Knowledge Distillation for Modulation Classification in Resource-Constrained Devices

Pedro Marcio Raposo Pereira
Inatel
Santa Rita do Sapucaí, Brazil
pedro.marcio@inatel.br

***Abstract*—Automated Modulation Classification (AMC) is crucial in electronic warfare (EW) for enhancing situational awareness and enabling prompt responses to hostile transmissions. This research addresses the challenges associated with deploying AMC in limited computational environments by proposing the use of response-based knowledge distillation (KD) to train compact yet accurate AMC models. The methodology involves a framework that integrates a Signal-Based Convolutional Neural Network (SBCNN) and an Image-Based Convolutional Neural Network (IBCNN). The SBCNN extracts features from preprocessed signal data, which are subsequently used to train the IBCNN. Experimental results indicate that the SBCNN-based approach, when trained with teacher distillation, achieves superior performance compared to its standalone counterpart. The findings suggest that KD has significant potential to enhance AMC performance in real-time applications by balancing computational demands with classification accuracy.**

***Index Terms*—Real-time Signal Classification, Knowledge Distillation, Deep Learning, Computational Efficiency, Feature Extraction.**

I. Introduction

Military applications for Automatic Modulation Classifier (AMCs) are vital in Electronic Warfare (EW), enhancing situational awareness and enabling quick responses to hostile transmissions. Acting as an intermediary between signal detection and system reaction, AMCs identify and classify signals without prior knowledge of parameters like carrier frequency or bandwidth [1]. This capability is crucial for detecting, locating, and identifying enemy transmissions, facilitating timely and precise jamming or countermeasures.

Classification algorithms primarily follow two approaches: likelihood-based (LB) and feature-based (FB). LB algorithms, or decision-theoretic approaches, optimize classification using the likelihood function of received signals but often face high computational complexity, posing challenges for real-time implementation [2]–[4]. As a result, simplified versions are typically used to balance computational demands with accuracy. Conversely, FB algorithms use a pattern recognition approach, extracting and processing signal features for classification. Despite being sub-optimal, FB algorithms offer simpler implementation and lower computational overhead [3].

Deep Learning (DL) has garnered attention for its ability to achieve robust classification performance, once it is able to identify and adapt to diverse modulations present in heterogeneous environments. For instance, A novel data preprocessing method aimed at enhancing the efficiency of Convolutional Neural Networks (CNN)-based classification by optimizing the form of signal samples is proposed in [5], resulting in a significant 10% improvement in accuracy compared to traditional CNN approaches.

Additionally, a CNN architecture incorporating residual blocks is proposed, achieving a maximum accuracy of 93.7% on the RadioML2016.10a. In the context of resource-constrained devices, a high-efficiency AMC is proposed in [6]. Leveraging stacking quasi-recurrent neural network (S-QRNN) layers for feature extraction, the architecture integrates convolutional layers for low latency and a recurrent pooling function for enhanced classification accuracy. Evaluation against state-of-the-art classifiers demonstrates superior efficiency, with the proposed S-QRNN classifier exhibiting, on average, a 75.83% higher efficiency and a 26.54% lower execution latency. Additionally, by introducing gated recurrent units (GRUs) for temporal feature extraction, the classifier achieves an average efficiency increase of 191% compared to existing models, with a 59.31% lower execution latency on average.

Signal-Based Convolutional Neural Networks (SBCNN) [7] and Image-Based Convolutional Neural Networks (IBCNN) [8] represent two complementary approaches within the realm of DL-based AMC. In SBCNN, the focus lies on extracting relevant features from preprocessed signal data through a CNN. The extracted features from pre-training SBCNN are then transformed into images to train the IBCNN model.

