LOW-PRECISION DEEP-LEARNING-BASED AUTOMATIC MODULATION RECOGNITION SYSTEM

Satish Kumar[1], Aakash Agarwal[1], Neeraj Varshney[2], Rajarshi Mahapatra[1]
[1] Department of Electronics & Communication Engineering, International Institute of Information Technology, Naya Raipur, India, [2] Wireless Networks Division, National Institute of Standards and Technology, Gaithersburg, MD 20899 USA

NOTE: Corresponding author: Satish Kumar, satishkumarvatsa@gmail.com

Abstract – *Convolution Neural Network (CNN)-based deep learning models have recently been employed in Automated Modulation Classification (AMC) systems, with excellent results. However, hardware deployment of these CNN-based AMC models is very difficult due to their large size, floating point weights and activations, and real-time processing requirements in hardware such as Field Programmable Gate Arrays (FPGAs). In this study, we designed CNN-based AMC techniques for complex-valued temporal radio signal domains and made them less complex with a small memory footprint for FPGA implementation. This work mainly focuses on quantized CNN, low precision mathematics, and quantization-aware CNN training to overcome the problem of larger model sizes, floating-point weights, and activations. Low precision weights, activations, and quantized CNN, on the other hand, have a considerable impact on the accuracy of the model. Thus, we propose an iterative pruning-based training mechanism to maintain the overall accuracy above a certain threshold while decreasing the model size for hardware implementation. The proposed schemes are 21.55 times less complex and achieve at least 1.6% higher accuracy than the baseline. Moreover, results show that our convolution layer-based Quantized Modulation Classification Network (QMCNet) with pruning has 92.01% less multiply-accumulate bit operations (bit_operations), 61.39% less activation bits, and 87.58% less weight bits than the 8 bit quantized baseline model whereas the quantized and pruned Residual-Unit based model (RUNet) has 95.36% less bit_operations, 29.97% less activation bits and 98.22% less weight bits than the 8 bit quantized baseline model.*

Keywords – Automatic modulation classification, convolution neural network, FPGA, iterative pruning, quantization-aware training

1. INTRODUCTION

Automatic Modulation Classification (AMC) is a key technique in frequency spectrum monitoring, software-defined radio, interference detection, cognitive radio networks, spectrum management, network traffic control, and electronic warfare etc. among other applications [1, 2, 3, 4, 5]. Different signal processing techniques can only be applied once the modulation of the signal is known. Without prior knowledge of the signal power, offset in frequency and phase, timing error, and carrier frequency etc., identification of the modulation scheme remains challenging [1, 2, 3]. Two crucial aspects in the creation of a modulation classifier are choosing the right classification technique and signal preprocessing [2]. The statistical Machine Learning (ML) approaches and the maximum likelihood-based theoretical decision strategy are both used to find the modulation scheme [1, 3]. The maximum likelihood approaches, in theory, minimize the joint probability of errors and provide optimal results when used with Bayesian methods. Furthermore, maximum likelihood approaches require prior knowledge of the actual probability density functions of noise and received signal. Not only that, but phase jitter, erroneous channel state information, uncertainty in the probability density functions of noise and received signal, frequency offset, and residual channel effect all have a significant impact on maximum likelihood approaches. However, the ML approaches, on the other hand, use feature extraction to classify patterns [1, 2, 3, 4, 5]. The accuracy of these ML-based classifiers such as the support vector machine and the K-Nearest Neighbor is determined by the selection of features. It is difficult to choose features with high discriminant qualities, which in turn degrade the overall performance of ML-based classifiers [6]. Moreover, the computational complexity of these approaches increases exponentially as the number of users grows [1, 3]. Whereas in the Deep Learning (DL) model, the features are automatically extracted in abstract form at every layer. This removes the need for manual feature selection and hence enhances overall accuracy of the system.

Recent advancements in Convolutional Neural Network (CNN)-based DL techniques reduce the difficulties of AMC while increasing its accuracy. These DL models with huge CNN layers show the remarkable capacity of feature learning on raw multidimensional data with supervised objectives. Furthermore, these CNN-based DL techniques reduce the difficulties of AMC while increasing its accuracy. With the ever increasing number of wireless devices in today's era, it becomes important that these CNN-based AMC systems be implemented in hardware such as Field Programmable Gate Arrays (FPGAs) and Radio Frequency Systems-on-Chip (RFSoCs). Despite the amazing performance of DL networks, due to very high computing and storage requirements of convolutional and fully connected layers [7, 8], the implementation of such CNN-based AMC networks in hardware is still far-fetched. These difficulties in real-time hardware implementation motivate us to provide the strategies to optimize the DL networks so that the DL-based AMC methods can be implemented on resource-constrained FPGA and RFSoC platforms.

