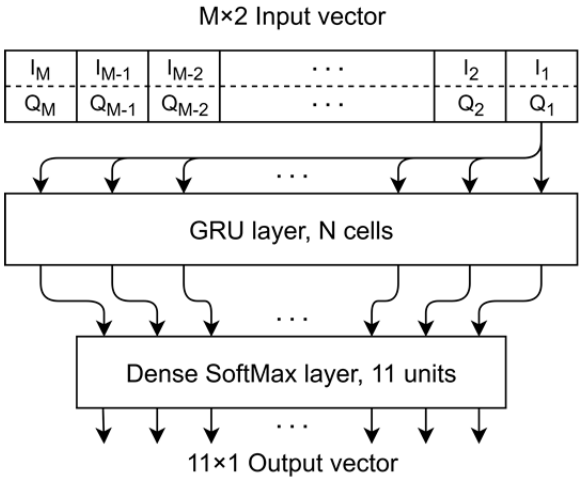

FIGURE 4. Our proposed GRU network model for AMC.

The second layer of our proposed AMC model is a dense SoftMax layer with 11 units. It maps the signal features extracted in the first layer to each of the 11 output classes that represent the possible modulation schemes.

To prevent overfitting problems, a dropout layer with a drop rate of 0.2 is placed between the two layers during the training phase. This technique is a common practice to address overfitting issues in neural network training [44]. Finally, the RMSprop optimizer, with the categorical cross entropy loss function, a learning rate of 0.001 and a clipnorm parameter of 0.1, is adopted for the model training.

The same activation functions, optimizer, and loss function as in the reference work were used for comparison purposes. However, the learning rate and the clipnorm parameter were empirically selected as they were not specified in that work.

E. IMPLEMENTATION DETAILS

The training and testing of the different models evaluated were performed with the Keras library using the TensorFlow backend. Specifically, the Keras 2.2.5 and TensorFlow 1.15.0 versions were used. In addition, its execution was carried out on a server equipped with an NVIDIA Graphics Processing Unit (GPU) with Compute Unified Device Architecture (CUDA), speeding up the process.

In all the tests, we employed the same number of vectors for the model training and for its validation. These vectors were randomly selected from the dataset and were not repeated. The set of vectors used for training is known as the training set.

All tests were conducted with 300 training epochs. However, a callback was used for early stopping when the validation loss value did not improve for T consecutive epochs. The parameter T is known as patience and, along with the batch size, it had to be empirically adjusted in some tests to avoid vanishing/exploding gradient problems. The configuration used in each case is detailed in the next section.

IV. RESULTS AND DISCUSSION

All tests presented in this section have been carried out using raw I/Q samples, which have only been normalized as described in subsection III-C. No synchronization or timing recovery has been performed.

The main metrics evaluated in the following subsections are the classification accuracy and the memory footprint. In all cases, accuracy is given for both distances, 1 and 6 meters. Moreover, when comparing our proposed model with the baseline one, an average of both is also used.

Since the memory required by a model is mainly due to the storage of its trainable parameters, its memory footprint is calculated as the number of trainable parameters of the model multiplied by the memory occupied by each parameter. In TensorFlow, the weights or trainable parameters are 32-bit floating point numbers, so each one occupies 4 bytes of memory.

A. BASELINE MODEL

First, we tested the baseline model with our dataset in order to replicate the results obtained by Hong *et al.* using the RadioML dataset. In this way, we established a reference with which to later compare our AMC model, and we validated our dataset generation method.

TABLE 3. Baseline model test parameters and results.

Dataset	RadioML2016.10a	Our dataset
MOD-SNR pairs	220	22
Number of vectors per MOD-SNR pair	500	5,000
Input vector size	128×2	128×2
Batch size	-	512
Early stopping patience	-	30
Training set size (vectors)	110 k	110 k
Classification accuracy at high SNRs	~85%	85.4%
Trainable parameters	38,079	38,079
Memory footprint (kBytes)	152.3	152.3

RadioML contains signals with 20 different SNRs for each of the 11 modulations, representing a total of 220 MOD-SNR pairs. Hong *et al.* trained their model with 500 vectors per each pair, which is a total of 110,000 vectors. Our dataset only has signals recorded at two different distances, i.e., with two different SNRs. Therefore, having 22 MOD-SNR pairs, we had to employ 5,000 vectors per pair in order to keep the same training set size, which is an important aspect in DL. The parameters and results of this test are shown in Table 3.

As can be seen, using the same training set size, the classification accuracy at high SNR obtained by the baseline model using our dataset is equivalent to the original. Specifically, the classification accuracy reached with our dataset at 6 meters was 84.6%, while at 1 meter it was 86.2%. This serves to validate the method and tools used to generate our dataset, and to validate the baseline model with real signal measurements. It also establishes a solid starting point for our subsequent study.