While the integration of SBCNN and IBCNN offers potential benefits in terms of classification accuracy, it also introduces certain challenges. One notable concern with the adoption of a double-stage CNN approach is the increase in model size. Combining SBCNN and IBCNN into a single framework results in a larger and more complex model architecture. The increased number of parameters and layers may demand higher processing capability, both in terms of computational power and memory resources, leading to higher system latency. In scenarios where computational resources are limited, deploying such large models could pose significant challenges, potentially leading to performance degradation or operational constraints. Moreover, the larger model size can exacerbate system latency, which is a critical factor in many real-time applications. The additional computational burden imposed by the double-stage CNN may lead to longer processing times, delaying the classification of incoming signals.

This work aims to demonstrate the effectiveness of response-based knowledge distillation (KD) in addressing computational constraints in AMC. Response-based KD facilitates the training of compact yet accurate models by transferring knowledge from larger, more complex models while maintaining high accuracy [9], [10]. Specifically, the proposed single-stage SBCNN exhibits improved performance when trained using teacher distillation compared to the standalone training approach. This approach allows for the creation of AMC models that are better suited for deployment on resource-constrained devices.

The rest of the article is divided as follows. Section II details the dataset preprocessing. Section III shows how both SBCNN and IBCNN models are implemented. Section IV describes the distillation process and how the SBCNN-based based is constructed. Section V presents a discussion about the results and, finally, Section VI concludes the paper.

II. Data Preprocessing

The DeepSig Dataset: RadioML 2018.01A [11] is a publicly available dataset of wireless communication signals from 24 different modulations: On-Off Keying (OOK), m-level Amplitude Shift Keying (m-ASK) with $m = \{4, 8\}$, m-level Phase Shift Keying (m-PSK) with $m = \{2, 4, 8, 16, 32\}$, m-level Amplitude and Phase Shift Keying (m-APSK) with $m = \{16, 32, 64, 128\}$, m-level Quadrature Amplitude Modulation (m-QAM) with $m = \{16, 32, 64, 128, 256\}$, Amplitude Modulation Single Side Band with Carrier (AM-SSB-WC), Amplitude Modulation Single Side Band Suppressed Carrier (AM-SSB-SC), Amplitude Modulation Double Side Band with Carrier (AM-DSB-WC), Amplitude Modulation Double Side Band Suppressed Carrier (AM-DSB-SC), Frequency Modulation (FM), Gaussian Minimum Shift Keying (GMSK), Offset Quadrature Phase Shift Keying (OQPSK). It contains over 2.5 million frames, each of which is a 1024-sample sequence of complex time-series data. The Signal-to-Noise Ratio (SNR) levels range from -20 dB to +30 dB in steps of 2 dB.

The dataset is divided into three groups: 80% for training, 10% for validation, and 10% for testing for all SNR ranging from 0 dB to 16 dB. For the training and validation datasets, all SNR values are grouped. This ensures that the model is trained and validated on a diverse set of data with a wide range of SNR levels. For the testing dataset, separate datasets are created for each SNR value. This allows us to evaluate the performance of the model on each SNR level individually.

The data preprocessing pipeline consists of three steps: First, a root-mean-square (RMS) normalization is performed to ensure that all signals are on a similar scale, regardless of their absolute power level; Then, the data is transposed so that the outer dimensions of the data are the real and the imaginary parts. The last step adds a dimension to the end of the data so that it has three dimensions in total, i.e. (time, channel, 1).

III. Model

The SBCNN network architecture, shown in Figure 1, is a convolutional neural network (CNN) composed of seven

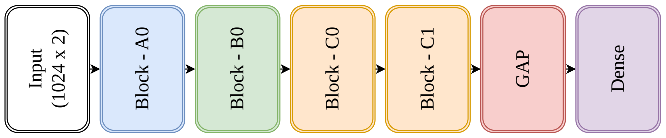

Fig. 1. SBCNN Block Diagram.

blocks in sequence: an input layer, one A-type block, one B-type block, two C-type blocks, Global Average Pooling (GAP) layer, and a Dense layer.