1.1 Related work and contributions

To reduce the complexity and memory requirement of DL-based approaches, researchers have proposed several schemes, in which some of these schemes have used compressed DL networks that exploit efficient computation while others have used quantization of DL networks and low-precision mathematics. Iandola *et al.* in [9] replaced the larger convolutional kernels with smaller kernels that further reduce the AlexNet parameters by 50% with similar performance as the AlexNet. Liu *et al.* in [10] proposed Sparse Convolutional Neural Networks (SCNNs) to reduce the inherent redundancy. Denil *et al.* in [11] demonstrated that up to 95% of the weights of a DL model can be predicted by only having the knowledge of a few weight values for each feature. This work also showed that we do not need to store all the weights and hence only a few of the weights are important, thus reducing the memory footprint. With the goal of achieving efficient inference in hardware, many quantization approaches for Neural Networks (NN) have been investigated. In [12], an efficient method for training networks with various forward and backward functions was presented. This resulted in new uniform quantization functions for low-precision NNs [13, 14]. Under low-precision weights and activations, authors in [15] obtained state-of-the-art accuracies for the AMC problem. Several accelerator architectures for low-precision CNNs with uniform quantization arithmetic have been proposed in [16]. Alternative FPGA architectures have been developed to make use of CNNs' extremely adaptable nature, which limit weight parameters to binary or ternary representations [17, 18].

CNNs are computationally expensive in terms of inference and training. CNNs must be less resource and memory hungry in order to be implemented on FPGAs [15, 19]. We must also note that current CNN models are overly parameterized [20, 21] which opens up the opportunity of reducing the precision of input, weights, and activations without affecting the accuracy [14, 18]. Generally, during the training of DL models, single or double floating-point arithmetic is used; the reduced precision and pruning can be used during training and testing such models without a significant decrease in accuracy. This leads to new opportunities for building accelerators for the FPGA deployment of such models.

To have a less complex DL network, small kernel sizes and layers having a few output channels are being used. In addition, quantized CNN is also a popular choice. Quantization is one of the methods explored in literature for efficient NN inferences. This method has proved to give good and consistent results for both training and inference phases. Quantization is the process of reducing the precision of the numbers that are used for the representation of different parameters in the CNN model. Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) are the two methods to make DL models quantized. In PTQ, quantization is applied to a model after it has been trained on GPUs. The training happens with weights and activations represented as 32-bit or 16-bit floating-point numbers. After the training is complete with a satisfactory model accuracy, the model is then calibrated using tools such as TensorRT INT8 entropy calibrator to a lower precision (8 bit INT or lower). While PTQ provides an easy way to model quantization errors, there are many cases where this scaling cannot preserve the statistics of the model weights and leads to performance degradation [22]. In contrast to PTQ, the quantization error in QAT is considered during training of the model. The training graph is modified to simulate the lower precision behavior in the forward pass of the training process. This introduces the quantization errors as part of the training loss, which the optimizer tries to minimize during the training. Thus, QAT helps in modeling the quantization errors during training and mitigates its effects on the accuracy of the model at deployment. We can apply quantization to input, weights, and activations [19, 23, 24]. Quantization reduces the size of the model considerably at the trade-off of some accuracy. Further, note that quantizing too much might degrade the NN performance, so a proper precision bits must be chosen carefully. For implementation of quantized NN and QAT, Brevitas [25] with Pytorch library in python has been used.

O'Shea *et al.* in [4] designed and evaluated two CNN-based models i.e., VGG10 and ResNet33 for AMC and demonstrated competitive accuracy performance over 24 modulation classes. It is worth noting that these VGG10 and ResNet33 networks show very good classification accuracy, but they are huge in size, having large memory requirements and require very complex hardware for deployment. To overcome these issues, in this paper, we propose two CNN-based AMC techniques for wireless communication and make them less complex, having a smaller memory footprint, using quantized DL and low precision arithmetic with QAT [19].

This work is an extension of our submission in the ITU AI/ML in 5G challenge 2021[a]. In this challenge, the VGG10 network [4] is provided as a baseline model. The goal of the challenge was to minimize the memory requirement and the computational complexity/hardware cost while maintaining the classification accuracy above a certain threshold. The main contributions of this paper are summarized below.

- We propose two low complexity CNN models i.e., convolution layer-based Quantized Modulation Classification Network (QMCNet) and Residual-Unit-

[a]For details visit: `https://aiforgood.itu.int/about-ai-for-good/aiml-in-5g-challenge/challenge-2021/`

based neural Network (RUNet) for AMC. These CNN architectures reduce complexity as well as memory size with similar accuracy performance with respect to VGG10 network of [4].

- To further reduce the memory and resource requirements for hardware implementation, a low precision fully quantized input, weights and activations-based CNN implementation is considered.
- Further, an iterative pruning-based training method is used to maintain the classification accuracy above a certain threshold while making the proposed network sparse.

The organization of this paper is as follows. Section 2 presents the RadioML dataset which we considered in this work. The system model, consisting of the different NN architectures developed for hardware deployment, is discussed in Section 3. The simulation results along with model complexity and classification performance are presented in Section 4. Finally, Section 5 concludes the work.

2. RADIOML DATASET

The open-source RadioML 2018.01A dataset has been used to train and test our DL-based AMC schemes. This RadioML 2018.01A dataset has been generated in real time using universal software radio peripheral devices in natural environments as described in [4]. In this dataset, a total of 24 analog and digital modulation schemes have been generated and recorded over a wide range of Signal-to-Noise Ratio (SNR) (-20dB to +30dB in a gap of 2dB). Each recorded data is a time series of length 1024 samples. The recorded data frame is divided into I and Q sample pairs and stored in an 2×1024 array. There are 4096 training samples for each modulation class and SNR pair, for a total of 2.56M labeled I/Q time-series data in the dataset. The 24 modulation classes include a broad range of modulation types such as OOK, 4ASK, 8ASK, BPSK, QPSK, 8PSK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, AM-SSB-WC, AM-SSB-SC, AM-DSB-WC, AM-DSB-SC, FM, GMSK, OQPSK details of which can be found in [4]. In the next section, we present the system model of the proposed AMC schemes.