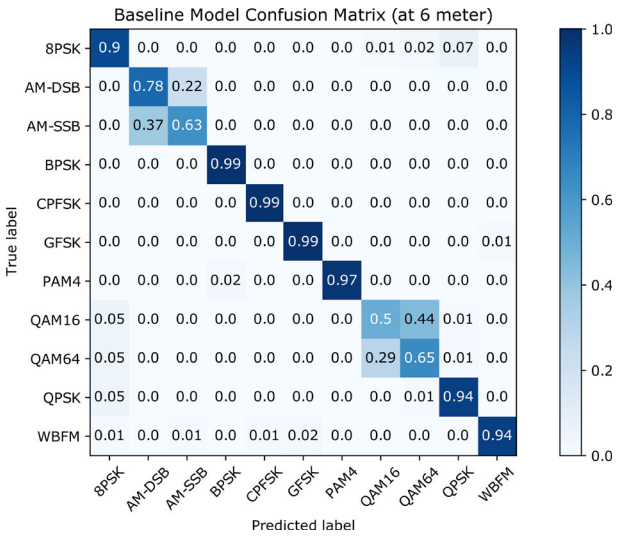

FIGURE 5. Confusion matrix of the baseline model trained with 110,000 vectors (at 6 meters).

Figure 5 shows the confusion matrix of the worst case, at 6 meters. As we can see, this model has difficulties in distinguishing between AM-DSB and AM-SSB modulations. This is because the source signal, which is an analog audio signal, has multiple silence periods in which both modulations continue to transmit the carrier. Thus, in these silence samples, these modulations are indistinguishable. The model also confuses QAM16 and QAM64 modulations. This is due to the constellation points of the former are contained in those of the latter. Both difficulties were also discussed by Hong *et al.*

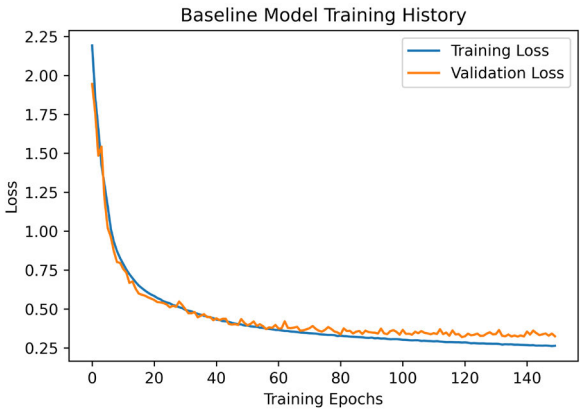

FIGURE 6. Training history of the baseline model trained with 110,000 vectors.

Figure 6 shows the training history of the baseline model with our dataset. This training took 150 epochs (47.5 minutes) and was stopped by the early stopping callback after 30 consecutive epochs in which the validation loss value did not improve. In addition, the Keras' ModelCheckpoint callback was also used to save the model weights with its best last result, in this case on epoch 120. Thus, the results in Table 3 correspond to this version of the model. During the last epochs, it can be observed that the model begins to experience overfitting, i.e., the training loss continues to decrease but not the validation loss. This indicates that what the model is learning with the training set does not generalize to the validation set, which are new samples. This fact led us to suspect that the potential of the model was not fully exploited with a training set of this size.

Finally, it should be noted that the baseline model has 38,079 trainable parameters, which entails a memory footprint of 152.3 kBytes.

B. TRAINING SET SIZE

In this test, we first evaluated the performance of our proposed AMC model under the same conditions as in the previous subsection, i.e., using a training set of 110,000 vectors randomly extracted from our dataset. Next, we progressively increased the training set size to assess whether our model can still learn more by using a larger number of sample vectors during training. In all the tests of this subsection, the validation sets used were the same size as the corresponding training set, and 128 × 2 input vectors were used in all cases.

Also, the GRU layer was configured with 100 cells, as the baseline model.

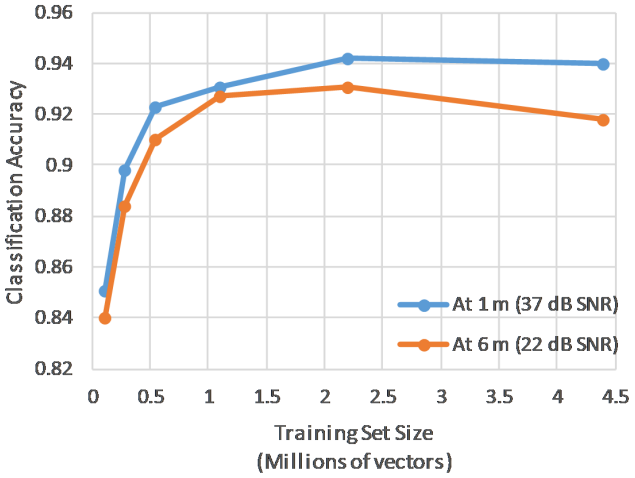

FIGURE 7. Classification accuracy for different training set sizes.

