RFNET: FAST AND EFFICIENT NEURAL NETWORK FOR MODULATION CLASSIFICATION OF RADIO FREQUENCY SIGNALS

Mohammad Chegini[1], Pouya Shiri[1], Amirali Baniasadi[1]
[1]University of Victoria, Canada

NOTE: Corresponding author: Pouya Shiri, pouyashiri@uvic.ca

Abstract – *Automatic Modulation Classification (AMC) is a well-known problem in the Radio Frequency (RF) domain. Solving this problem requires determining the modulation of an RF signal. Once the modulation is determined, the signal could be demodulated making it possible to analyse the signal for various purposes. Deep Neural Networks (DNNs) have recently proven to be successful in solving this problem efficiently. However, since deep networks consist of several layers resulting in a high number of trainable parameters, the hardware implementations of these solutions are resource-demanding. In order to address this challenge, we propose an efficient deep neural network referred to as RFNet to tackle the AMC problem efficiently. This network introduces the novel Multiscale Convolutional (MSC) layer to extract robust features in different resolutions. In addition, the network takes advantage of several Separable Convolution Blocks (SCB). These blocks employ pointwise and depth-wise convolutions to reduce network complexity. We further introduce RFNet+ and RFNet++ as extensions of RFNet with fewer number of parameters. These variants include fewer floating-point operations and hence a lower hardware implementation cost. Experimental results using the challenging RadioML 2018 dataset show that RFNet-32++ achieves an average classification accuracy of 56.09% over all Signal-to-Noise Ratios (SNRs) and an accuracy of 92.21% in +20dB SNR using only 3.1K parameters. The small number of parameters makes the RFNet family a promising solution for future AMC systems.*

Keywords – Automatic modulation classification, deep learning, depth-wise convolution, pointwise convolution, pruning, quantization, quantization aware training, resource efficient deep learning

1. INTRODUCTION

Wireless communication is a very important part of our daily lives. In order to make wireless systems efficient, the data transmission rate needs to be under control. One of the methods to control data rates in wireless radio environments is to apply different modulation methods to the Radio Frequency (RF) signal. Prior to the demodulation of signal at the receiver's end, Automatic Modulation Classification (AMC) is used to identify the modulation method in various applications e.g., electronic countermeasures [1, 2] and spectrum monitoring methods [3]. There are different methods to implement AMC. One important category of such methods includes conventional solutions, which are based on maximum-likelihood and hand-crafted feature extraction [4]. An alternative and efficient method is to employ Deep Learning (DL) [5, 6].

DL takes advantage of deep structures to create a representation of the input data. This representation is used for building computational models. During recent years, DL models have become mature enough to effectively make meaningful contributions to the wireless communication domain. Recent research shows that DL not only enables cognitive radio applications, but also serves as a useful method for providing algorithms for different applications of spectrum sensing including signal classification, estimation, and detection [7].

DL solutions are based on Deep Neural Networks (DNNs) which extract rich features from the input signal using several consecutive layers. These methods, however, come with high computational complexity. This is due to the high number of floating point operations and the activation functions of the neural layers in the network. This in turn increases the associated power and area cost [8].

In this work, we propose RFNet, a specialized DNN for the Radio Frequency (RF) domain. This network builds on a novel architecture to extract robust features effectively. We propose a new layer referred to as the Multiscale Convolutional (MSC) layer, to extract the low-level features from the input signal in different scales. In addition, we use Separable Convolution Blocks (SCBs) which contain pointwise and depth-wise convolutions. RFNet comes in different configurations, depending on the depth of the network (e.g. RFNet-32, RFNet-48, etc). Generally, deeper variants have better accuracy and a higher implementation cost. Making a network deeper adds to the complexity of the network which in turn leads to obtaining a more robust representation of the input data. However as the network gets deeper, more parameters are required which requires more memory, area and computation, thus, increasing the implementation costs. We further use different network compression techniques to carefully reduce the complexity of RFNet. Accordingly, we introduce RFNet+ and RFNet++, each of which com-

presses RFNet to a certain degree. In summary, we make the following contributions:

- We design RFNet as an efficient network with a classification accuracy of 56.09% (on average for all signal-to-noise ratios) using the RadioML 2018 dataset. We carefully design the MSC layer to extract features effectively. Moreover we build a deep network equipped with high generalization power by employing SCB blocks. The designed network creates a robust representation of the input signal, which in turn leads to a high accuracy.
- We reduce the complexity of RFNet by taking advantage of quantization to reduce the number of bits used for the trainable parameters. To this end, we reduce the number of bits from 32 to only 6 bits. We show that 6 bits is sufficient to obtain a high classification accuracy.
- We propose a resource-efficient network by compressing RFNet using unstructured pruning. Using this method, we remove unnecessary neurons from RFNet to utilize fewer resources. As Table 1 shows, the smallest variant is RFNet-32++ and contains only 3.1K trainable parameters.
- We provide a multi-target solution by taking into account the trade-off between accuracy and the number of parameters in our network design. This is achieved by investigating the depth parameter for the network, and using different pruning strategies. Hence, we offer a wide range of options to the end user.

The rest of the paper is organized as follows. In Section 2 we present related work. Section 3 presents the background information for modulation classification and deep learning in this domain. We explain our proposed network, RFNet and its variations, in detail in Section 4. Afterwards, we present the experimental results and report the performance metrics in Section 5. Finally, we conclude the paper in Section 6.

2. RELATED WORK

There are several pieces of work that use DNN-based solutions to tackle the AMC problem, which either propose a new neural network, or propose methods to extract additional features to feed to an existing network.

O'Shea et al. adapt the VGG architecture [9] to a one dimensional CNN and represent a simple deep CNN which can train and become deployed for different signal classification tasks. This network obtains 95.6% test accuracy on the dataset developed by the authors (which was later presented as RadioML 2018). This work includes a detailed analysis of the effects of offset in the carrier frequency, the symbol rate, and multipath fading in simulations.