3. SYSTEM MODEL

In this section, the details of proposed CNN-based AMC system architectures, how to make them less complex with smaller memory footprint and training methodology of the models, are discussed.

3.1 Quantized Modulation Classification Network (QMCNet)

The proposed AMC models are designed efficiently according to the requirement of the hardware implementation of such models. More specifically, a less complex CNN architecture is proposed in this work that consists of six one dimensional (1D) convolution (conv1D) layers with a small kernel size and lower number of output channels, compared to the VGG10 network in [4].

Table 1 – Proposed architecture for QMCNet

Layer	**Output Dimension**
Input	2×1024
Conv	16×1024
Maxpool	16×512
Conv	24×512
BatchNormalization	24×512
Maxpool	24×256
Conv	32×256
BatchNormalization	32×256
Maxpool	32×128
Conv	48×128
BatchNormalization	48×128
Maxpool	48×64
Conv	64×64
BatchNormalization	64×64
Maxpool	64×32
Conv	96×32
BatchNormalization	96×32
Avgpool	96×1
FullyConnected/ReLu	24
FullyConnected/Softmax	24

This proposed architecture gives complexity and memory size reduction upon the unquantized VGG10 networks in [4] while having 1.2% degradation in accuracy. The overall architecture of the proposed QMCNet is displayed in Table 1. To make QMCNet less memory greedy, quantization of inputs, weights, biases and activations have been used. The 8-bit quantized QMCNet network has 1% lower accuracy than the 8-bit quantized baseline VGG10 network of [4]. On the one side, using low precision quantization saves the memory, while on the other side it reduces the accuracy. To further reduce the complexity and the memory requirement of our proposed QMCNet, after heuristic search[b] we choose inputs, weights and activations to be represented by 4, 5 and 6 bits respectively. This quantized QMCNet has only 19.14%, 30.0% and 38.60% multiply-accumulate bit operations (bit_operations), weight bits and activation bits respectively than the 8-bit quantized baseline. This reduction in the memory and computational complexity comes at a cost of reduction of 2.11% accuracy as compared to the-8 bit quantized baseline VGG10 network [4].

[b]We tried all the possible combinations of choosing the number of bits heuristically, from 8, 8, 8 bits to 4, 4, 4 bits for input, weight and activation respectively, and the best out of them for meeting minimum accuracy was considered.

3.2 Residual-Unit-based neural Network (RUNet)

In previous subsection, we have discussed a simple convolution layer-based model with six conv1D layers to lower the complexity over the VGG10 network [4]. However, the quantized (4,5,6 bit) QMCNet shows 2.11% degradation in accuracy. To increase the accuracy, we further utilize deeper networks. Note that the accuracy, in the case of a deeper network, gets saturated which causes high training errors when more layers are added to the network. However, with the recent advancements in DL techniques [26], training the deeper networks with less error is now possible. In [26], authors have shown that compared to swallow non-residual networks, residual networks can gain accuracy from the considerably increased depth and are easier to optimize. To extract the deep features from the received raw input signal, a Residual Unit (RU) and a composite convolution block (ConvB) which contain multiple one dimensional convolutional (Conv1D) layers play a significant role.

In the RU, the skip connections are used to operate the features at multiple depths and scales throughout the network. Fig. 1a represents the internal structure of the RU. It consists of two conv1D layers followed by the batch normalization layer and ReLu activation function. A skip connection is made between the input and the output of this layer arrangement to make the RU a residual network and also to prevent the network from overfitting during the training process. More details about the skip connections and the residual network can be found in [26]. The main difference between the RU and ConvB block is that in ConvB block 2 conv1D layers with different kernels, strides and output channel sizes have been used to extract the vertical features on the spatial dimension in a better fashion. Fig. 1b and Fig. 1c represent the layering structure of the ConvB block and proposed RUNet for AMC. In Fig. 1, O, O_1 and O_2 represent the number of output channels, K represents the kernel size, $Num\_classes$ denote the total modulation classes which are 24, S represents the stride of the respective layers and FC represents a fully connected or dense layer. To maintain good accuracy, after many trails we choose inputs, weights and activations to be represented by 6 bits each.

3.3 Training methodology

Quantization of the network and the use of a small kernel size and lower number of output channels in the networks make the proposed QMCNet and RU-based AMC less complex and offer small memory footprints. Still, the large number of trainable parameters is a hinder to its hardware implementation. In general, many CNN models include too many redundant parameters. Identifying and eliminating these parameters helps reduce model complexity without compromising performance. To address this issue and further make our model less complex, we use iterative pruning methodology while training the networks.

