Automatic Modulation Classification using LSTMs with an Alternative Input Shape

Alexander Gros
Electromagnetism and Telecommunications Department
Faculty of Engineering, UMONS
Mons, Belgium
alexander.gros@umons.ac.be

Aurélien Niébes
Electromagnetism and Telecommunications Department
Faculty of Engineering, UMONS
Mons, Belgium
aurelien.niebes@alumni.umons.ac.be

Véronique Moeyaert
Electromagnetism and Telecommunications Department
Faculty of Engineering, UMONS
Mons, Belgium
veronique.moeyaert@umons.ac.be

Patrice Mégret
Electromagnetism and Telecommunications Department
Faculty of Engineering, UMONS
Mons, Belgium
patrice.megret@umons.ac.be

***Abstract*—This paper investigates the impact of input representation and architecture on the performance of Long Short-Term Memory (LSTM) networks for Automatic Modulation Classification (AMC). Building on previous work, we reproduce and extend LSTM-based AMC models using the CSPB.ML.2018 dataset, introducing an alternative input shape that inverts the roles of time steps and features. This approach reduces training time by over 90% compared to conventional LSTMs and improves classification accuracy by up to 13.4% when using IQ inputs, particularly at low Signal-to-Noise Ratios (SNR). However, it requires longer input signal samples to match the performance of conventional models at high SNRs. We evaluate both cartesian (IQ) and polar input representations, demonstrating that IQ inputs consistently outperform polar inputs across all models. Our results highlight the trade-offs between computational efficiency, input length, and classification accuracy, offering insights for optimizing AMC systems in resource-constrained and real-time applications, such as cognitive radio and spectrum monitoring.**

Index Terms—Automatic modulation classification, Cognitive Radio, Deep Learning, Input Shape, Long Short-Term Memory, LSTM, Polar Transformation

I. Introduction

Automatic Modulation Classification (AMC), also known as Automatic Modulation Recognition (AMR), is the process of detecting and identifying the modulation scheme of a received radio signal, typically after signal detection and before demodulation. AMC plays a critical role in modern wireless communication systems, enabling both civilian and military applications. In the civilian domain, AMC supports spectrum monitoring for regulatory agencies, the generation of coverage maps for wireless operators, and the detection of unused frequency bands to mitigate interference. In cognitive radio systems, it allows receivers to dynamically adapt their modulation parameters without the need for additional signaling. In the military domain, AMC supports applications such as electronic warfare, surveillance, and drone detection. Traditional AMC techniques rely on expert designed feature extraction methods combined with classical machine learning algorithms. While effective in controlled conditions, these methods often degrade in performance under challenging environments such as high noise, multi-path fading, or frequency offsets. These limitations have motivated the adoption of deep learning approaches, which can automatically learn discriminative features directly from the signal data. Depending on the chosen input representation, such as raw temporal in-phase and quadrature (IQ) data, constellation diagrams, amplitude-phase data, fast Fourier transform (FFT) spectrograms, or advanced decompositions, deep models like convolutional neural networks (CNNs), long short-term memory networks (LSTMs), or hybrid architectures have shown significant improvements in classification accuracy [1], [2]. However, deep learning approaches introduce their own challenges, including the need for large labeled datasets and, in many cases, fixed length input sequences. In this work, we adopt the model architecture proposed by Raj et al. [3] but, to our knowledge, are the first to evaluate its performance on Spooner's CSPB.ML.2018 dataset [4], providing new insights into its generalization capabilities. We further analyze how input representations impact performance and discuss how deep learning, when combined with alternative input shapes, can improve computational efficiency of AMC systems.

This paper is organized as follows. Section II introduces the fundamental principles of LSTM networks. Section III presents the dataset used for training and validation. Section IV details the experimental setup and the three neural network architectures evaluated. Finally, section V discusses and compares the performance of these models.