Each A-type, B-type, and C-type block is composed of a stack of shared Convolutional Block (CB), each of which consists of a Convolutional layer, a Batch Normalization (BN) layer, and a ReLU activation layer. The Convolutional layer performs a pointwise 2D convolution to extract features from the input feature map. The BN layer normalizes the output of the convolution block to improve the stability of the network and speed up the training process. The ReLU activation layer introduces non-linearity into the network to allow it to learn more complex relationships between the features. A detailed description of each block is presented in Figure 2.

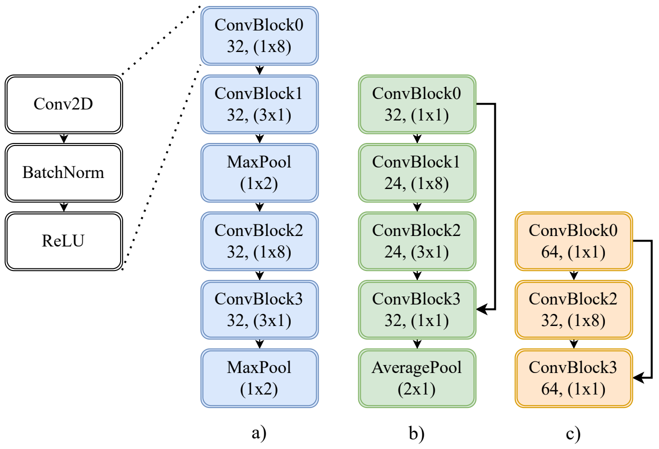

Fig. 2. SBCNN Blocks architectures. Blocks a) A0, b) B0, and c) C0 and C1.

The A-type block consists of two CB, both with 32 filters and a kernel size of 1×8 and 3×1 respectively, followed by a max-pooling layer with pool size 1×2. This sequence is performed twice, with two more CB and a max-pooling layer. The max-pooling layer downsamples the feature map by reducing its spatial dimensions to extract more general features. The B-type block is similar to the A-type block, but it uses an average-pooling layer instead of a max-pooling layer at its end and includes a skip connection between the first and fourth CB. The first CB has 32 filters and a kernel size 1×1, and the second one has 24 filters and a kernel size 3×1. The average-pooling layer has a pool size 1×2. The C-type block is similar to the B-type block, but it does not have the average pooling layer and has one less CB. The first and the least CB have 64 filters and a 1×1 kernel size. The intermediate block has 32 filters and 1×8 kernel size. The GAP layer completely compresses the feature map size to 1×1, which is then transferred to the Dense layer. In other words, it reduces the spatial dimensions of the feature map to zero while preserving the channel dimensions, resulting in a single vector of feature values that is fed into the fully connected layer. The Dense layer then performs a linear combination of these feature values to produce the final output, a vector of probability scores for each class.

The conversion from the output of the SBCNN to the input format suitable for the IBCNN is accomplished through a series of operations. Firstly, a quantization layer is applied to the input tensor, ensuring that its values fall within a specified range defined by the minimum and maximum bounds. Here, the minimum bound is set to -128 and the maximum to 127, with 8 bits allocated for quantization. Next, a normalization layer is introduced, which scales the quantized values to a range between 0 and 1. Finally, a reshape layer is employed to reshape the tensor into the desired format for IBCNN input, effectively converting it into an image-like tensor with a single channel. Note that the conversion method adopted here is different from [8].

The IBCNN architecture follows a similar design to the one adopted for the SBCNN as depicted in Figure 3. It is composed of the same seven blocks, with the addition of a dropout layer, introduced to mitigate overfitting, by pruning 20% of hidden units. The block types of IBCNN are illustrated in Figure 4.

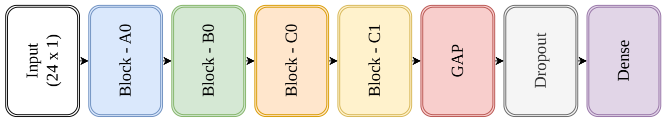

Fig. 3. IBCNN Block Diagram.