Algorithm 1: Iterative pruning-based training method

```
Input: Total number of pruning iteration $k = 50$,
       Pruning Amount per iteration $\delta = 0.2$, Total
       number of training Epochs = $E = 30$,
       Minimum Accuracy = 58%
Output: Pruned model
for Pruning Iterations=1:k do
    Best Accuracy = 0
    for Epochs =1:E do
        Train the Model and find Test Accuracy
        if Test Accuracy $\geq$ Best Accuracy then
            save model Weights
            Best Accuracy = Test Accuracy
        end
        if Test Accuracy $<$ Best Accuracy for $10$
          consecutive Epochs then
            Stop Training
        end
    end
    if Test Accuracy $<$ Minimum Accuracy then
        Stop Pruning Iteration
    end
    Prune the model weight to $\delta$
end
```

Pruning reduces the number of trainable parameters and network connections having negligible weights, and as a result the computational complexity of the provided model decreases. Algorithm 1 represents the iterative pruning methodology used for training of the networks, where minimum achievable accuracy is set as 58%. Note that this value can be easily changed based on the requirement, but within the limitations of the architecture performance. We have used the Adam's optimizer with a learning rate of 0.01 and a batch size of 1024 is used during training. We have trained the model for 50 pruning iterations and 30 epochs for each iteration. If the accuracy drops below the minimum desirable accuracy (58%) the training stops. After successive pruning and retraining, it is seen that the weights those are having near zero values are converted to zero and retraining distributes the contribution of those weights to remaining non-zero weights. Fig. 2 represents the weights distribution after successive pruning and iteration(s). Each successive pruning increases the number of zero valued weights. Table 2 represents the performance of proposed QMCNet(4,5,6) with different numbers of pruning iterations. With increasing numbers of pruning iterations, the complexity (Total $bit\_operations$) reduces significantly at a cost of minor reduction in accuracy.

4. SIMULATION RESULTS

Inference or testing is the process of using the pretrained AMC models to classify the different modulation schemes. In this section, we present the performance of the pro-

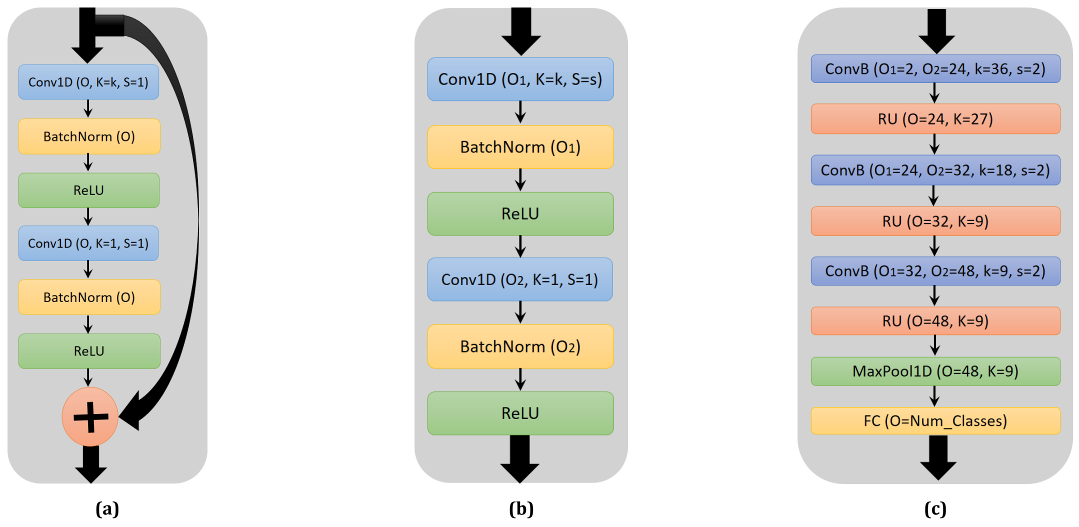

Fig. 1 – (a) Internal architecture of a Residual Unit (RU), (b) Internal architecture of a convolutional block (ConvB) and (c) Proposed RU-based AMC system model (RUNet), where O, O_1 and O_2 represent the number of output channels, K denotes the kernel size, and S represents the stride of the respective layers.

posed AMC systems. The proposed systems have been evaluated on the basis of classification accuracy over a wide range of SNRs and complexity. For the sake of clarity, in this paper, we represent the (Input, Activation, Weight) quantization bits along with the model name in brackets. Also, superscript ^{p} in (Input, Activation, Weight)^p represents that the model is pruned. In simulation, the VGG10 network in [4] is considered as the baseline.

4.1 Classification performance

The proposed system has been tested using software simulation. Fig. 3 represents the average classification accuracy of all 24 modulation formats vs SNRs. From this figure, the following observations were made:

1. The classification accuracy of the proposed schemes and the baseline is more than 50% above 0 dB SNR.

2. The classification accuracy of the RUNet(6,6,6)^p scheme is at least 1.4% higher than baseline(8,8,8) scheme above -12 dB SNR.

3. The accuracy of proposed QMCNet(8,8,8) and baseline(8,8,8) are nearly the same throughout the SNR range.

4. The proposed QMCNet(4,5,6)^p scheme provides similar accuracy than baseline(8,8,8) up to +8 dB SNR. Above +8 dB the baseline(8,8,8) accuracy is higher.