The classification accuracy results for each training set size are shown in Fig.7 and Table 4, which also includes the test parameters used in each case and the time spent in the training.

TABLE 4. Test parameters and results for different training set sizes.

Training set size	Batch size	Patience	Training time (minutes)	Classification accuracy (%)	
				At 1m	At 6m
110 k	512	30	52	85.1	84.0
275 k	512	30	189	89.8	88.4
550 k	512	30	328	92.3	91.0
1.1 M	1024	30	367	93.1	92.7
2.2 M	1024	12	494	94.2	93.1
4.4 M	2048	12	380	94.0	91.8

The average classification accuracy of our model with a training set size of 110,000 vectors is 84.55%, slightly less than the 85.4% obtained by the baseline model under the same conditions. This is the precision we have lost by removing the intermediate dense fully connected layer of the baseline model. In return, our model has fewer trainable parameters and, therefore, a smaller memory footprint. Specifically, with 100 GRU cells, our model has 32,011 trainable parameters, which entails a memory footprint of 128 kBytes, 24.3 kBytes less that the reference model.

If we observe the results with the different training set sizes, we realize that the potential of the model was not fully exploited with the initial size. We see that as we increment this size, the classification accuracy increases, reaching almost 10% improvement before saturating. These results demonstrate that to maximize the performance of a model and, therefore, not to oversize it without need, it is essential to have enough training data to verify that its capabilities have saturated.

On the other hand, increasing the size of the training set has a direct impact on the time required to train the model. However, this does not affect the inference time once the model is implemented and executed on a device. Finally, it should be noted that when the size of the training set was increased, hyperparameters such as batch size or patience had to be adjusted to stabilize the training process and avoid vanishing/exploding gradient problems.

C. INPUT VECTOR LENGTH

In this test, we evaluated the influence of the input vector length on the classification accuracy. This length corresponds to the number of I/Q samples that the vector contains, which is the amount of signal information that it has in order to infer the modulation. To illustrate this, the different lengths evaluated are represented over a modulated signal in Fig.8.

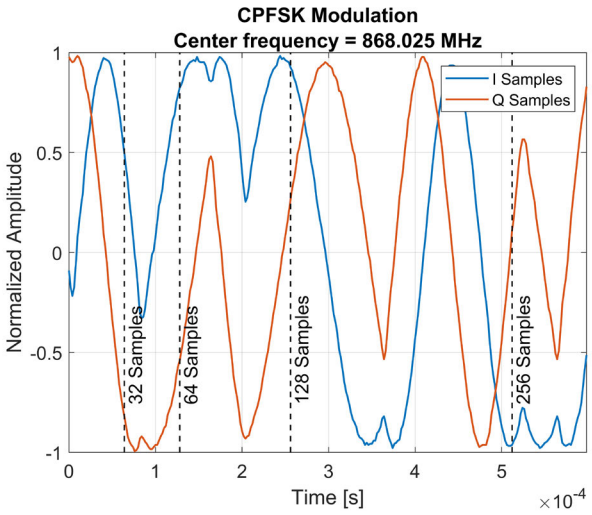

FIGURE 8. Different input vector lengths represented over a signal.

All tests carried out in this subsection use our AMC model with 100 cells in the GRU layer. In addition, the training set consisted of 70.4 million samples. This size results from multiplying the size of the training set, expressed in vectors, by the number of I/Q samples per vector. In order to maintain the same number of I/Q samples in the training set, a change in the input vector length must be compensated by adjusting accordingly the number of vectors in the training set. Additionally, for input vector lengths other than 128, the raw I/Q samples normalization process shown in (3) was repeated with the corresponding vector length, i.e., 32, 64 and 256 samples. These details and other test parameters are shown in Table 5. The classification accuracy results are depicted in Fig.9 and are also included in Table 5.

To perform a classification or, in general terms, an inference, the I/Q samples required to form an input vector must first be acquired. These samples are acquired at a fixed sampling rate. Then, they must be fed to the model, where they are processed to obtain a result. As explained

TABLE 5. Test parameters and results for different input vector lengths.