Huynh-The et al. [10] develop an efficient CNN for solving the AMC problem. This network consists of different asymmetric parallel convolutional layers and includes two important mechanisms. First, it uses skip-connections to avoid the problem of vanishing gradients. Second, it concatenates features with concentration on the depth level. This method utilizes the features optimally. This network obtains a classification accuracy of 85.10% at Signal-to-Noise Ratio (SNR) of 10dB for the RadioML 2018.01A dataset and contains 151K parameters.

Tunze et al. [11] propose a low-complexity CNN by leveraging the depthwise and regular grouped sparse layers. This network uses three main modules: a module to provide a trade-off between speed and accuracy, one for deep feature extraction and processing, and another one for generic feature extraction. These blocks use various special layers, including the depthwise convolutional and cascaded regular grouped convolutional layers. They also utilize inter-layer skip-connections to reduce the computational complexity. This network obtains an accuracy of 94.39% at an SNR of 20dB for the RadioML 2018.01A dataset.

Wang et al. [12] use compressive sensing and introduce a scaling factor for each neuron in the network with the goal of reducing the model size and speeding up the inference. This scaling factor induces sparsity in the CNN which helps with pruning redundant neurons. The datasets used in this work are created by the authors.

Njoku et al. [13] propose CGDNet as a hybrid and cost efficient network. CDGNet uses a shallow CNN, a Gated Recurrent Unit (GRU) and a deep classifier. The CNN module uses small filter sizes and includes Gaussian dropout layers and skip-connections. This network obtains an accuracy of 93.5% for an SNR range of 6dB to 18dB for the RadioML 2016.01A dataset and contains 124K parameters.

Zheng et al. [14] propose a novel two-level augmentation method based on spectrum interference to improve the modulation classification. They expand the samples by adding different amounts of interference to the frequency spectrum of the radio signals. This helps make a robust representation of the input signal, as the most important difference between various modulation methods is the variation of frequency over time. The authors use bidirectional noise masks to add interference to the frequency spectrum of the signal derived using short-time Fourier transform. This method obtains 58.76% accuracy on average for all SNRs for the RadioML 2016A dataset.

Wang et al. [15] propose a Hierarchical Multi-feature Fusion (HMF) method which is based on a multidimensional Long Short-Term Memory (LSTM) network. The input features of the LSTM network are prepared by a multidimensional CNN module which compensates the features between those extracted by the one-dimensional and two-dimensional convolutional filters. The LSTM network ex-

tracts temporal features. It also employs a Softmax classifier to provide the final classification results. This work was tested on the RadioML 2016A and obtained an average accuracy of 90.88% over SNRs from 4 to 18dB.

Tu et al. [16] propose a complex-valued network for the AMC problem. They design the key building blocks of a convolutional network including the convolutional layers, batch normalization and weight initialization. This network is twice more expensive than its real counterpart. This is because in contrast to the real networks, this network considers the correlation between the real and imaginary parts of the signal. However, this consideration results in obtaining a more robust network. This network obtains 85% accuracy for SNR of 18dB for the RadioML 2016 dataset.

Note that the different pieces of work in the AMC domain do not usually report the same metrics and provide the accuracy over a specific range of SNRs. Thus providing a fair comparison is rather difficult. In contrast to the studies mentioned above, which mostly focus on the accuracy, we primarily focus on network size and complexity. In addition, we provide a family of networks providing the user with options to choose from based on the requirements of their application. For example, as we show in the results section, RFNet-32++ is a very tiny network containing only 3.1K parameters, achieving slightly less than 90% for an SNR range of 10dB to 30dB. RF-Net128, another variant of our proposed family of networks, obtains 95% accuracy over the same range, but includes 137.3K parameters.

3. BACKGROUND

In this section, we present the background information related to this work. First, we review our network compression techniques. Afterwards, we explain our evaluation metrics.

3.1 Network compression

Recently, network compression has been of utter significance. Deploying networks with millions or billions of parameters on the edge devices is practically impossible [17]. Large models require high volumes of memory and include many Floating-point Operations (FLOP)s, therefore, high energy is required to train and test these models. To leverage DL benefits on the edge devices, ML practitioners have aimed to make models more efficient. Several model compression techniques such as pruning, quantization, factorization [18] and knowledge distillation [19] have been proposed in the literature. In this work, we use the first three methods to compress RFNet.

3.1.1 Pruning

It has been observed that after successfully training a neural network, a significant number of the neurons will not fire, effectively acting as dead neurons. Also, some weights make very little contribution to the final model inference. Therefore, it would only be logical to prune redundant weights (neurons). The pruning process might lead to losing accuracy if implemented too aggressively. Pruning is generally divided into two categories: structured pruning [20] and unstructured pruning [21, 22]. While the former prunes the whole channels/filters, the latter only prunes the selected few neurons.

3.1.2 Quantization

Quantization has proven to be a very promising technique for compressing neural networks [23]. Often, 32 bits is more than enough to contain the information of the weights of a neural network. Studies show that quantizing a neural network, can result in far fewer bits without compromising accuracy. To make the neural network even more efficient, in addition to weights of the neural network, the activation maps can also be quantized [24, 25].

Quantization is generally categorized into two groups: Post-training-quantization [26] and Quantization Aware Training (QAT) [25]. In the former, the network is trained using full precision. Then and once the network is fully trained, the weights are quantized. In the latter, the neural network is quantized during the training process. Therefore, QAT directly takes into account network accuracy and loss. QAT methods generally result in a higher accuracy. [23].