The A-type block incorporates three ConvBlocks with 32, 64, and, 32 filters and kernel sizes of 3×1, 1×1, and 3×1, respectively, followed by downsampling via max-pooling layer with a 2×1 pool size. In contrast to SBCNN, the B-type block of IBCNN exhibits modifications, such as the removal of one CB from the horizontal aspect with 1×8 kernel size and the usage of max-pooling. The structure of the C-type blocks 0 and 1 remains consistent with that of SBCNN. For the C0 block, the filter sizes remain the same, while the kernel size of the central CB is changed to 3×1. The C1 block has doubled the filter sizes of the C0 block, while the kernel sizes remain the same.

IV. KNOWLEDGE DISTILLATION

KD is a technique used to transfer knowledge from a large, complex model, often referred to as the "teacher", to a smaller, more compact model, the "student". This process is useful when the teacher model is computationally expensive or impractical to deploy in real-world applications. The goal of KD is to distill the knowledge encoded in the teacher model into the student model, enabling the student to achieve comparable performance while being more lightweight and efficient. Figure 5 illustrates how KD is processed during the

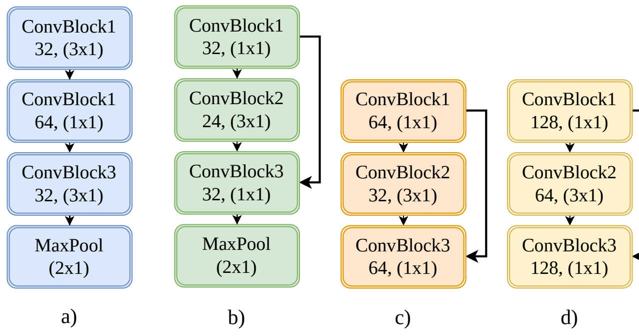

Fig. 4. IBCNN Blocks architectures. Blocks a) A0, b) B0, c) C0 and d) C1

student's training. Consider a dataset D comprising N samples

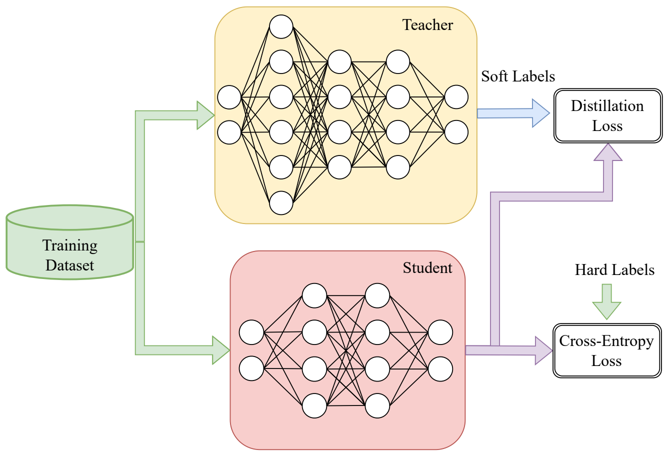

Fig. 5. KD training process.

denoted as $\{x_n, y_n\}_{n=1,\ldots,N}$, where x_n represents the n-th target data and y_n corresponds to its label within a range $[1, K]$. Given an input pair $\{x_n, y_n\}$, both a teacher model f_T and a student model f_S generate logit vectors, v_n and $z_n \in \mathbb{R}^{1\times K}$, where $z_n = f_S(x_n)$ and $v_n = f_T(x_n)$.

The logit vectors v_n and z_n are transformed into probability distributions $q(v_n)$ and $q(z_n)$, respectively, using a softmax function parameterized by temperature T. This transformation ensures that the probabilities in these distributions are appropriately scaled and normalized. The softmax function computes the probability of each class, ensuring that the probabilities sum up to one. Specifically, for each sample, the softmax function computes $q(v_n)(k)$ and $q(z_n)(k)$, representing the probability of the k-th class, where

$$q(z_n)(k) = \frac{\exp(z_n(k)/T)}{\sum_{m=1}^{K} \exp(z_n(m)/T)}, \tag{1}$$

and

$$q(v_n)(k) = \frac{\exp(v_n(k)/T)}{\sum_{m=1}^{K} \exp(v_n(m)/T)}. \tag{2}$$

KD aims to transfer knowledge from the teacher model to the student model by aligning their output probability distributions. This alignment is achieved by minimizing the