Input vector length (samples)	Training set size (vectors)	Batch size	Training time (minutes)	Classification accuracy (%)	
				At 1m	At 6m
32	2.2 M	1024	295	78.0	73.9
64	1.1 M	1024	270	87.3	86.4
128	550 k	512	328	92.3	91.0
256	275 k	512	246	93.1	92.3

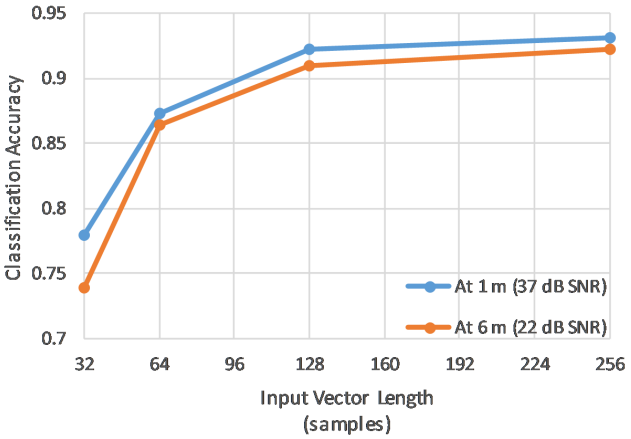

FIGURE 9. Classification accuracy for different input vector lengths.

in subsection III-B, at each time step, a sample of the input vector is fed simultaneously to all cells of the first layer. Therefore, a $M \times 2$ vector of I/Q samples requires M time steps to be fully fed to the model. Taking that into consideration, using an input vector with fewer I/Q samples reduces its acquisition and processing time, which will result in a reduction of the inference latency in the devices. However, by reducing the length of the input vector, the classification accuracy decreases, as can be seen in Fig.9. This decrease is especially pronounced for input vector lengths shorter than 128 samples. The selection of this length implies a tradeoff between inference speed and classification accuracy. Nevertheless, it should be noted that the accuracy saturates after a certain length value.

As seen in Table 5, in this case the differences in training time were not as significant as in the previous subsection. This is because although the length of the input vector varies, the number of I/Q samples in the training set is the same in all cases. In this test, it was also necessary to adjust the batch size to stabilize the training process. However, patience remained fixed at 30 epochs. Finally, it is worth mentioning that since the I/Q samples of the input vector are sequentially fed to the model, the length of this vector does not affect the number of trainable parameters of the model. Therefore, the memory footprint of the 100-cell model is still 128 kBytes. However, it should be noted that a longer input vector will temporarily require more device memory for its storage, but this memory can be dynamically allocated and deallocated at runtime.

TABLE 6. Test parameters and results for a different number of cells in the GRU layer.

GRU cells	Memory footprint (kBytes)	Patience	Training time (minutes)	Classification accuracy (%)	
				At 1m	At 6m
50	34.0	30	503	86.8	86.4
64	54.3	30	894	91.8	90.2
75	73.5	30	823	93.1	91.7
100	128.0	12	494	94.2	93.1
128	206.9	12	429	94.6	93.5

D. NUMBER OF CELLS IN THE GRU LAYER

In this test, we evaluated the influence of the number of cells in the GRU layer on the classification accuracy of the proposed model. For these experiments, we fixed the training set size to 2.2 million vectors, the input vector size to 128×2, and the batch size to 1024. As in the previous tests, patience was adjusted in each case to avoid training problems. The test parameters and results are shown in Table 6. In addition, the results are depicted in Fig.10.

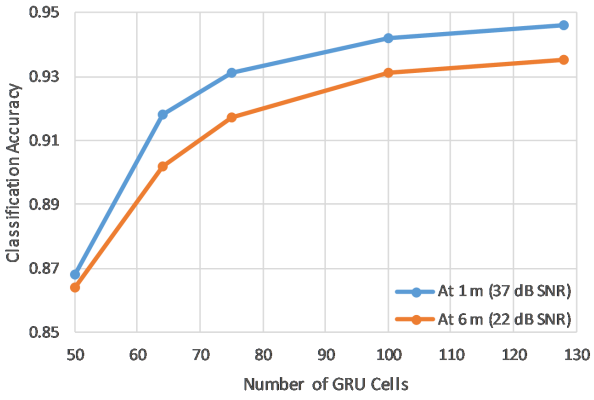

FIGURE 10. Classification accuracy for a different number of cells in the GRU layer.

As we can see, increasing the number of cells in the GRU layer improves classification accuracy. However, from a certain point, there are diminishing marginal benefits in increasing the number of GRU cells. In addition, it should be noted that increasing the number of cells also has an impact on the memory footprint; doubling the number of cells means an almost 4-fold increase in the memory footprint. This is an important fact to consider for the implementation of these solutions in resource-constrained devices.