3.1.3 Factorization

Conventionally, CNNs combine and filter different segments of the input and form the outputs. Usually in order to create a more robust representation of the input, the number of channels is increased. However, this results in an increase in the number of trainable weights. Furthermore, network accuracy does not scale linearly with the computational overhead. Therefore, Sifre et al [27, 28] suggested performing a combination and filtering in two separate stages. Accordingly, depthwise separable convolutions factorize a standard convolution into a depthwise convolution layer (filtering stage) followed by a 1×1 convolution [29] layer (also referred to as pointwise convolution) (combination stage).

Depth-wise separable convolutions have been used in a multitude of efficient networks such as MobileNetV1 [30], MobileNetV2 [31], Inception [32] and Xception [33]. In particular, and for MobileNet, depth-wise convolutions is applied as a separate filter to each input channel. Accordingly, a 1×1 convolution (pointwise) is used to combine

the outputs of depth-wise convolution. It has been shown that this factorization approach drastically reduces the model size and the amount of computations required for inference in a CNN [30].

3.2 Evaluation metrics

3.2.1 Number of parameters

The number of parameters is the total sum of all of the trainable weights and biases in a neural network. This number can serve as an indicator of the size of the model. Normally, models with more parameters achieve higher accuracy. However, complex models can heavily attenuate the hardware efficiency. In addition, practical hardware implementation of such models is a very difficult task. Therefore, it is essential for the network to be light enough to enhance the practicality of the model from the hardware perspective. This is particularly critical in edge devices with a small memory footprint.

3.2.2 Inference cost

In addition to the number of parameters, there are two important metrics when it comes to hardware implementation: computation cost and memory cost. The activation, bit_i, and weights, bit_w, have a specified number of bits. These two metrics are evaluated based on the following equations:

$$bit\_ops = \sum^{Layers} (\sum^{MACs} bit_w \times bit_i) \quad (1)$$

$$bit\_mem = \sum^{Layers} (\sum^{Weights} bit_w) \quad (2)$$

To report the cost of the inference, one can combine the amount of computations (the computation cost) and the amount of memory required to store the parameters for performing the inference (the memory cost) into a single metric referred to as the inference cost. This metric was introduced in a recent modulation classification challenge issued by Xilinx to propose a fast and small network [1]. The following shows the equation for the inference cost.

$$0.5 \times \frac{bit\_ops}{baseline\_bit\_ops} + 0.5 \times \frac{bit\_mem}{baseline\_bit\_mem} \quad (3)$$

Hereinafter, we refer to this score as the normalized **inference cost**.

4. RFNET

In this section we present RFNet and its building blocks. Afterwards, we explain the employed network compression methods.

[1] https://aiforgood.itu.int/event/lightning-fast-modulation-classification-with-hardware-efficient-neural-networks/

The MSC block is shown in Fig. 1. This block consists of three scales. Each scale starts with a convolutional layer employing a relatively large kernel size. Afterwards, there is a Batch Normalization (BN) layer, the ReLU activation, and another convolutional layer followed by another BN. The three scales differ in kernel size and the padding applied in the first layer. The proposed block serves as a robust low-level feature extractor.

Fig. 2 shows the architecture of the SCB block. This block is similar to each scale of the MSC layer. The difference is the addition of a max-pooling layer. Factorizing a convolutional layer into depth-wise and pointwise equivalents, results in significant parameter reduction, which in turn helps make the network deeper and yet more economic.

The architechture of RFNet is shown in Fig. 3. As stated before, the network starts with MSC to extract low-level features. Afterwards, we employ five SCB layers to gradually create a deep representation of the input signal. Finally, there are two Fully-Connected (FC) layers to learn the non-linear combination of the high-level features extracted by the network.

The BN layer is used in RFNet several times. This layer corrects the activations by zero-mean and unit standard deviation, and hence enables larger steps in the gradient. This in turn results in faster convergence of the network and bypasses sharp local minima.

4.1 RFNet variations

RFNet comes in different variations. Such variations include RFNet32, RFNet48 and RFNet128, wherein the number following the name is the number of filters used in the MSC and SCB layers. This family of networks provides the end user with a wide variety of choices. Generally a higher number of filters per layer results in better accuracy while requiring a higher number of parameters.

4.2 Compressing RFNet

We also aim to further improve RFNet. To this end, we use different model compression techniques to make the network even more energy efficient. Our study shows that unstructured pruning and QAT are the most promising choices in enhancing RFNet. To this end, we propose two alternatives: RFNet+ (slightly pruned) and RFNet++ (extremely pruned). Essentially, we take RFNet and apply unstructured pruning to obtain RFNet+. Afterwards, we take RFNet+ as the base network, and apply further pruning to obtain RFNet++. Note that, theoretically, a higher level of pruning can result in losing more accuracy. However, we limit prunning to a certain level protecting accuracy. Accordingly, we set a lower threshold for accuracy and stop pruning if accuracy falls below this threshold.

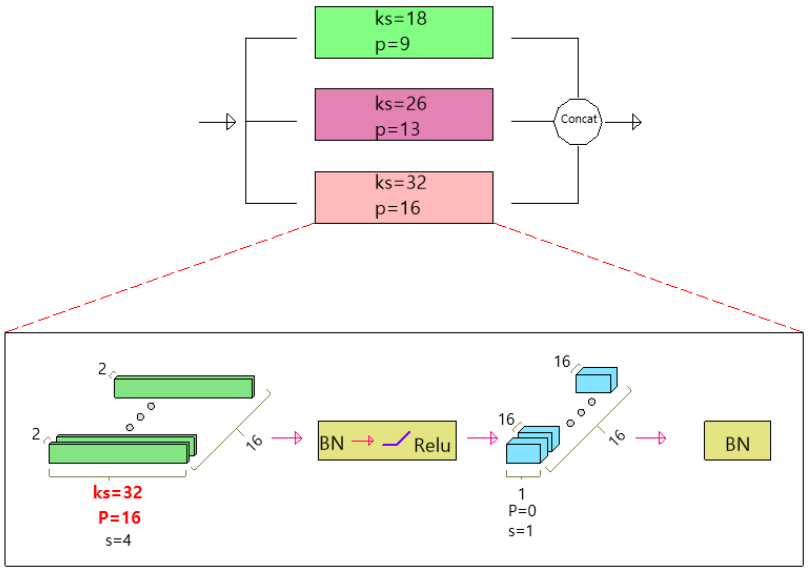


Fig. 1 – MSC layer. This layer consists of pointwise and depth-wise convolutions implemented in three scales, serving as a robust low-level feature extractor. ks and p stand for kernel size and padding respectively.