Kullback-Leibler (KL) divergence between the probability distributions $q(v_n)$ and $q(z_n)$. The KL divergence measures the difference between the two distributions, quantifying how much information is lost when using $q(z_n)$ to approximate $q(v_n)$, also called soft labels. Thus the loss function is set to minimize the KL divergence given by

$$L_{KL}(q(v_n)||q(z_n)) = \sum_{k=1}^{K} q(v_n)(k) \log\left(\frac{q(v_n)(k)}{q(z_n)(k)}\right). \quad (3)$$

The student model also undergoes independent learning, as it is tasked with classification duties. This independent learning is guided by a loss function, specifically the cross-entropy loss, denoted as

$$L_{CE}(y_n, q(z_n)) = -\sum_{k=1}^{K} y_{n,k} \log q(z_n)(k), \quad (4)$$

this loss function quantifies the discrepancy between the predicted probability distribution $q(z_n)$ and the true labels y_n, also called hard labels, for each sample n. Consequently, the total loss is formulated as

$$L_{KD}(y_n, q(v_n), q(z_n)) = \alpha L_{CE} + (1-\alpha) L_{KL} T^2, \quad (5)$$

where α acts as a weighting factor for the student and distillation losses. The first component refers to the cross-entropy with the hard targets. This is computed using the logits from the softmax function at a temperature $T = 1$ in (1). Meanwhile, the second component denotes the loss with soft targets. Here, the softmax functions in (1) and (2) are computed using the same high-temperature $T \geq 1$. Since the gradients produced by the soft targets scale as $1/T^2$, it's essential to scale them by T^2 when utilizing both hard and soft targets. This ensures that the relative contributions of the hard and soft targets remain consistent, even if the temperature used for distillation varies during parameter tuning [9]. Generally, better results are achieved by assigning a significantly lower weight to the hard target's objective function.

It's essential to emphasize that in this framework, gradients are solely computed for the student model, as the teacher model has already been trained on the specific task. This ensures that the training process focuses on refining the student's capabilities while leveraging the knowledge distilled from the teacher model.

The student model in this study is implemented using a simplified SBCNN architecture. Aiming to reduce the model's complexity, the student model is composed of a single A-type block, a GAP layer, and a final dense layer for classification.

V. RESULTS

In this section, we present the results obtained while evaluating the proposed model. The necessary codes for reproducing the project are publicly available[1]. The implementation of the SBCNN and IBCNN utilized the TensorFlow and Keras API. The training and testing processes were conducted on a machine equipped with a Ryzen 5600H and an RTX 3050 Laptop GPU. Both models were trained with a batch size of 64 samples over 20 epochs. The Adam optimizer was adopted with a learning rate of 0.001. At the end of each epoch, the weights that performed best on the validation dataset, measured by accuracy, were saved, and the training data was reshuffled for the next epoch. The Sparse Categorical Cross Entropy loss function was chosen, with labels modified to integer values for better efficiency, although originally stored as one hot. Additionally, a softmax function was not applied to the network's output, requiring the loss function to handle logits values.

To distill the knowledge from the IBCNN model to the student model, the best-performing parameters found through a non-exhaustive search are $\alpha = 0.35$ and $T = 1$. The distillation was carried over 20 epochs with a batch size of 64 samples, also using the Adam optimizer with a learning rate of 0.001. To evaluate the advantage of the KD, a second student model was trained from scratch using the same setup as described above.