E. SUMMARY

In this subsection, we have selected some of the previous experiments in order to summarize the procedure carried out to optimize an existing AMC model for its use in end-devices. All these experiments were performed with our dataset and with an input vector length of 128 I/Q samples. The classification accuracy and memory footprint of the models tested in the selected experiments are shown in Fig.11 and Fig.12, respectively.

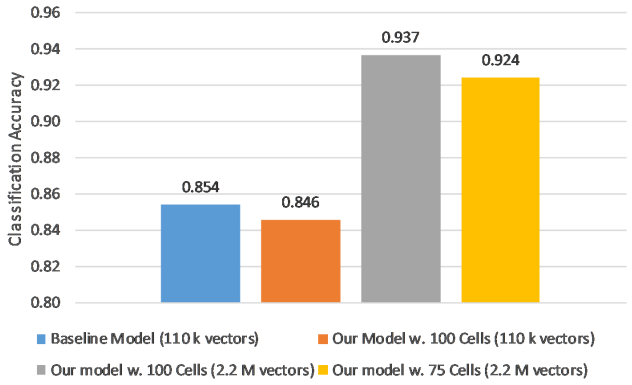

FIGURE 11. Classification accuracy for different representative AMC models and training set sizes.

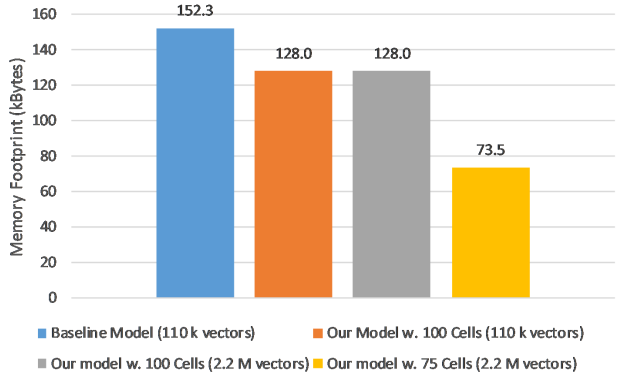

FIGURE 12. Memory footprint of different representative AMC models.

As a reference point, we have selected the experiment carried out in subsection IV-A, with the baseline model (100 GRU cells) and a training set size of 110,000 vectors, as the one originally used by Hong *et al.* If we compare these results with those of our model, keeping 100 cells in the GRU layer and the same size of the training set, we observe a 16% reduction in the memory footprint and only 1% in the classification accuracy. This is because our model eliminates the intermediate dense fully connected layer of the baseline model, which does not contribute significantly to accuracy.

The next aspect we observe is that the potential of our model is not fully exploited and that, by using a larger training set size (2.2 million vectors), it is possible to achieve a classification accuracy of almost 94%, compared to the 85% achieved with 110,000 vectors. As mentioned before, increasing the size of the training set has no effect on the memory footprint, it only increases the training time. Having a higher classification accuracy allows the developer to reduce the number of cells in the GRU layer while still obtaining good performance. This results in a reduction of the memory footprint. For example, using our AMC model with 75 GRU cells instead of 100 represents a reduction in the memory footprint of 42.6% and only 1.4% in the classification accuracy, which still remains above 90%.

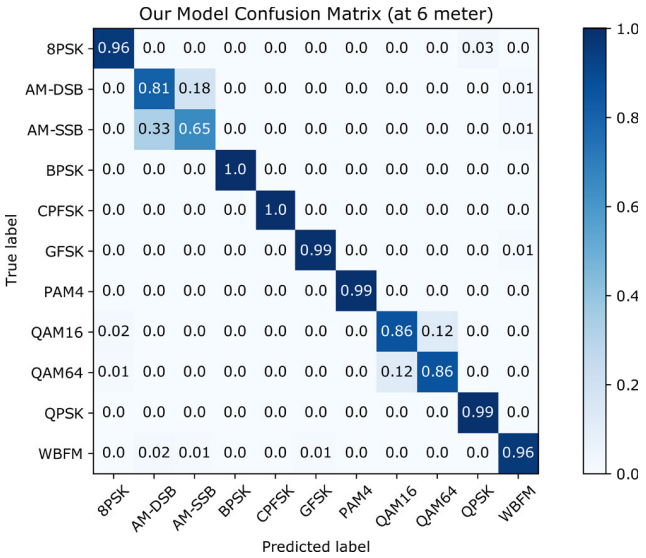

FIGURE 13. Confusion matrix of our model with 75 GRU cells and trained with 2.2 million vectors (at 6 meters).

The confusion matrix of this last model is shown in Fig.13. If we compare these results with those of the baseline model depicted in Fig.5, we observe that it is possible to improve the classification of AM and QAM modulations using a simpler model but trained with more I/Q sample vectors.