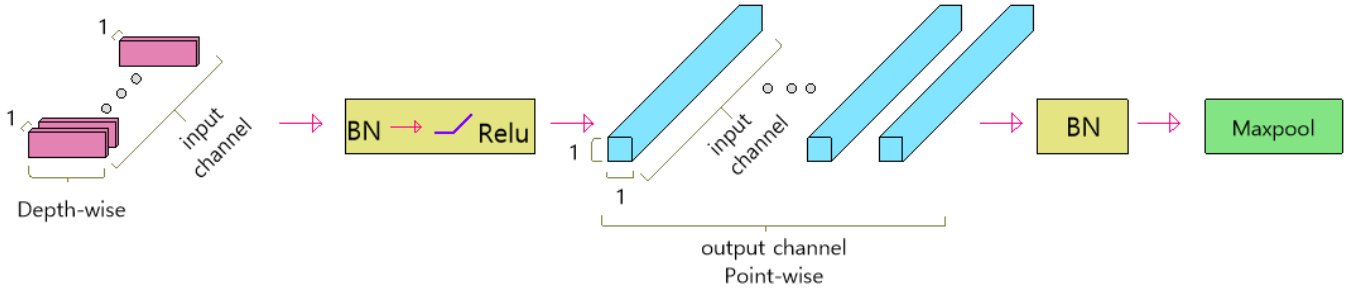


Fig. 2 – The architecture of SCB block. This block contains depth-wise and pointwise convolutions in addition to batch normalization and ends with a max-pooling layer.

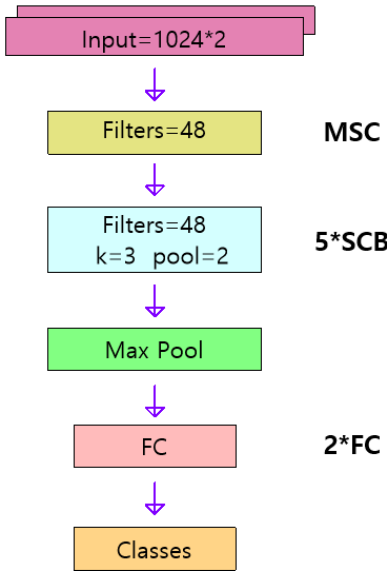


Fig. 3 – RFNet48 architecture. The network consists of the MSC layer, 5 SCB blocks, and 2 FC layers.

4.3 Pseudo-codes for unstructured pruning

The pseudo-code for unstructured pruning can be found in Algorithm 1. This algorithm requires three hyper-parameters: α, β and the vector X. α indicates the number of pruning iterations. This hyper-parameter reports the number times the model is pruned and then retrained. β is the number of epochs that the model is trained for in each pruning iteration. This parameter should be chosen carefully since a high number not only requires more training, but also promotes the risk of overfitting. On the contrary, if a low number is acquired for β, the lost accuracy might not be recovered. Finally, X shows how much of the weights are pruned at each pruning iteration.

Algorithm 1 Unstructured pruning steps

1: Initialize x vector. x_i is the percentage of weights which will be pruned in the i^{th} iteration
2: Initialize α and β
3: **for** i in α **do**
4: Prune x_i percent of the model's weights
5: **for** $epoch$ in range β **do**
6: Retrain the model to recover the lost accuracy
7: Save model's weights resulting in highest accuracy

5. EXPERIMENTS AND RESULTS

In this section, we present experimental results.

5.1 RadioML 2018 dataset

We used the RadioML 2018 dataset [9] to evaluate RFNet for modulation classification. This dataset consists of 24 modulation categories [2], and includes 2.5M data points at 26 different Signal-to-Noise Ratios (SNRs) per modulation (-20dB to +30dB in steps of 2dB). Each data point consists of 2x1024 samples: 1024 in-phase and 1024 quadrature samples.

5.2 Experiment setup

We used RTX A6000 GPUs of a LambdaLab Cloud GPU service with 48 GB of VRAM to perform the experiments. We used a research library developed in a PyTorch framework for Quantization-Aware Training (QAT) which is referred to as Brevitas [34].

5.3 Dynamic hyper-parameter: SNR

As stated, RadioML 2018.01A consists of signals from -20dB SNR to +30dB SNR. Generally, in the low-SNR regime (from -20 dB to -10 dB) classifying the modulation of the signal is practically impossible due to the heavy distortion created by the noise. In the high-SNR regime, however, the noise power is quite low with respect to the signal, and hence distinguishing signals from one another is easier.

In this work, we train our model using the data with SNRs higher than -10dB. This is very useful for the training process, since the signals with an SNR below -10dB, are heavily distorted by the noise. Such samples will provide the model with false gradients. This in turn results in lower average accuracy over all SNRs.

5.4 Precision bits

In order to achieve best quantization results, we experimented with different values. After running several experiments we concluded that using 6 bits works best. Our study shows that a network using 6 bits while being five times lighter than a full precision network offers competitive accuracy.

In addition, as we use quantization alongside other optimization techniques, our choice of the number of bits can impact the effectiveness of other optimizations. For example, while reducing the number of bits will make the network more compact, it stops the network from being pruned as extensively. As we rely on co-optimization in order to get the best results, we avoid further quantization.