5. The accuracy of the QMCNet(4,5,6)^p and QMCNet(4,5,6) schemes is nearly the same. This indicates that pruning the networks has little impact on classification accuracy.

6. The maximum accuracy provided by RUNet (6,6,6)^p network, QMCNet(4,5,6)^p and baseline(8,8,8) are 94.46%, 90.58% and 93.07% respectively at +30 dB SNR.

The confusion matrix for the QMCNet(4,5,6)^p classifier is shown in Fig. 4 for all 24 classes at SNRs ranging from -20 dB to +30 dB. High-order Quadrature Amplitude Modulation (QAM) (64/128/256-QAM) and AM modulations are the biggest sources of errors. Large inaccuracy is expected for short-time observations with significant noise, especially with higher order QAM, due to a lack of information and similar symbol structure. The confusion matrix for the RUNet (6,6,6)^p classifier is shown in Fig. 5 for all 24 classes at SNRs ranging from -20 dB to +30 dB. This RUNet technique reduces the classification error among high order QAM schemes (e.g., 64/128/256-QAM) and attains higher accuracy than QMCNet(4,5,6)^p by 3.88% at +30dB SNR.

4.2 Model complexity

Higher computational complexity of any DL model hinders its implementation on any area and power constrained FPGA hardware. For analyzing the complexity of our proposed AMC models, the following methodology has been used[c]:

- The complexity calculation will take into account the number of Multiply-Accumulate (MAC) operations and the number of parameters for convolutional (Conv) and matrix multiplication operations, assuming a naive/direct implementation.

- The following layers are assumed to be zero-cost and will not be taken into account for the inference cost: element-wise nonlinearities, pooling, reshape/transpose/concatenation, batch normalization, multiplication/division and addition/subtraction for applying biases or quantization

[c]Details of complexity cost calculation function is shown at: `https://github.com/Xilinx/finn-base/blob/feature/itu_competition_21/src/finn/analysis/inference_cost.py`

Table 2 – Performance of QMCNet(4,5,6) at different numbers of pruning iterations

Pruning Iteration(s)	Average Accuracy	Accuracy at 30 dB	Total $bit\_operations$
0	58.7051%	90.9654%	154.63m
2	58.6667%	90.7520%	135.27m
4	58.6267%	90.6820%	111.72m
6	58.5827%	90.6439%	92.65m
8	58.5493%	90.6073%	79.23m
10	58.5057%	90.5204%	64.47m

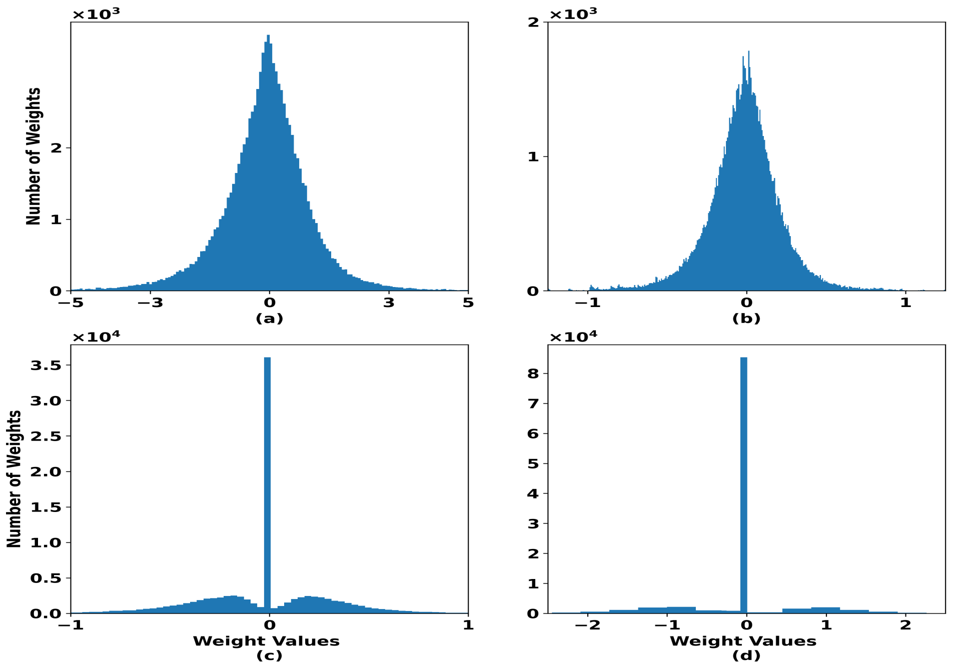

Fig. 2 – Weight distribution of QMCNet (a) Unquantized unpruned model (b) Quantized unpruned model (c) Quantized pruned model with 1 iteration (d) Quantized pruned model with 10 iterations

scaling factors. These layers are element-wise and/or otherwise have little impact on total cost of inference based on existing DNN topologies.

- The complexity cost will discount for parameter sparsity by

$$discounted\_MAC = MAC \times \frac{num\_nonzero\_Weights}{num\_total\_Weights},$$

and

$$discounted\_mem = mem \times \frac{num\_nonzero\_Weights}{num\_total\_Weights},$$

where $discounted\_MAC$ is total MAC operations in a sparse network after pruning, $num\_nonzero\_Weights$ is the number of non-zero weight values, $num\_total\_Weights$ is a total number of weight values in the network, $discounted\_mem$ is total memory requirement in the sparse network after pruning, mem is total memory requirement of the network before pruning.