II. LSTM: Modeling Long-Term Dependencies

Recurrent Neural Networks (RNNs) are designed for sequential data such as time series, text, or speech. At each step in a sequence, an RNN updates its hidden state based on the current input and the previous hidden state. Because the same function is applied at every step, an RNN can process sequences of varying length without retraining. Despite this flexibility, classical RNNs struggle to capture long-term dependencies, as gradients tend to vanish or explode during backpropagation through many steps. Long Short Term Memory networks (LSTMs), introduced by Hochreiter and Schmidhuber [5], address this limitation by adding an explicit

979-8-3315-8130-5/25/$31.00 ©2025 IEEE

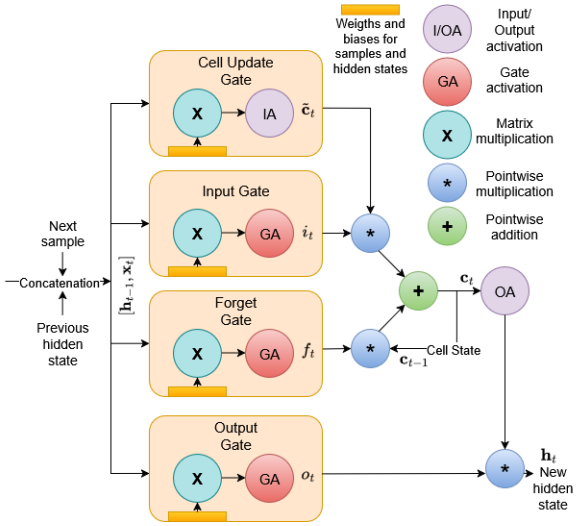


Fig. 1: Internal structure of a single LSTM cell. The four gates (forget, input, cell update, and output) regulate information flow through the hidden state and long-term cell state.

memory mechanism. The central idea is the cell state, a vector that acts as long-term memory and is updated through carefully controlled interactions. Information flow is regulated by gating functions that decide what to forget, what to store, and what to output. Several variants have been proposed, such as the Gated Recurrent Unit (GRU) [6] and Depth Gated LSTM [7]. In this work we describe the original LSTM cell [3]. Figure 3 shows the unrolled architecture, where the cell state runs alongside the hidden state. A single LSTM cell (Fig. 1) consists of four gates: the forget gate, the input gate, the cell update, and the output gate. Sometimes, the input gate and the cell update are grouped together as a single input operation. The behavior of the LSTM cell at time step t can be summarized as follows:

$$
\begin{aligned}
f_t &= \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_f) && \text{(forget gate)} \\
i_t &= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_i) && \text{(input gate)} \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_c) && \text{(cell update)} \\
\mathbf{c}_t &= f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t && \text{(new cell state)} \\
o_t &= \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_o) && \text{(output gate)} \\
\mathbf{h}_t &= o_t \odot \tanh(\mathbf{c}_t) && \text{(new hidden state)}
\end{aligned}
$$

In the above equations, $\mathbf{x}_t$ denotes the input vector, $\mathbf{h}_t$ the hidden state, $\mathbf{c}_t$ the cell state, $\mathbf{W}$ the weight matrices, b the bias terms, σ the sigmoid activation function, $\tanh$ the hyperbolic tangent function, and $\odot$ the element-wise (Hadamard) product.

This architecture allows LSTMs to preserve relevant information over long sequences, enabling them to model dependencies that classical RNNs cannot. As a result, LSTMs have become widely used in applications such as natural language processing, speech recognition, and time series forecasting.

III. Dataset

This work relies on the CSPB.ML.2018 dataset introduced by Spooner [4]. The dataset was synthetically generated in MATLAB by creating random message sequences whose symbols are independent and identically distributed. Each

TABLE I: CSPB.ML.2018 Dataset Parameters [4]

Parameter	Range / Values
Modulations	BPSK, QPSK, $\frac{\pi}{4}$-DQPSK, 8PSK, MSK, 16QAM, 64QAM, 256QAM
Base symbol period	1–15 samples
Carrier freq. offset (norm.)	$[-10^{-3}, 10^{-3}]$ cycles sample^{-1}
Roll-off factor	0.1–1
SNR	–2 to 12.8 dB
Up/down sampling	(1,1), (3,2),(4,3)→(10,9)
Noise spec. density	0 dB
Signal length	32,678 samples
Number of waveforms	112,000