[2]The modulations available in RadioML 2018 include OOK, ASK4, ASK8, BPSK, QPSK, PSK8, PSK16, PSK32, APSK16, APSK32, APSK64, APSK128, QAM16, QAM32, QAM64, QAM128, QAM256, AMSSBWC, AMSSBSC, AMDSBWC, AMDSBSC, FM, GMSK and OQPS.

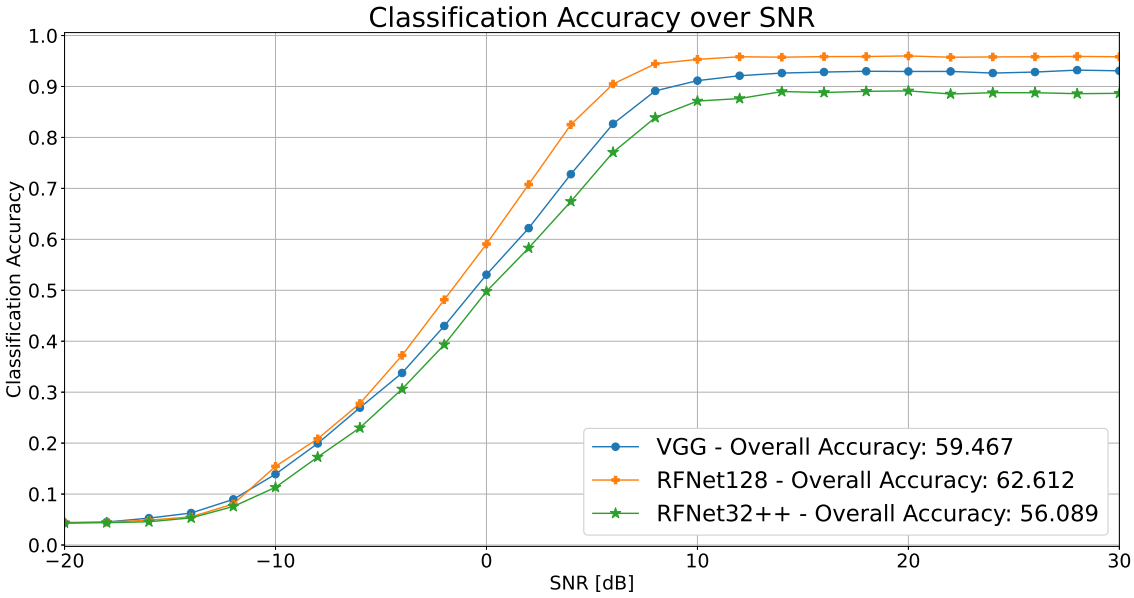


Fig. 4 – SNR can be seen on the x-axis while the y-axis reports accuracy. In SNRs less than -10dB, accuracy is not very different to that of a random guess since the noise power is quite significant. As SNR increases, so does the model accuracy (almost linearly) until +10dB. In SNRs beyond +10dB, accuracy stays intact.

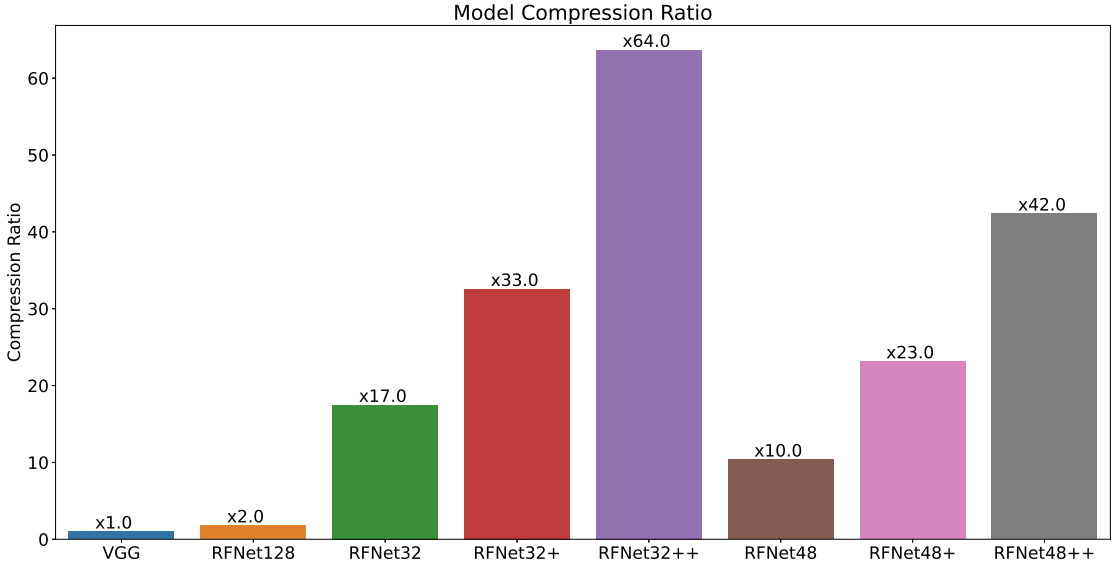


Fig. 5 – The compression ratio of different networks. The compression ratio can be derived by dividing baseline inference score by the target's inference score. The higher the compression ratio, the better the model is.

5.5 Pruning experiments

When pruning the network, we took two approaches: gradual pruning and greedy pruning. In gradual pruning, a small portion of the weights are pruned at each iteration. After each iteration, the network is trained for some epochs and then the same process is repeated. This process is followed so long further pruning does not result in significant accuracy loss.

In greedy pruning, a significant portion of the weights are pruned at the first iteration. Hence, after retraining, a small portion of the weights are removed. The process repeats until further pruning results in significant accuracy loss.