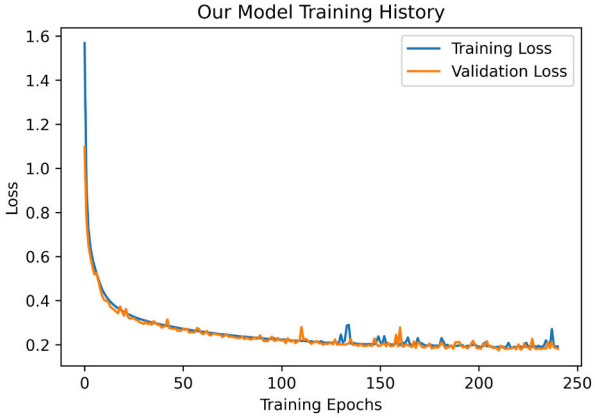

FIGURE 14. Training history of our model with 75 GRU cells and trained with 2.2 million vectors.

In addition, the training history of our 75-cell model is shown in Fig.14. The training of this model took 241 epochs (823 minutes) to complete, representing a 17-fold increase in training time compared to the baseline model, which took 47.5 minutes. This is the main cost of using a larger training set.

In this case, the training and validation loss curves jointly decrease to a point of stability with a minimal gap between their values. This means that the model is well fitted and performs the same with the training and validation sets, that is, no overfitting/underfitting problems occur.

Exploring all these tradeoffs between training time, performance, and resource utilization is essential when looking for DL-based solutions for low-resource end-devices. For example, the MIGOU platform has an embedded Non-Volatile Memory (eNVM) of 256 kBytes. If we consider this platform as the target device, the baseline model would occupy 59.5% of its memory, while our 75-cell model would occupy only 28.7%.

Allocating almost 30% of a device's memory to the AMC task may not be acceptable. In that case, other techniques such as post-training quantization should be explored. This technique converts the neural network weights from 32-bit floating point numbers to 8-bit integers with minimal classification accuracy losses [26]. This conversion reduces a model's memory footprint by a factor of four, which for our 75-cell model would suppose a memory footprint of 18.4 kBytes, i.e., 7.2% of MIGOU's eNVM.

The evaluation of these quantization techniques, as well as the implementation and evaluation of the AMC model in the MIGOU platform will be addressed in our future work.

V. CONCLUSIONS

In this work, we propose a new GRU neural network model for AMC that has been specifically designed for resource-constrained end-devices. This model consists of a GRU layer followed by a SoftMax layer. To come up with this model, a study was carried out on how multiple parameters affect the classification accuracy and the memory footprint of a reference model. The parameters evaluated were the size of the training set, the length of the input vector, the number of layers in the model, and the number of cells in each layer. As part of this work, a dataset has also been generated with over-the-air measurements of real radio signals acquired with MIGOU, a low-resource SDR experimental platform. All experiments have been performed with this dataset. Our results show that the proposed model has a memory footprint of 73.5 kBytes, 51.74% less than the reference model, and achieves a classification accuracy of 92.4%. The evaluation of quantization techniques to further reduce the model's memory footprint, as well as its implementation and evaluation on the MIGOU platform will be addressed in our future work.

This study has shown that the tradeoffs between training time, model performance, and resource utilization should be explored when developing neural network-based solutions for resource-constrained end-devices. Increasing the training set size, and consequently the training time, can lead to improvements in the performance of a model without the need to increase its complexity. In turn, these improvements may allow developers to reduce to some degree the complexity of the model and, therefore, the device resources it requires. However, longer training processes can also lead to diverse fitting and gradient problems and dealing with these problems require fine tuning of the hyperparameters, which can be a time-consuming task. The procedure carried out in this work provides good insights on how to optimize other DL-based solutions.

ACKNOWLEDGMENT

The authors would like to thank Alba Rozas, Julia Docampo, and Irene Azpiazu for their valuable comments and suggestions that greatly improved this article.

REFERENCES

[1] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017-2022*, Cisco Systems, San Jose, CA, USA, 2019.

[2] F. Wunsch, F. Paisana, S. Rajendran, A. Selim, P. Alvarez, S. Müller, S. Koslowski, B. Van den Bergh, and S. Pollin, "DySPAN spectrum challenge: Situational awareness and opportunistic spectrum access benchmarked," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 3, pp. 550–562, Sep. 2017.

[3] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[4] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[5] J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, and T. Melodia, "Machine learning for wireless communications in the Internet of Things: A comprehensive survey," *Ad Hoc Netw.*, vol. 93, Oct. 2019, Art. no. 101913.

[6] L. Zhang and Y.-C. Liang, "Joint spectrum sensing and packet error rate optimization in cognitive IoT," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7816–7827, Oct. 2019.