Note that for each layer, activation sparsity and any other form of dynamic sparsity is not taken into account.

- The complexity cost will account for precision as follows:

$$bit\_operations = discounted\_MAC \times weight\_bits \times act\_bits,$$

and

$$bit\_mem = discounted\_mem \times weight\_bits,$$

for each layer. Here bit_ops represents the total bit operations required by the network, $weight\_bits$ represents the total bits required to store weights and act_bits represents the total bits required to store activations.

Table 3 represents the calculated complexity cost of the proposed AMC schemes with baseline. Here, "Total act_bits" represents the total number of activation bits, "Total $weight\_bits$" represents the total number of

Table 3 – Complexity cost and accuracy comparison

Model	**Precision in bits (input, weight, activation)**	**Total** *bit_operations*	**Total** *act_bits*	**Total** *weight_bits*	**Average accuracy**	**Accuracy at +30 dB SNR**
QMCNet	unquantized	9578.28m	1.71m	3.14m	59.7311%	93.5061%
QMCNet	(8, 8, 8)	487.24m	1.71m	0.629m	59.1159%	92.0732%
QMCNet	(4, 5, 6)	154.63m	1.61m	0.372m	58.7051%	90.9654%
QMCNet	$(4, 5, 6)^p$	64.47m	1.61m	0.154m	58.5057%	90.5894%
RUNet	(6, 6, 6)	74.59m	2.92m	0.044m	60.6823%	94.6443%
RUNet	$(6, 6, 6)^p$	37.45m	2.92m	0.022m	60.5775%	94.46%
Baseline[4]	(8, 8, 8)	807.69m	4.17m	1.24m	59.4657%	93.0793%
Baseline[4]	unquantized	13173.26m	4.17m	5.09m	60.6950%	94.7053%

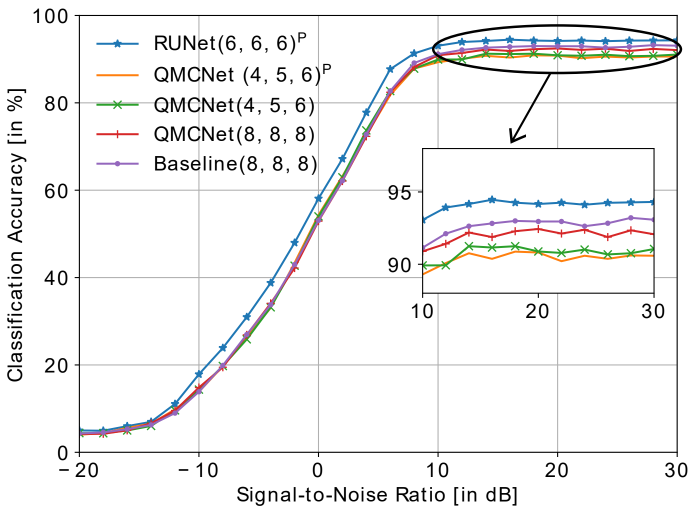

Fig. 3 – Overall accuracy of all modulation schemes averaged over SNR ranges [-20, +30] dB for CNN model

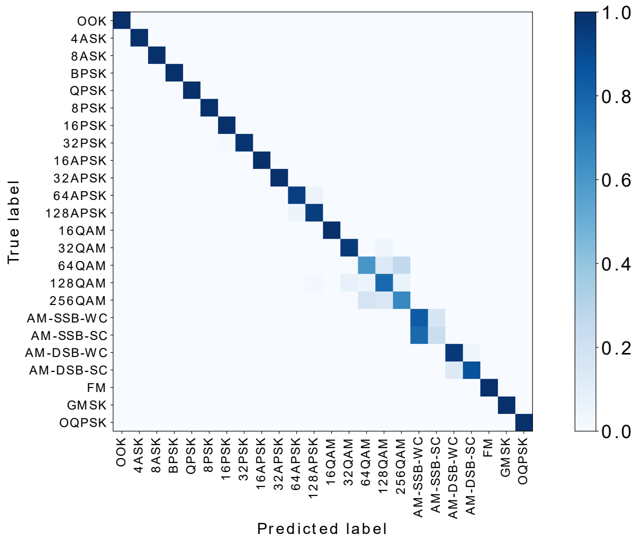

Fig. 4 – Confusion matrix for QMCNet(4,5,6)^P for all 24-modulations averaged over the SNR range -20 dB to +30 dB