In gradual pruning, 10-20% of the weights are pruned at each iteration. In greedy pruning, we start with pruning 60-70% of weights in the first iteration. Then we prune 5-10% of the weights during the following iterations.

5.6 RFNet results

We compare our work against VGG-10 network [9]. As Table 1 shows, this network obtains 59.46% average accuracy for the RadioML 2018 dataset. Fig. 6 compares the computation and memory cost of the different networks in RF-Net family with those of the baseline VGG-10 network. As the plot shows, the amount of improvement in terms of the memory usage and the amount of computations are very close. Next, we report inference cost to compare our network to the baseline network. Table 1 reports the inference score for our proposed network along with average accuracy over all SNRs. As the table shows, we obtain a significantly lower inference cost compared to the baseline VGG network, while maintaining the accuracy. Different configurations result in small changes in accuracy while further lowering the inference cost.

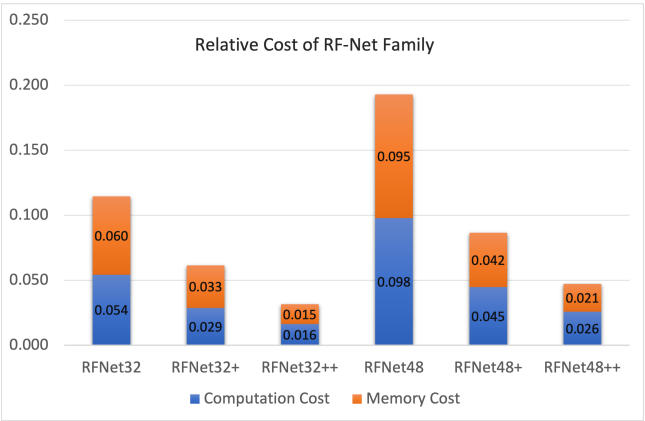


Fig. 6 – Comparing the network cost of RF-Net family with the baseline VGG network.

Table 1 also compares the number of parameters in RFNet and its different configurations to that of the baseline VGG network. RFNet32 and RFNet48 reduce the number of parameters by 91.6% and 86.7% compared to the baseline. The slightly pruned configurations i.e. RFNet32+ and RFNet48+ further reduce the number of parameters by 48.8% and 59.0% compared to base networks respectively (RFNet32 and RFNet48). Finally, extremely pruning variants i.e. RFNet32++ and RFNet48++ highly compress the base networks and reduce the number of parameters compared to the slightly pruned networks (RFNet32+ and RFNet48+) by 53.6% and 48.8%.

With 98% fewer parameters compared to the base VGG10 network, RFNet32++ is the smallest network we provide. We can conclude that unstructured pruning is very promising. For instance, RFNet32++ prunes more that 10K parameters used by the baseline network. Also when comparing RFNet48 to RFNet48+, nearly 12.5K parameters used in the base network (RFNet48) are redundant.

To better compare the inference cost, it is best to visualize the compression ratio for the evaluated networks.

Fig. 5 shows that although RFNet48 has the same accuracy as the baseline, it is 10.4×more compressed. Furthermore, it can also be seen that although RFNet32 has a deficit of 1% in accuracy, it is 17.47×more compressed than VGG10. Finally, RFNet128 has a significantly higher accuracy compared to VGG10 while being 1.71×more compressed.

Table 1 – Comparing accuracy, inference cost and the number of parameters for different configurations of RFNet with the baseline VGG network.

Network	Inference cost	Accuracy	Nummer of parameters
VGG10	1.000000	59.466%	159.1K
RFNet32	0.057224	58.497%	13.3K
RFNet32+	0.030711	58.312%	6.8K
RFNet32++	**0.015705**	56.089%	**3.1K**
RFNet48	0.096463	59.606%	21.2K
RFNet48+	0.043207	60.073%	8.7K
RFNet48++	0.023599	57.046%	4.5K
RFNet128	0.584253	**62.612%**	137.3K

5.7 Accuracy over SNRs

In order to provide better insight we also report the network's accuracy over different SNRs. To this end, we report for VGG10, RFNet128 and RFNet32++.

Fig. 4 shows how RFNet128 outperforms the rest. In the high-SNR regime, RFNet128 outperforms the baseline VGG10 by 3% while being a 2×compressed network. Furthermore, the highly compressed RFNet32++ underperforms VGG10 by as much as 3% at a cost of being 64×lighter. In order to better visualize the performance of our classifier, the confusion matrix for RFNet32++ is shown in Fig. 7.

5.8 Accuracy vs inference cost

We visualize accuracy vs inference cost of different models to provide better analysis (Fig. 8). As we show both RFNet32 and RFNet48 families follow a similar trend. RFNet48+ has a slightly better accuracy compared to RFNet48 while having less parameters and being 2.24×lighter. Furthermore, RFNet32+'s overall accuracy