[7] J. Ploennigs, A. Ba, and M. Barry, "Materializing the promises of cognitive IoT: How cognitive buildings are shaping the way," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2367–2374, Aug. 2018.

[8] Z. Zhu and A. K. Nandi, *Automatic Modulation Classification: Principles, Algorithms and Applications*, 1st ed. Hoboken, NJ, USA: Wiley, 2014.

[9] Y.-S. Kuo, P. Pannuto, T. Schmid, and P. Dutta, "Reconfiguring the software radio to improve power, price, and portability," in *Proc. 10th ACM Conf. Embedded Netw. Sensor Syst.*, 2012, pp. 267–280.

[10] S. Szilvási, B. Babják, P. Völgyesi, and Á. Lédeczi, "Marmote SDR: Experimental platform for low-power wireless protocol stack research," *J. Sensor Actuator Netw.*, vol. 2, no. 3, pp. 631–652, Sep. 2013.

[11] R. Utrilla, R. Rodriguez-Zurrunero, J. Martin, A. Rozas, and A. Araujo, "MIGOU: A low-power experimental platform with programmable logic resources and software-defined radio capabilities," *Sensors*, vol. 19, no. 22, p. 4983, Nov. 2019.

[12] R. Utrilla. (2020). *MIGOU-MOD: A Dataset of Modulated Radio Signals Acquired With MIGOU, a Low-Power IoT Experimental Platform*. [Online]. Available: https://data.mendeley.com/datasets/fkwr8mzndr/1

[13] D. Hong, Z. Zhang, and X. Xu, "Automatic modulation classification using recurrent neural networks," in *Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, Dec. 2017, pp. 695–700.

[14] T. J. O'Shea and N. West, "Radio machine learning dataset generation with GNU radio," in *Proc. GNU Radio Conf.*, Sep. 2016, vol. 1, no. 1. [Online]. Available: https://pubs.gnuradio.org/index.php/grcon/article/view/11

[15] S. Rajendran and S. Pollin, "Large-scale wireless spectrum monitoring: Challenges and solutions based on machine learning," in *Spectrum Sharing: The Next Frontier in Wireless Networks*, 1st ed. Hoboken, NJ, USA: Wiley, 2020, pp. 321–339.

[16] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: Classical approaches and new trends," *IET Commun.*, vol. 1, no. 2, pp. 137–156, Apr. 2007.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[18] X. Li, F. Dong, S. Zhang, and W. Guo, "A survey on deep learning techniques in wireless signal recognition," *Wirel. Commun. Mobile Comput.*, vol. 2019, Feb. 2019, Art. no. 5629572.

[19] *GNU Radio—Free & Open-Source Software Development Toolkit*. Accessed: May 10, 2020. [Online]. Available: https://www.gnuradio.org/

[20] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional Radio Modulation Recognition Networks," in *Proc. 17th Int. Conf. Eng. Appl. Neural Netw. (EANN)*, 2016, pp. 213–226.

[21] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-Air deep learning based radio signal classification," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 168–179, Feb. 2018.