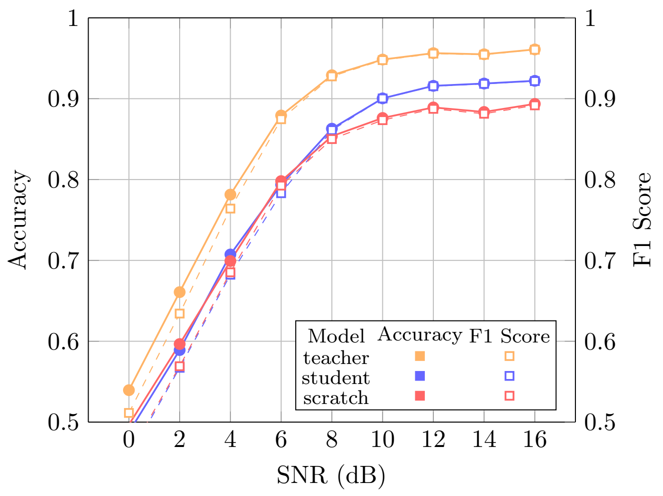

Fig. 6. Comparison between the teacher model, the student model trained by the teacher, and the student model trained from scratch. The x-axis represents the SNR levels ranging from 0 to 16 dB. The left y-axis shows the accuracy, while the right y-axis displays the F1 score.

Figure 6 compares the results of the teacher (IBCNN), distilled student, and the student trained from scratch over the SNR range in the test set. The left y-axis represents accuracy, while the right y-axis represents the F1 score. Each model's performance is plotted with a distinct color. Solid lines denote accuracy, and dashed lines denote the F1 score. The student model shows a steady increase in both accuracy and F1 score as the SNR improves, starting from around 0.49 accuracy and 0.46 F1 score at 0 dB, and reaching about 0.92 accuracies and F1 score at 16 dB. However, when trained from scratch, the model follows a similar trend but starts slightly higher, at approximately 0.50 accuracy and 0.46 F1 score at 0 dB, peaking at around 0.89 accuracy and 0.89 F1 score at 16 dB. The teacher model exhibits the best performance among the

[1]https://github.com/ErosloursViv/TP558/projeto_final

TABLE I
CLASSIFICATION REPORT COMPARISON OF STUDENT, STUDENT TRAINED FROM SCRATCH, AND TEACHER MODELS

Class	Student			Student from Scratch			Teacher		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
OOK	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
4ASK	0.87	0.97	0.91	0.92	0.93	0.93	0.94	0.95	0.94
8ASK	0.96	0.85	0.90	0.93	0.92	0.93	0.95	0.94	0.94
BPSK	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
QPSK	1.00	0.96	0.98	0.98	0.97	0.97	0.98	1.00	0.99
8PSK	0.96	0.82	0.88	0.99	0.76	0.86	0.89	0.90	0.90
16PSK	0.98	0.49	0.65	0.57	0.90	0.70	0.98	0.63	0.77
32PSK	0.59	0.96	0.73	0.83	0.51	0.63	0.69	0.92	0.79
16APSK	0.96	0.83	0.89	0.95	0.84	0.90	0.94	0.91	0.93
32APSK	0.92	0.83	0.87	0.64	0.94	0.76	0.94	0.89	0.92
64APSK	0.87	0.55	0.68	0.88	0.49	0.63	0.96	0.66	0.78
128APSK	0.76	0.61	0.67	0.64	0.60	0.62	0.79	0.80	0.79
16QAM	0.59	0.86	0.70	0.65	0.81	0.72	0.86	0.86	0.86
32QAM	0.85	0.58	0.69	0.60	0.66	0.63	0.89	0.73	0.80
64QAM	0.70	0.37	0.48	0.66	0.30	0.41	0.94	0.48	0.64
128QAM	0.37	0.66	0.47	0.40	0.51	0.45	0.47	0.81	0.59
256QAM	0.44	0.57	0.49	0.44	0.47	0.45	0.57	0.73	0.64
AM-SSB-WC	0.73	0.82	0.77	0.73	0.78	0.75	0.71	0.92	0.80
AM-SSB-SC	0.79	0.69	0.74	0.76	0.70	0.73	0.89	0.62	0.73
AM-DSB-WC	0.73	0.82	0.77	0.72	0.87	0.79	0.73	0.87	0.80
AM-DSB-SC	0.79	0.70	0.74	0.83	0.66	0.74	0.84	0.68	0.75
FM	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
GMSK	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
OQPSK	0.93	1.00	0.96	0.94	1.00	0.97	0.97	1.00	0.99
accuracy		0.79			0.78			0.85	
macro avg	0.82	0.79	0.79	0.79	0.78	0.77	0.87	0.85	0.85