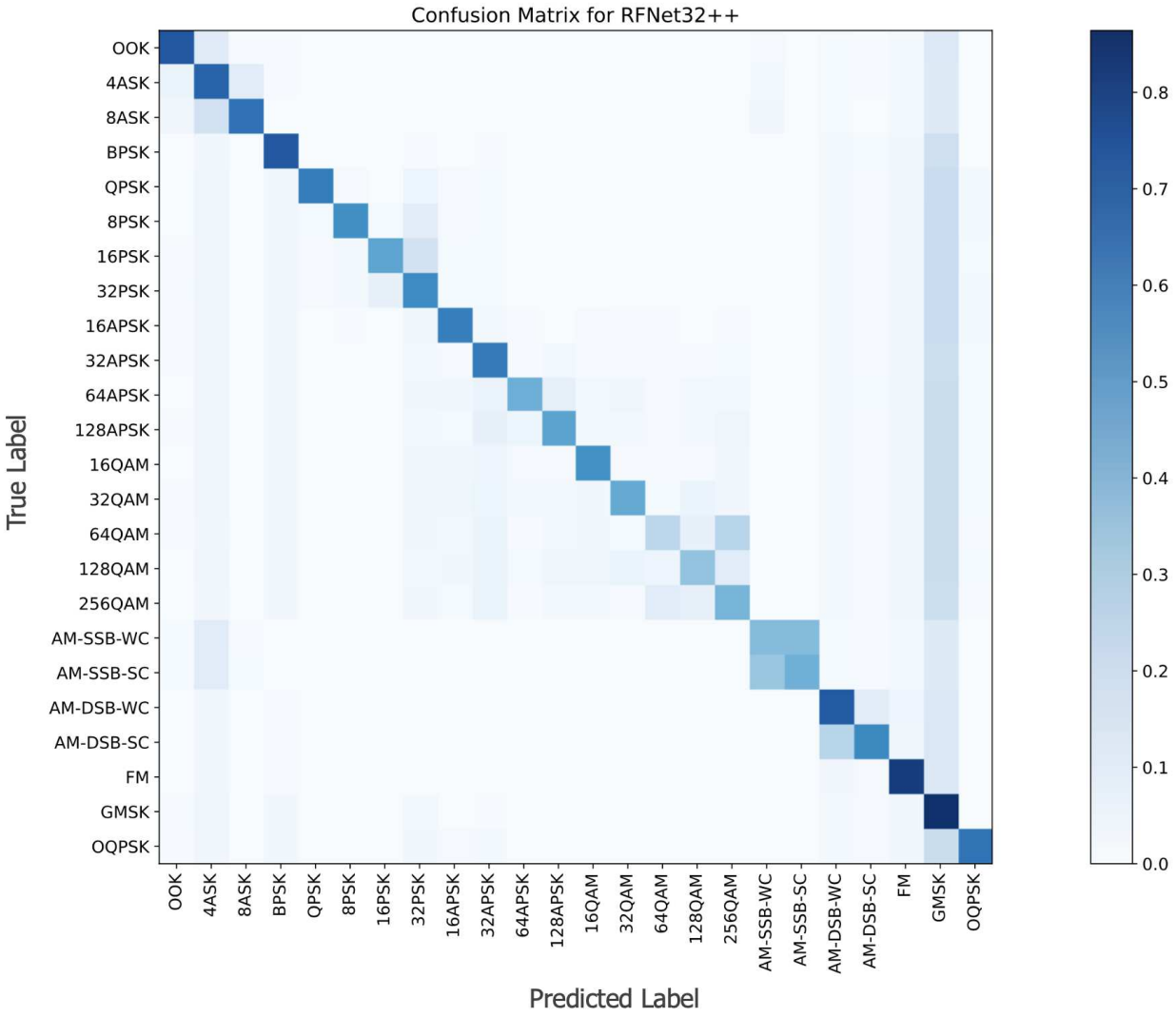


Fig. 7 – Confusion matrix for RF-Net32++

is almost on par with that of RFNet32. However, this is not the case when it comes to RFNet32++ and RFNet48++. Here, although they have less parameters and have lower inference scores, their accuracy is noticeably lower that their successors, RFNet32+ and RFNet48+.

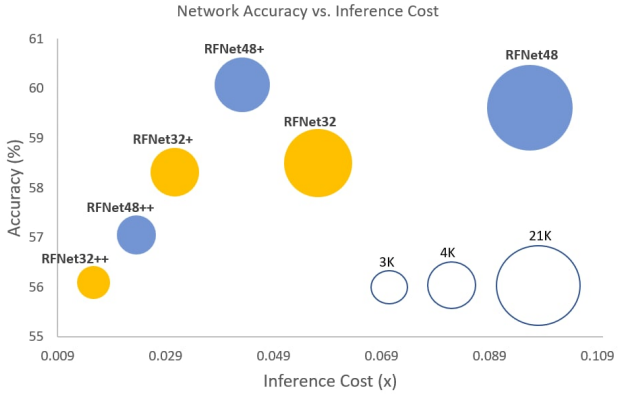


Fig. 8 – The x-axis reports the inference cost and y-axis reports the overall accuracy of the network. Also, the blob size signifies the number of parameters. Therefore the larger the blob, the more parameters the model has. The RFNet48 and RFNet32 family of networks have been color-coded to be better distinguished from one another.

6. CONCLUSION

In this work, we proposed RFNET as a light and practical DNN-based solution to the AMC problem. In their first layer, our proposed family of networks automatically collects a robust set of feature maps. Then, by leveraging pointwise and depth-wise convolutions, they manage to achieve accurate results for the RadioML 2018.0A dataset. We further enhance RFNet by using pruning and quantization techniques, reducing network complexity. The proposed network requires significantly less number of computations, memory space and less parameters. We also proposed a wide variety of networks providing the end user with opportunity to choose within the RFNet family of networks based on application demand and resource availability.

ACKNOWLEDGEMENT

This research has been funded in part or completely by the Computing Hardware for Emerging Intelligent Sensory Applications (COHESA) project. COHESA is financed under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP485577-15.

REFERENCES

[1] Yangyang Chen, Ming Yang, Jiang Long, Dianguo Xu, and Frede Blaabjerg. "A DDS-Based Wait-Free Phase-Continuous Carrier Frequency Modulation Strategy for EMI Reduction in FPGA-Based Motor Drive". In: *IEEE Transactions on Power Electronics* 34.10 (2019), pp. 9619–9631. DOI: 10.1109/TPEL.2019.2891572.

[2] Shahriar Shirvani Moghaddam and A. Habibzadeh. "Cooperative Spectrum Sensing Based on Generalized Likelihood Ratio Test for Cognitive Radio Channels with Unknown Primary User's Power and Colored Noise". In: 08 (Sept. 2018), pp. 217–227. DOI: 10.2174/2210327908666180730092433.