[22] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter, "End-to-End learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications," *IEEE Access*, vol. 6, pp. 18484–18501, 2018.
[23] S. Zhou, Z. Yin, Z. Wu, Y. Chen, N. Zhao, and Z. Yang, "A robust modulation classification method using convolutional neural networks," *EURASIP J. Adv. Signal Process.*, vol. 2019, no. 1, Mar. 2019.
[24] F. Meng, P. Chen, L. Wu, and X. Wang, "Automatic modulation classification: A deep learning enabled approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10760–10772, Nov. 2018.
[25] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Trans. Cognit. Commun. Netw.*, vol. 4, no. 3, pp. 433–445, Sep. 2018.
[26] M. Nagel, M. V. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1325–1334.
[27] Z.-L. Tang, S.-M. Li, and L.-J. Yu, "Implementation of deep learning-based automatic modulation classifier on FPGA SDR platform," *Electronics*, vol. 7, no. 7, p. 122, Jul. 2018.
[28] D. H. Noronha, B. Salehpour, and S. J. E. Wilton, "LeFlow: Enabling flexible FPGA high-level synthesis of TensorFlow deep neural networks," 2018, *arXiv:1807.05317*. [Online]. Available: http://arxiv.org/abs/1807.05317
[29] S. I. Venieris, A. Kouris, and C.-S. Bouganis, "Toolflows for mapping convolutional neural networks on FPGAs: A survey and future directions," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 1–39, Jul. 2018.
[30] M. Wielgosz and M. Karwatowski, "Mapping neural networks to FPGA-based IoT devices for ultra-low latency processing," *Sensors*, vol. 19, no. 13, p. 2981, Jul. 2019.
[31] Libelium Comunicaciones Distribuidas S.L. *Waspmote Datasheet*. Accessed: May 10, 2020. [Online]. Available: http://www.libelium.com/downloads/documentation/v12/waspmote_datasheet.pdf
[32] R. Rodriguez-Zurrunero, R. Utrilla, E. Romero, and A. Araujo, "An adaptive scheduler for real-time operating systems to extend WSN nodes lifetime," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–10, Oct. 2018.
[33] Microchip Technology. *ATSAMR21-XPRO Platform*. Accessed: May 10, 2020. [Online]. Available: https://www.microchip.com/DevelopmentTools/ProductDetails/PartNO/ATSAMR21-XPRO
[34] MEMSIC. *MICAz Wireless Measurement System*. Accessed: May 10, 2020. [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf
[35] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw. (IPSN)*, 2005, pp. 364–369.
[36] R. Rodriguez-Zurrunero, R. Utrilla, A. Rozas, and A. Araujo, "Process management in IoT operating systems: Cross-influence between processing and communication tasks in end-devices," *Sensors*, vol. 19, no. 4, p. 805, Feb. 2019.
[37] Z. Zuo, B. Shuai, G. Wang, X. Liu, X. Wang, B. Wang, and Y. Chen, "Convolutional recurrent neural networks: Learning spatial dependencies for image representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2015, pp. 18–26.
[38] M. Häsken and P. Stagge, "Recurrent neural networks for time series classification," *Neurocomputing*, vol. 50, pp. 223–235, Jan. 2003.
[39] H. Al-Askar, N. Radi, and Á. MacDermott, "Recurrent neural networks in medical data analysis and classifications," in *Applied Computing in Medicine and Health*, 1st ed. San Mateo, CA, USA: Morgan Kaufmann, 2016, pp. 147–165.
[40] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.
[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[42] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS*, 2014, pp. 1–7.
[43] Z. Wang, Y. Dong, W. Liu, and Z. Ma, "A novel fault diagnosis approach for chillers based on 1-D convolutional neural network and gated recurrent unit," *Sensors*, vol. 20, no. 9, p. 2458, Apr. 2020.
[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

RAMIRO UTRILLA received the M.Sc. degree in electronic systems engineering from the Universidad Politécnica de Madrid, in 2013, where he is currently pursuing the Ph.D. degree. He carries out research activities in the B105 Electronic Systems Lab, Universidad Politécnica de Madrid. His research interests include development of embedded systems and software-defined radio techniques to apply cognitive radio to low-power autonomous devices. He received the title of Telecommunication Engineer from the Universidad Autónoma de Madrid in 2011.

ERIKA FONSECA received the B.Sc. degree in telecommunications engineering and the M.Sc. degree in network computing from University Federal Fluminense, in 2013 and 2017, respectively. She is currently a Ph.D. Researcher with the CONNECT Research Centre for Future Networks and Communications, Trinity College Dublin. She has experience in wireless network research. Her research interests include 5G cellular networks, software-defined radio, and unmanned aerial vehicle (UAV) communication.

ALVARO ARAUJO received the Ph.D. degree from the Universidad Politécnica de Madrid, in 2007. He was a Postdoctoral Researcher with the Berkeley Wireless Research Center (BWRC), UC at Berkeley, working in a White Paper about Reliable Radio. He did a Research stay with BWRC, in 2018, working on human networks. He is currently an Assistant Professor with the Electronic Engineering Department, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, where he carries out both research and teaching activities. He is also the Director of the University-Company Chair Securitas Direct-Verisure and the Co-Director of the B105 Electronic Systems Lab. He has participated in more than 70 R+D+I projects funded in competitive tenders by public or private bodies, 21 as PI during the last eight years. His research has as results in three granted patents in exploitation phase, about 25 scientific publications in high-impact peer-reviewed international journals articles and more than 30 international and national conferences, including IWAAN, the IEEE PIRMC, and DySPAN. His main research interests include development of embedded systems focusing on security, wireless personal area network systems, cognitive radio, and wireless neural networks. He received the title of Telecommunication Engineer from the Universidad Politécnica de Madrid, in 2001.

LUIZ A. DASILVA (Fellow, IEEE) was the Chair of telecommunications with the Trinity College Dublin, where he was the Director of CONNECT, the Science Foundation Ireland Research Centre for Future Networks and Communications. He is currently the Executive Director of the Commonwealth Cyber Initiative. He is also a Professor with the Bradley Department of Electrical and Computer Engineering, Virginia Tech. He is also a Principal Investigator on research projects funded by the Science Foundation Ireland and the European Commission. He has contributed to cognitive networks and resource management in wireless networks. His research interests include distributed and adaptive resource management in wireless networks, and in particular radio resource sharing and the application of game theory to wireless networks.