waveform is then modulated and altered under a variety of conditions. The parameters varied include the symbol period, up and down sampling factors, carrier frequency offset (CFO), the raised cosine roll off factor, and the signal to noise ratio (SNR). The ranges of these parameters are summarized in Table I. Importantly, no channel effects such as fading (small or large scale) or multipath were applied, apart from the addition of Additive White Gaussian Noise (AWGN). The dataset contains only digital modulation formats: BPSK, QPSK, 8PSK, $\frac{\pi}{4}$-DQPSK, 16QAM, 64QAM, 256QAM, and MSK. Each signal sequence is 32,678 samples long, and the full dataset comprises 112,000 waveforms. The SNR values range from -2 dB to 12.8 dB, with a concentration around 10 dB. The deliberately long signal length serves two purposes. First, it ensures that models trained on this dataset have access to a wide variety of symbols within a single waveform, even when the number of samples per symbol is high. Second, but out of scope of this paper, it makes the dataset suitable for evaluation with expert cyclostationarity based classifiers, which require long observation windows to extract reliable cyclic features. In addition, the broad variation in signal parameters was designed to better reflect blind signal recognition scenarios, where communication settings such as CFO or Nyquist filters roll off factors are not known in advance. Overall, Spooner's dataset provides a challenging and diverse testbed for evaluating both neural and classical modulation recognition approaches, making it a robust benchmark for our study.

IV. Architecture and Data Flow

The input to all models considered in this study consists of complex IQ waveforms drawn from Spooner's dataset (see Section III). The task is to classify each signal into one of the eight modulation types present in the dataset. To evaluate the effectiveness of our proposed approach of alternative input shapes, we compare it against two widely used reference architectures from the literature: a convolutional neural network (CNN) originally proposed by O'Shea [8], and a two-layer Long Short Term Memory (LSTM) model from Rajendran et al. [9]. These serve as baselines and are further referred to as the "baseline CNN" and the "conventional LSTM model", respectively. With regard to our work, we introduce a modified LSTM variant (see Section IV-C) designed to improve performance under the challenging conditions of Spooner's dataset.

A. Baseline CNN Model

The baseline CNN model follows the architecture of O'Shea [8], which has become a standard reference for modulation recognition tasks. It consists of two convolutional

layers with 256 and 80 filters, respectively, using kernel sizes of 1×3 and 2×3. These convolutional layers are followed by a dense hidden layer with 256 units and an output layer with 8 neurons, corresponding to the 8 modulation classes in the dataset. The final layer applies a softmax activation to generate class probabilities. Convolutional networks are particularly effective at capturing local patterns and correlations within the input waveform. In the context of modulation recognition, this allows the CNN to exploit short-term spectral or temporal features without explicit feature engineering. A schematic representation of this architecture is provided in Fig. 2.

B. Conventional LSTM Model

The conventional LSTM model is based on the work of Rajendran et al. [9]. It consists of two stacked LSTM layers with hidden dimensions U_1 and U_2. Only the hidden state of the final cell in the second LSTM layer is forwarded to a dense classification layer with 8 output neurons with softmax activation function, producing a probability distribution over the classes. The architecture is illustrated in Fig.3. To improve generalization and reduce overfitting, dropout regularization with a rate of 0.5 is applied, following the recommendation of Srivastava et al. [10]. LSTM models are particularly well-suited for this task because they are able to capture temporal dependencies across long signal sequences, unlike CNNs which primarily exploit local features. This makes LSTMs a natural candidate for modulation recognition when long-range dependencies or temporal patterns play a role. The results of varying the parameters N, U_1, and U_2 are reported in Section V.

C. Alternative LSTM Model

In addition to the conventional model, we create and evaluate an alternative LSTM architecture, illustrated in Fig. 4. The key difference lies in how the input data is organized. Instead of treating each signal as a long sequence of complex IQ samples, the input is restructured as a sequence of two vectors: one containing the in phase (I) values and the other containing the quadrature (Q) values. Using the same notation as before, the length of these vectors is denoted by N. In this model, each LSTM layer in this model contains only two cells, one for I and one for Q, and each produces a hidden state of size U_1. This rearrangement changes the interpretation of the LSTM architecture. In the conventional

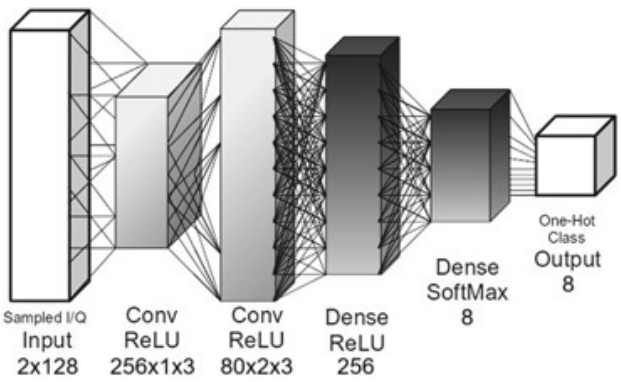


Fig. 2: Baseline CNN model based on O'Shea [8].