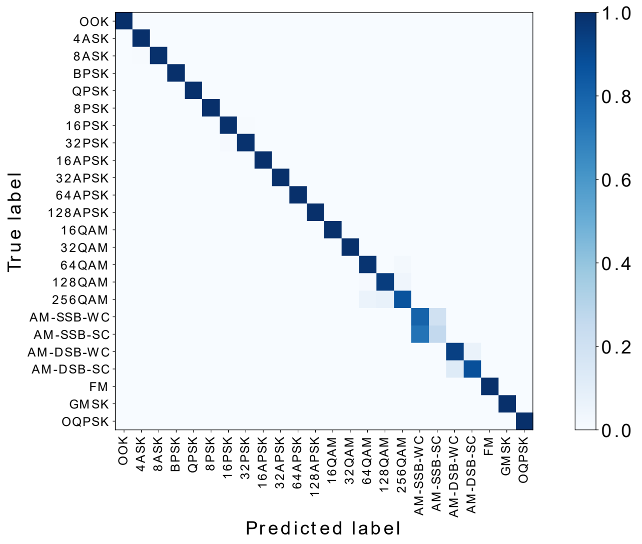

Fig. 5 – Confusion matrix for RUNet (6,6,6)^P for all 24-modulations averaged over the SNR range -20 dB to +30 dB

weight bits, and "Total *bit_operations*" represents total multiply-accumulate bit operations. This table shows that the proposed QMCNet(4,5,6)^p model has only 7.99% bit_operations and 12.42% weight bits than 8 bit quantized baseline whereas the RUNet(6,6,6)^p model has only 4.64% bit_operation and 1.78% weight bits than 8 bit quantized baseline.

5. CONCLUSION

In this paper, we demonstrate a real-time AMC convolution layer-based network (QMCNet) and a Residual-Unit-based Network (RUNet). We then propose quantization and iterative pruning methodologies to reduce the overall computational complexity of the model while keeping the average accuracy over a certain threshold. We can see that our design is 21.55 times less complex than the baseline model while having 1.6% more average accuracy. Our convolution layer-based QMCNet (Quantized and Pruned) has 92.01% less multiply-accumulate bit operations (bit_operations), 61.39% less activation bits and

87.58% less weight bit than the 8 bit quantized baseline model whereas the RUNet model (Quantized and Pruned) has 95.36% less bit_operations, 29.97% less activation bits and 98.22% less weight bits than the 8 bit quantized baseline model.

ACKNOWLEDGEMENT

Authors acknowledge the financial assistance from the UGC, New Delhi, in the form of the NET-Senior Research Fellowship (UGC NET-SRF), which supports the work.

REFERENCES

[1] Thien Huynh-The, Quoc-Viet Pham, Toan-Van Nguyen, Thanh Thi Nguyen, Rukhsana Ruby, Ming Zeng, and Dong-Seong Kim. "Automatic Modulation Classification: A Deep Architecture Survey". In: *IEEE Access* 9 (2021), pp. 142950–142971.

[2] Mohamed A. Abdel-Moneim, Walid El-Shafai, Nariman Abdel-Salam, El-Sayed M. El-Rabaie, and Fathi E. Abd El-Samie. "A survey of traditional and advanced automatic modulation classification techniques, challenges, and some novel trends". In: *International Journal of Communication Systems* 34.10 (2021), e4762.

[3] A. Swami and B.M. Sadler. "Hierarchical digital modulation classification using cumulants". In: *IEEE Transactions on Communications* 48.3 (2000), pp. 416–429.

[4] Timothy James O'Shea, Tamoghna Roy, and T. Charles Clancy. "Over-the-Air Deep Learning Based Radio Signal Classification". In: *IEEE Journal of Selected Topics in Signal Processing* 12.1 (2018), pp. 168–179.

[5] Seung-Hwan Kim, Jae-Woo Kim, Williams-Paul Nwadiugwu, and Dong-Seong Kim. "Deep Learning-Based Robust Automatic Modulation Classification for Cognitive Radio Networks". In: *IEEE Access* 9 (2021), pp. 92386–92393.

[6] Esra Mahsereci Karabulut, Selma Ayşe Ozel, and Turgay İbrikçi. "A comparative study on the effect of feature selection on classiication accuracy". In: *Procedia Technology* 1 (2012). First World Conference on Innovation and Computer Sciences (INSODE 2011), pp. 323–327. ISSN:2212-0173.

[7] Chen Zhang, Di Wu, Jiayu Sun, Guangyu Sun, Guojie Luo, and Jason Cong. "Energy-Efficient CNN Implementation on a Deeply Pipelined FPGA Cluster". In: *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. 2016, pp. 326–331. ISBN:9781450341851.

[8] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, Yu Wang, and Huazhong Yang. "Going Deeper with Embedded FPGA Platform for Convolutional Neural Network". In: *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. New York, NY, USA, 2016, pp. 26–35. ISBN:9781450338561.

[9] Iandola Forrest N., Song Han, Khalid Moskewicz Matthew W.and Ashraf, William J. Dally, and Kurt Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size". In: *arXiv:1602.07360* cs.CV (2016). DOI: 10.48550/arXiv.1602.07360.

[10] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Penksy. "Sparse Convolutional Neural Networks". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 806–814. DOI:10.1109/CVPR.2015.7298681.

[11] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. "Predicting Parameters in Deep Learning". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. Lake Tahoe, Nevada, 2013, pp. 2148–2156.

[12] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation". In: *arXiv:1308.3432v1* cs.LG (2013). DOI: 10.48550/arXiv.1308.3432.

[13] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 3123–3131.

[14] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients". In: *arXiv:1606.06160v3* cs.NE (2016). DOI:10.48550/arXiv.1606.06160.