three. It begins at around 0.54 accuracy and 0.51 F1 score at 0 dB and achieves nearly 0.96 accuracy and F1 score at 16 dB. All models improve as the SNR increases, with the IBCNN consistently outperforming the other two models in both accuracy and F1 score across all SNR levels. Notably, the student model trained by the teacher performs significantly better than when it was trained by itself.

Table I presents a comparison of the classification performance of three models. Performance metrics include precision, recall, and F1-score for all modulation schemes across the test dataset. As expected, the teacher model exhibits superior performance, particularly noticeable in the precision and recall values. For example, the teacher model achieves an F1-score of 0.94 for 8ASK, significantly higher than the 0.90 and 0.93 scored by the student and scratch models, respectively. The student model, which benefits from the teacher's knowledge, generally performs better than the student model trained from scratch. For instance, the student model achieves a recall of 0.96 and an F1-score of 0.98 for QPSK, outperforming the scratch model's recall of 0.97 and F1-score of 0.97. This trend is consistent across other classes, underscoring the efficacy of the KD paradigm. However, in certain classes such as AM-SSB-WC and AM-DSB-WC, the student model trained from scratch shows competitive performance, with F1-scores of 0.75 and 0.79, respectively, compared to the student model's 0.77 and 0.77. This suggests that while the KD approach generally yields better results, the student model trained from scratch can sometimes achieve comparable outcomes, thus highlighting the importance of the hyperparameter α to balance the distillation process.

Figure 7 (a) and (b) displays the confusion matrix of the student model at a SNR of 6dB and 16dB, respectively. The plot shows a diagonal pattern of higher values, indicating strong performance in correctly classifying instances within their respective classes for both SNR values. Conversely, off-diagonal elements highlight areas of confusion, where misclassifications occur between certain classes. For instance, there are noticeable off-diagonal values between m-QAM and m-ASK modulations with higher orders. Also within similar techniques such as AM-SSB-WC to AM-SSB-SC, and AM-DSB-WC to AM-DSB-SC. This indicates potential challenges in distinguishing these modulation types accurately, especially at low SNR, since increasing the SNR seems to mitigate the problem. At 16dB, while the confusion between certain classes persists, the overall performance improves as indicated by fewer off-diagonal values and higher diagonal concentrations.

VI. CONCLUSION

This study has demonstrated the effectiveness of response-based KD in training compact AMC models. The proposed single-stage SBCNN model, when trained with teacher distillation, consistently outperforms the student model trained from scratch across various SNR levels. The distilled student model generally achieves higher precision, recall, and F1 scores than the scratch model. For example, the distilled student model achieves an F1-score of 0.98 for QPSK, outperforming the scratch model's 0.97. However, the scratch model shows competitive performance in certain modulation schemes like AM-SSB-WC and AM-DSB-WC, suggesting that the effectiveness