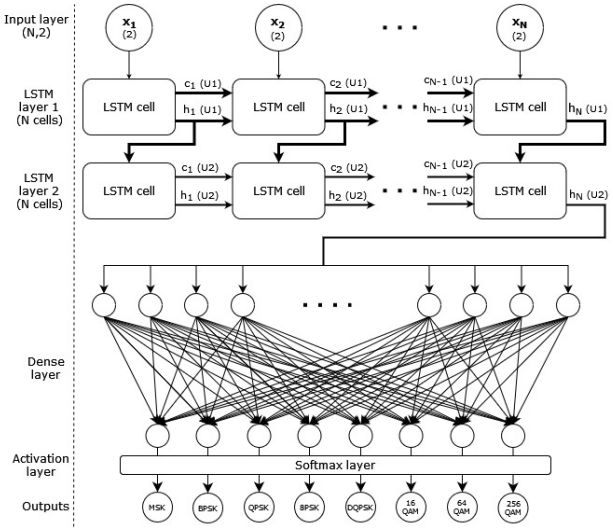


Fig. 3: Conventional two-layer LSTM model [9]. The input signal has temporal length N, and the hidden unit dimensions of the two LSTM layers are U_1 and U_2.

setup, the number of input features (I and Q) defines the dimensionality of each time step, while the sequence length corresponds to the number of samples processed in time. By contrast, in the alternative formulation the roles are inverted: the hidden state dimension is fixed to two, representing the I and Q components, while the unfolding takes place along the LSTM layers dimension. Conceptually, this transforms the model from a time domain sequence processor into a depth oriented structure. Instead of capturing long term temporal dependencies, the network behaves more like a deep feed forward architecture built from LSTM blocks, with I and Q acting as the core internal states. This perspective highlights the flexibility of recurrent structures and allows us to test whether such a reparameterization can capture discriminative features of modulation waveforms more effectively than the conventional LSTM model.

D. Training and Evaluation

For our experiments, 70% of the dataset is used for training and the remaining 30% for validation. Data is managed through a generator that dynamically assembles mini batches of 128 waveforms. Furthermore, waveforms are randomly shuffled between epochs to improve generalization and reduce the risk of overfitting. The code will be made publicly available on GitHub [11]. Model selection is performed by monitoring the validation loss: training continues until convergence, and the model parameters corresponding to the epoch with the lowest validation loss are saved. All models are trained using categorical cross entropy as the loss function. Optimization is carried out with the Adam algorithm [12], a stochastic gradient based method with a default learning rate of 0.001. LSTM cells are implemented using the standard Keras library, with the forget gate bias initialized to one. Weight initialization follows the Glorot uniform scheme [13]. Because the dataset is large (28.7 GB), direct loading into memory is impractical. Instead, training is performed using a data generator. This generator dynamically extracts mini batches from disk, rearranges dimensions when required, and

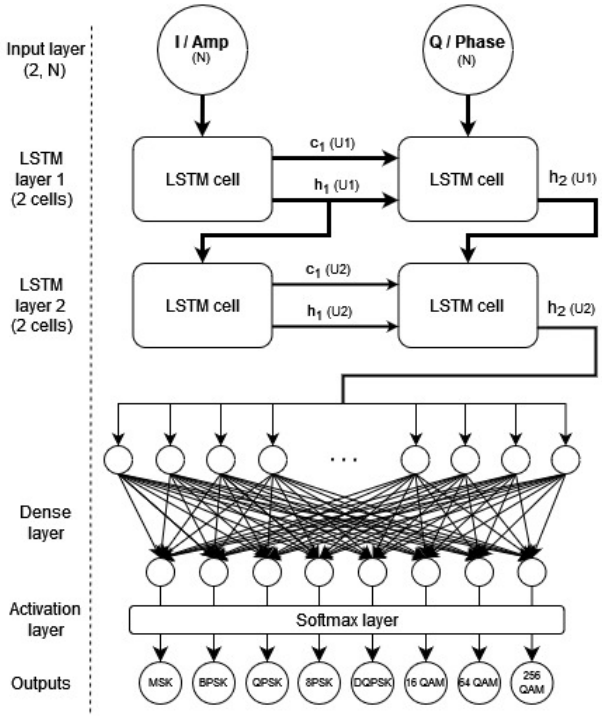


Fig. 4: Alternative two-layer LSTM model. The input signal has temporal length N, and the hidden unit dimensions of the two LSTM layers are U_1 and U_2.

optionally converts the input signals to polar coordinates (see section V-A). This streaming approach minimizes memory requirements while preserving flexibility in data representation.