[15] Stephen Tridgell, David Boland, Philip H.W. Leong, Ryan Kastner, Alireza Khodamoradi, and Siddhartha. "Real-time Automatic Modulation Classiication using RFSoC". In: *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 2020, pp. 82–89. DOI: 10.1109/IPDPSW50202.2020.00021.

[16] Chen Wu, Mingyu Wang, Xinyuan Chu, Kun Wang, and Lei He. "Low-Precision Floating-Point Arithmetic for High-Performance FPGA-Based CNN Acceleration". In: *ACM Trans. Reconfigurable Technol. Syst.* 15.1 (Nov. 2021). ISSN:1936-7406. DOI:10.1145/3474597.

[17] Duncan J.M Moss, Srivatsan Krishnan, Eriko Nurvitadhi, Piotr Ratuszniak, Chris Johnson, Jaewoong Sim, Asit Mishra, Debbie Marr, Suchit Subhaschandra, and Philip H.W. Leong. "A Customizable Matrix Multiplication Framework for the Intel HARPv2 Xeon+FPGA Platform: A Deep Learning Case Study". In: FPGA '18. Monterey, CALIFORNIA, USA: Association for Computing Machinery, 2018, pp. 107–116. ISBN: 9781450356145. DOI: `10.1145/3174243.3174258`.

[18] Ganesh Venkatesh, Eriko Nurvitadhi, and Debbie Marr. "Accelerating Deep Convolutional Networks using low-precision and sparsity". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 2861–2865. DOI: `10.1109/ICASSP.2017.7952679`.

[19] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. "A Survey of Quantization Methods for Eficient Neural Network Inference". In: *arXiv:2103.13630v3* cs.CV (2021). DOI:`10.48550/arXiv.2103.13630`.

[20] Karthik Abinav Sankararaman, Soham De, Zheng Xu, W. Ronny Huang, and Tom Goldstein. "The Impact of Neural Network Overparameterization on Gradient Confusion and Stochastic Gradient Descent". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 8469–8479.

[21] Nidhi Gowdra, Roopak Sinha, Stephen MacDonell, and Wei Qi Yan. "Mitigating severe overparameterization in deep convolutional neural networks through forced feature abstraction and compression with an entropy-based heuristic". In: *Pattern Recognition* 119 (2021). ISSN:0031-3203. DOI:`https://doi.org/10.1016/j.patcog.2021.108057`.

[22] Varun Praveen. *Improving INT8 Accuracy Using Quantization Aware Training and the NVIDIA TAO Toolkit*. URL:`https://developer.nvidia.com/blog/improving-int8-accuracy-using-quantization-aware-training-and-tao-toolkit/` (visited on 07/07/2022).

[23] Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference". In: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. FPGA '17. Monterey, California, USA, 2017, pp. 65–74. ISBN:9781450343541. DOI:`10.1145/3020078.3021744`.

[24] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. "Training Deep Neural Networks with 8-Bit Floating Point Numbers". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada, 2018, pp. 7686–7695.

[25] Alessandro, Giuseppe Franco, nickfraser, Javier Duarte, SpontaneousDuck, Yaman Umuroglu, derpda, and vfdev et.al. *Xilinx/brevitas: Release version 0.7.0*. Oct. 2021. DOI:`10.5281/zenodo.5615725`.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI:`10.1109/CVPR.2016.90`.

AUTHORS

Satish Kumar received B.E. degree in electronics and communication engineering from the Rajiv Gandhi Proudyogiki Viswavidyalaya, Bhopal, India in 2011; an M.E. degree in wireless communication from Birla Institute of Technology, Mesra, Ranchi, in 2015, and is currently pursuing Ph.D. degree in electronics and communication engineering from Dr. SPM IIIT- Naya Raipur, India.

Aakash Agarwal is currently pursing his Bachelor of Technology in electronics and communication from the International Institute of Information Technology, Naya Raipur, Chattishgarh. His areas of interest lie in machine learning in communication technologies and IoT in healthcare.

Neeraj Varshney received a B.Tech. degree in electronics and communication engineering from Uttar Pradesh Technical University, Lucknow, India, in 2008, an M.Tech. degree in electronics and communication engineering from the Jaypee Institute of Information Technology, Noida, India, in 2011, and a Ph.D. degree in electrical engineering from Indian Institute of Technology Kanpur, India, in 2018. Since Nov' 18 he is an international research associate with the wireless networks division at National Institute of Standards and Technology, Maryland, USA. His research interests are in signal processing, communications and networks which include mmWave and THz wireless communication, MIMO technology, and molecular communication. From Oct ’11 to Aug ’12, he was a project research fellow at Jaypee Institute of Information Technology, Noida, India. From May ’18 to Sep ’18, he was a visiting researcher with the department of electrical engineering and computer science at Syracuse University, New York, USA. He has served as a TPC member for 2019/2020 IEEE Globecom conferences in the track on molecular, biological, and multiscale communications.

Rajarshi Mahapatra received a Ph.D. degree in electronics and electrical communication engineering from the Indian Institute of Technology Kharagpur, Kharagpur, India, and a postdoctoral degree from the Commissariat à l’Energie Atomique et aux Énergies, Grenoble, France. Presently, he is an associate professor with the Department of Electronics and Communication Engineering, Dr. SPM IIIT- Naya Raipur, India.