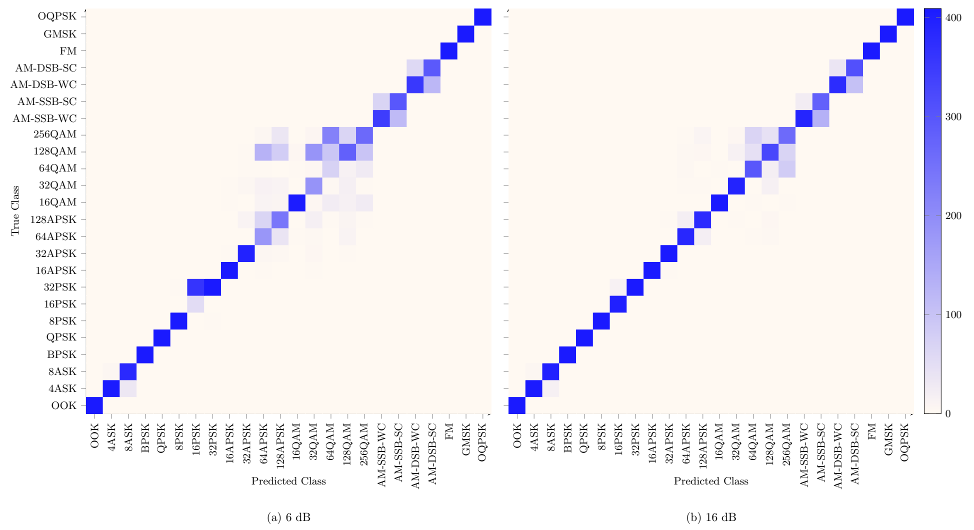

Fig. 7. Figures (a) and (b) display the confusion matrix of the classifier at a SNR of 6dB and 16dB, respectively. The plot shows a diagonal pattern of higher values, indicating strong performance for both SNR values. Conversely, off-diagonal, such as between modulations with higher orders and similar techniques, suggests potential challenges, especially at low SNR.

of KD can vary depending on the modulation type, thus highlighting the importance of proper parameter optimization. The student model can correctly classify most instances at different SNR levels, although some confusion persists between similar modulation types, especially at lower SNRs. This indicates areas for improvement in model robustness and signal differentiation. Future research should focus on optimizing the parameters α and temperature T to further enhance the KD process. Additionally, employing more robust models like the XGBoost classifier as teachers and incorporating preprocessing methods such as PCA (Principal Component Analysis) analysis could further improve performance.

References

[1] V. Iglesias, J. Grajal, and O. Yeste-Ojeda, "Automatic modulation classifier for military applications," in *2011 19th European Signal Processing Conference*, 2011, pp. 1814–1818.

[2] Y. Yuan, P. Zhao, B. Wang, and B. Wu, "Hybrid maximum likelihood modulation classification for continuous phase modulations," *IEEE Communications Letters*, vol. 20, no. 3, pp. 450–453, 2016.

[3] X. Yan, G. Zhang, and H.-C. Wu, "A novel automatic modulation classifier using graph-based constellation analysis for M -ary qam," *IEEE Communications Letters*, vol. 23, no. 2, pp. 298–301, 2019.

[4] A. Kumar, S. Majhi, G. Gui, H.-C. Wu, and C. Yuen, "A survey of blind modulation classification techniques for ofdm signals," *Sensors*, vol. 22, no. 3, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/3/1020

[5] H. Zhang, M. Huang, J. Yang, and W. Sun, "A data preprocessing method for automatic modulation classification based on cnn," *IEEE Communications Letters*, vol. 25, no. 4, pp. 1206–1210, 2021.

[6] P. Ghasemzadeh, M. Hempel, and H. Sharif, "Gs-qrnn: A high-efficiency automatic modulation classifier for cognitive radio iot," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9467–9477, 2022.

[7] S.-H. Kim, J.-W. Kim, V.-S. Doan, and D.-S. Kim, "Lightweight deep learning model for automatic modulation classification in cognitive radio networks," *IEEE Access*, vol. 8, pp. 197 532–197 541, 2020.

[8] S.-H. Kim, C.-B. Moon, J.-W. Kim, and D.-S. Kim, "A hybrid deep learning model for automatic modulation classification," *IEEE Wireless Communications Letters*, vol. 11, no. 2, pp. 313–317, 2022.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[10] Z. Li, H. Li, and L. Meng, "Model compression for deep neural networks: A survey," *Computers*, vol. 12, no. 3, 2023. [Online]. Available: https://www.mdpi.com/2073-431X/12/3/60

[11] "Radioml 2018.01a," https://www.deepsig.ai/datasets/, accessed: April 16, 2024.