V. Results and Analysis

In this section we present the experimental results obtained with the models introduced in Section IV. We evaluate classification performance under different conditions and analyze the impact of input representation, input length, and hidden state size on validation accuracy. In addition, we compare the conventional and alternative LSTM models, as well as the reference CNN baseline. Our implementation of the baseline architectures follows the original publications. The CNN model is taken from O'Shea [8], while the conventional LSTM model is based on Rajendran et al. [9]. The publicly available code from [8] is already implemented in Keras with TensorFlow as backend, whereas the code from [9] was originally written with the older TFLearn wrapper. For consistency, we reimplemented this model using Keras. Figure 5 illustrates the validation accuracy of the alternative LSTM model with IQ inputs for different input lengths (from 64 to 4096 first samples) across a range of SNR values. Similarly, Fig. 6 compares validation accuracy for IQ and amplitude–phase input representations, alongside the CNN baseline, using input sequences of length 128. Unless otherwise noted, the gap between training and validation accuracies across all models remains small (Table II, III), indicating that overfitting is not a significant concern in our experiments. Performance values are summarized in the tables provided in the following subsections. Figures 7a, 7b, and 7c present the confusion matrices for the CNN, alternative LSTM, and standard LSTM architectures, respectively. These matrices

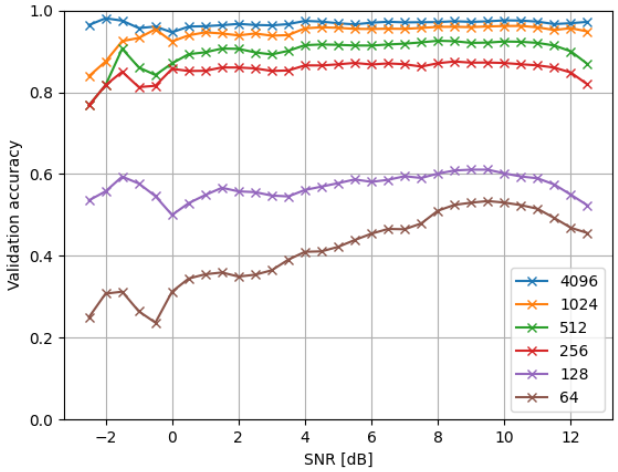


Fig. 5: Validation accuracy in function of the SNR for multiple instances of the "alternative" LSTM model with IQ inputs with different input lengths

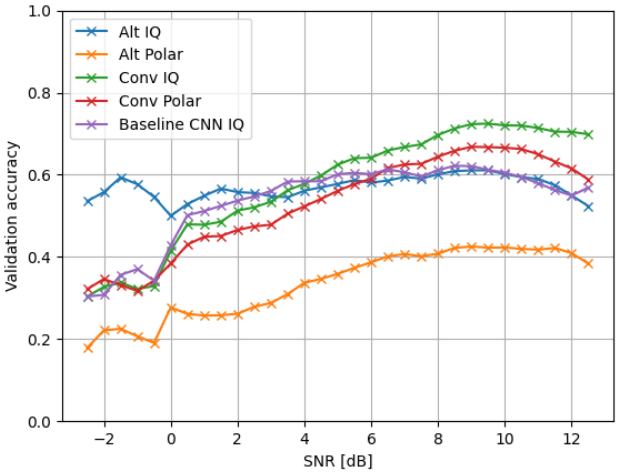
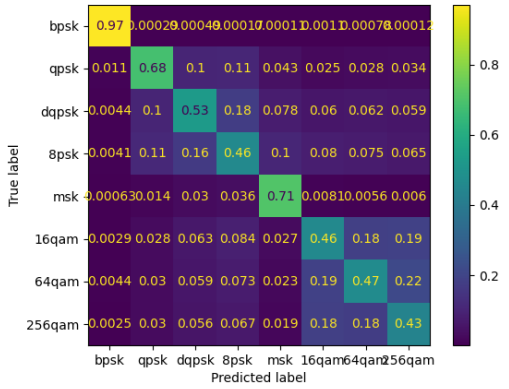


Fig. 6: Validation accuracy in function of the SNR for the LSTM models with IQ and amplitude-phase inputs and the baseline CNN model for input sample lengths of 128

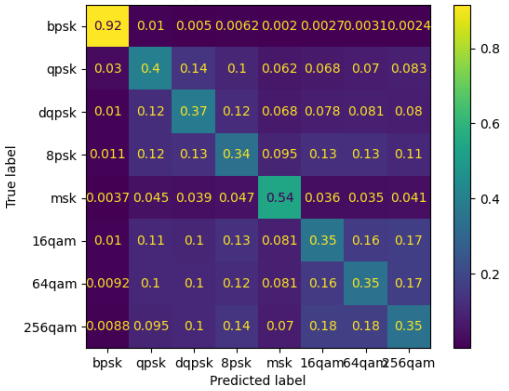
illustrate the overall classification performance across all SNR levels using IQ inputs with a fixed length of 128 samples. Notably, the limited input length appears to constrain the performance of the alternative LSTM model, highlighting the sensitivity of its architecture to shorter input sequences.

A. Discussion: IQ vs Polar Input

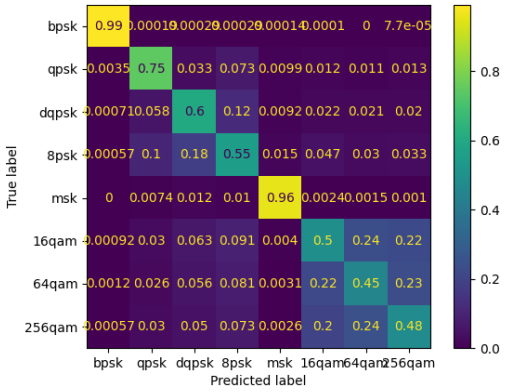
A central question in modulation classification is whether signals should be represented in Cartesian (In-phase and Quadrature) or polar (amplitude and phase) coordinates before being fed into the network. Previous studies have reported mixed results, often depending on the dataset and network architecture [9], [14]. We first evaluate both input formats for our three architectures using sequences of length 128. The results are summarized in Table II, and the validation accuracy as a function of SNR is shown in Fig. 6. For both the conventional and alternative LSTM models, IQ inputs consistently outperform polar inputs. The average improvement is approximately 3.1% for the conventional model and 13.4% for the alternative model, with the advantage being most pronounced in the latter case. Interestingly, this finding



(a) CNN model



(b) Alternative LSTM Model



(c) Conventional LSTM Model

Fig. 7: Confusion matrices for models trained on IQ inputs with a length of 128 samples (validation set)

contradicts the results of Rajendran et al. [9], who reported that, comparatively to polar inputs, LSTM models performed poorly with IQ inputs, in some cases failing to exceed random chance. The discrepancy can be attributed to differences in the datasets: our experiments use Spooner's dataset, which includes additional modulations, wider parameter variation, and carrier frequency offset (CFO). The presence of CFO makes the amplitude–phase relationship less stable, which may explain why polar representations are less effective in this context. Since the superiority of IQ inputs persists across longer input lengths, the following sections focus exclusively on results obtained with IQ representations. These results suggest that IQ signals preserve more discriminative information under blind estimation conditions and are therefore better suited for neural network based modulation classification.

B. Conventional vs Alternative Model

As shown in Fig. 6 and Table II, the conventional LSTM model consistently achieves higher validation accuracy than the alternative model when both are configured with the same input length and hidden state dimension. The alternative model, however, exhibits a more stable performance across SNR values, which makes it comparatively stronger under low-SNR conditions. A further advantage of the alternative design is its computational efficiency: for 128-sample inputs, the conventional model requires on average 380–401 seconds per epoch ($\approx$ 6 minutes), whereas the alternative model trains in only 18–19 seconds per epoch. This represents a speedup of more than one order of magnitude, making large-scale experiments feasible with the alternative architecture. Nevertheless, this gain in efficiency comes at the cost of lower classification accuracy, as the conventional model remains superior across most SNR levels.

C. Effect of Input Length using a Hypermodel

To enable comparison with prior work by Rajendran and O'Shea, we first evaluate models using input sequences of 128 samples. This choice, however, is not ideal, since such short sequences often fail to cover even a single full modulation symbol. Increasing the number of samples provides more information for feature extraction, but also enlarges the model and raises computational cost. Thanks to the alternative model short training time in respect to the conventional model, we could try larger input lengths, up to 4096 samples, both for IQ and polar inputs. The results of a first analysis with IQ inputs, represented on Fig. 5, showed that, for the alternative model, longer inputs lead to better validation accuracy. But this figure also shows that this improvement saturates and tapers off for long inputs. Longer inputs also lead to longer training time as shown in Table III for IQ inputs. However, when polar inputs with 4096 samples are fed to the model, the performance is extremely poor, with a validation accuracy of only 12.5%, which is equivalent to random chance. The optimizer doesn't seem to be able to reduce the validation cost nor the training cost. Furthermore, nearly all the signals are classified as DQPSK. Keras provides HyperModels to streamline hyperparameter optimization. A HyperModel defines both the model architecture and the search space for parameters such as layer size, activation function, or learning rate. Combined with a tuner, it enables automated exploration of different configurations to identify the best-performing model without manual trial and error. To further analyze this phenomenon, we used a hypermodel called "Hyperband" that trained several instances of the alternative model with inputs randomly chosen between 64 and 4096 samples for 20 epochs each. Fig. 8 summarizes the results of the hypermodel. The validation accuracy peaks at 56.78% for an input length around 2250. The performances drop sharply for input lengths above 3200 samples.

D. Physical Interpretation of the Alternative Input Shape

The proposed alternative input shape fundamentally reinterprets the temporal processing paradigm used in conventional LSTM based modulation classification. In traditional approaches, the input signal is treated as a long sequence of time steps, where each step corresponds to a pair of

TABLE II: Results of the Models with 128 Samples Long Inputs

	CNN	Conv LSTM (IQ)	Conv LSTM (Polar)	ALT LSTM (IQ)	ALT LSTM (Polar)
Number of Epochs	7	33	24	20	16
Epoch Avg. Duration [s]	182	380	401	18	19
Training Accuracy [%]	58.88	66.05	60.61	50.25	35.56
Validation Accuracy [%]	52.50	60.31	57.20	46.68	33.24

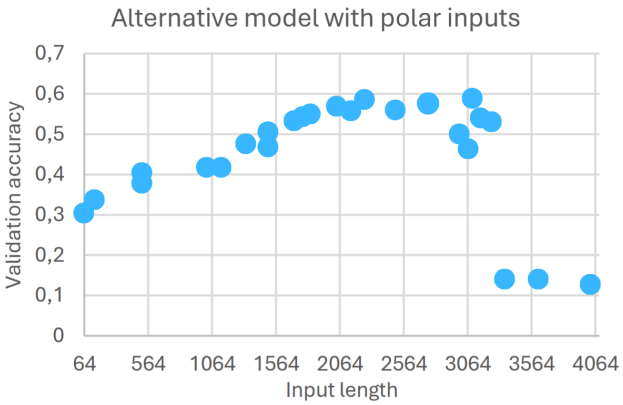


Fig. 8: Alternative LSTM model architecture for polar inputs, evaluated as a function of input sample length

TABLE III: Performance for varying input lengths (Alternative model)

Input Length	Validation Acc.	Training Acc.	Time (min)
4096	90.42%	91.09%	184
1024	85.76%	95.49%	31
512	73.34%	88.62%	6
256	60.87%	77.84%	6
128	46.68%	50.25%	6

in phase (I) and quadrature (Q) samples. This requires the LSTM to perform recurrent operations across all time steps, which becomes computationally expensive as the sampling frequency increases due to the large number of sequential dependencies to model. In contrast, our method inverts the roles of time steps and features. The input is restructured such that only two time steps are used, one for the I component and one for the Q component, while the features correspond to the entire sequence of sampled amplitudes. This transformation effectively converts the LSTM into a deep feature extractor, where each hidden layer processes the relationship between I and Q across all time steps simultaneously. This approach offers two key advantages:

- Computational efficiency: By reducing the number of recurrent steps from N (the sequence length) to just 2, the training time is significantly decreased (see Section V for quantitative comparisons).
- Enhanced feature representation: The increased number of features, that is, time steps as features, allows the model to capture finer grained dependencies between I and Q components. This can mitigate early overfitting, though standard regularization techniques (e.g., dropout) are still applied to ensure robustness.

From a physical perspective, this architecture directly examines the constellation dynamics of the modulation scheme. Instead of sequentially analyzing I/Q pairs, the model evaluates how I and Q amplitudes covary across the entire signal, providing implicit insights into the modulation constellation geometry. While this design accelerates training, it may initially yield lower classification accuracy for a fixed model size, as the reduced temporal depth limits long range dependency modeling. However, by increasing the number of hidden layers, the model can achieve comparable performance to conventional LSTMs, as the additional layers compensate for the reduced sequential processing.

VI. Conclusion

This study aimed to investigate how input representation and neural network architecture influence the performance of LSTM-based Automatic Modulation Classification systems, with a particular emphasis on balancing computational efficiency and classification accuracy in challenging environments. Using the CSPB.ML.2018 dataset, we introduced and evaluated an alternative input shape that inverts the conventional roles of time steps and features in LSTM networks. Our results demonstrate that Cartesian IQ inputs consistently outperform polar representations, especially under low SNR and in the presence of carrier frequency offset. This finding underscores the suitability of IQ inputs for AMC tasks in blind estimation scenarios. The proposed alternative input shape significantly reduced training time by over ninety percent while improving classification accuracy by up to thirteen point four percent at low SNRs. This approach effectively transforms the LSTM into a deep feature extractor, enabling it to capture more nuanced relationships between in-phase and quadrature components. Although this method offers substantial computational advantages and enhanced robustness in low-SNR conditions, it requires longer input sequences to achieve comparable accuracy to conventional models at high SNRs. While these findings provide valuable insights for optimizing AMC systems in applications such as cognitive radio and spectrum monitoring, practical deployment will require further validation beyond simulation. Real-world impairments, including multipath fading, dynamic interference, and hardware constraints, are not fully represented in synthetic datasets and must be addressed to ensure operational reliability. Future research should focus on hybrid architectures, adaptive input lengths, and real-world testing to better align simulated performance with practical effectiveness. This work contributes to the advancement of efficient and adaptable AMC systems, offering a foundation for their integration into dynamic and resource-constrained wireless environments.

Acknowledgment

This work is supported by the Walloon Region research project ”CyberExcellence”, n° 2110186. Computational resources have been provided by the ”Consortium des Equipements de Calcul Intensif (CECI)”, funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.- FNRS) under Grant n° 2.5020.11, and by the Walloon Region.

REFERENCES

[1] A. Gros, V. Moeyaert, and P. Megret, "Joint use of bivariate empirical mode decomposition and convolutional neural networks for automatic modulation recognition," in *2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2022, pp. 957–962.

[2] ——, "Enhancing modulation classification through lightweight dyadic down-sampling schemes and cnn layer fusion," in *2025 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2025.

[3] 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 Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, 2018.

[4] C. Spooner, "Dataset for the machine-learning challenge [cspb.ml.2018]," https://cyclostationary.blog/2019/02/15/data-set-for-the-machine-learning-challenge/, accessed: 02.09.2025.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] 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," Sep. 2014, arXiv:1406.1078 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1406.1078

[7] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer, "Depth-Gated LSTM," Aug. 2015, arXiv:1508.03790 [cs] version: 4. [Online]. Available: http://arxiv.org/abs/1508.03790

[8] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional Radio Modulation Recognition Networks," in *Engineering Applications of Neural Networks*, C. Jayne and L. Iliadis, Eds. Cham: Springer International Publishing, 2016, pp. 213–226.

[9] 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 Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, Sep. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8357902/

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[11] A. Gros, "Amc_lstm," https://github.com/AlexanderGros/AMC_LSTM, 2025, (Source code).

[12] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6980

[13] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, Mar. 2010, pp. 249–256, iSSN: 1938-7228. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html

[14] A. Ahmed, B. Quoitin, A. Gros, and V. Moeyaert, "A comprehensive survey on deep learning-based lora radio frequency fingerprinting identification," *Sensors*, vol. 24, no. 13, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/13/4411