[3] Abdelmohsen Ali and Walaa Hamouda. "Spectrum Monitoring Using Energy Ratio Algorithm for OFDM-Based Cognitive Radio Networks". In: *IEEE Transactions on Wireless Communications* 14.4 (2015), pp. 2257–2268. DOI: 10.1109/TWC.2014.2384024.

[4] Jianping Zheng and Yanfang Lv. "Likelihood-Based Automatic Modulation Classification in OFDM With Index Modulation". In: *IEEE Transactions on Vehicular Technology* 67.9 (2018), pp. 8192–8204. DOI: 10.1109/TVT.2018.2839735.

[5] Jitong Ma and Tianshuang Qiu. "Automatic Modulation Classification Using Cyclic Correntropy Spectrum in Impulsive Noise". In: *IEEE Wireless Communications Letters* 8 (2019), pp. 440–443.

[6] Qinghe Zheng, Mingqiang Yang, Jiajie Yang, Qingrui Zhang, and Xinxin Zhang. "Improvement of Generalization Ability of Deep CNN via Implicit Regularization in Two-Stage Training Process". In: *IEEE Access* 6 (2018), pp. 15844–15869.

[7] Lauren J. Wong, William H. Clark IV au2, Bryse Flowers, R. Michael Buehrer, Alan J. Michaels, and William C. Headley. *The RFML Ecosystem: A Look at the Unique Challenges of Applying Deep Learning to Radio Frequency Applications*. 2020. arXiv: 2010.00432 [eess.SP].

[8] Gihan Janith Mendis, Jin Wei-Kocsis, and Arjuna Madanayake. "Deep Learning Based Radio-Signal Identification With Hardware Design". In: *IEEE Transactions on Aerospace and Electronic Systems* 55.5 (2019), pp. 2516–2531. DOI: 10.1109/TAES.2019.2891155.

[9] Timothy James O'Shea, Tamoghna Roy, and T. Charles Clancy. "Over-the-Air Deep Learning Based Radio Signal Classification". In: *IEEE Journal on Selected Topics in Signal Processing*. 2018. DOI: 10.1109/JSTSP.2018.2797022. arXiv: 1712.04578.

[10] Thien Huynh-The, Cam-Hao Hua, Van-Sang Doan, and Dong-Seong Kim. "Accurate Modulation Classification with Reusable-Feature Convolutional Neural Network". In: *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*. Jan. 2021, pp. 12–17. DOI: 10.1109/ICCE48956.2021.9352042.

[11] Godwin Brown Tunze, Thien Huynh-The, Jae Min Lee, and Dong Seong Kim. "Sparsely Connected CNN for Efficient Automatic Modulation Recognition". In: *IEEE Transactions on Vehicular Technology* (2020). ISSN: 19399359. DOI: 10.1109/TVT.2020.3042638.

[12] Yu Wang, Jie Yang, Miao Liu, and Guan Gui. "LightAMC: Lightweight Automatic Modulation Classification via Deep Learning and Compressive Sensing". In: *IEEE Transactions on Vehicular Technology* (2020). ISSN: 19399359. DOI: 10.1109/TVT.2020.2971001.

[13] Judith Nkechinyere Njoku, Manuel Eugenio Morocho-Cayamcela, and Wansu Lim. "CGDNet: Efficient Hybrid Deep Learning Model for Robust Automatic Modulation Recognition". In: *IEEE Networking Letters* (2021). DOI: 10.1109/lnet.2021.3057637.

[14] Qinghe Zheng, Penghui Zhao, Yang Li, Hongjun Wang, and Yang Yang. "Spectrum interference-based two-level data augmentation method in deep learning for automatic modulation classification". In: *Neural Computing and Applications* 33.13 (Nov. 2020), pp. 7723–7745. DOI: 10.1007/s00521-020-05514-1. URL: https://doi.org/10.1007/s00521-020-05514-1.

[15] Na Wang, Yunxia Liu, Liang Ma, Yang Yang, and Hongjun Wang. "Multidimensional CNN-LSTM Network for Automatic Modulation Classification". In: *Electronics* (2021).

[16] Ya Tu, Yun Lin, Changbo Hou, and Shiwen Mao. "Complex-Valued Networks for Automatic Modulation Classification". In: *IEEE Transactions on Vehicular Technology* 69 (2020), pp. 10085–10089.

[17] Xiaowei Xu, Yukun Ding, Sharon Xiaobo Hu, Michael Niemier, Jason Cong, Yu Hu, and Yiyu Shi. "Scaling for edge inference of deep neural networks". In: *Nature Electronics* (2018). DOI: 10.1038/s41928-018-0059-3.

[18] Song Han, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding". In: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*. 2016. arXiv: 1510.00149.

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML].

[20] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. *Pruning Filters for Efficient ConvNets*. 2017. arXiv: 1608.08710 [cs.CV].

[21] Michela Paganini and Jessica Forde. *On Iterative Neural Network Pruning, Reinitialization, and the Similarity of Masks*. 2020. arXiv: 2001.05050 [cs.LG].

[22] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. *Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks*. 2021. arXiv: 2102.00554 [cs.LG].

[23] Raghuraman Krishnamoorthi. *Quantizing deep convolutional networks for efficient inference: A whitepaper*. 2018. arXiv: 1806.08342 [cs.LG].

[24] Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. *WRPN: Wide Reduced-Precision Networks*. 2017. arXiv: 1709.01134 [cs.CV].

[25] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. *PACT: Parameterized Clipping Activation for Quantized Neural Networks*. 2018. arXiv: 1805.06085 [cs.CV].

[26] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. *Post-training 4-bit quantization of convolution networks for rapid-deployment*. 2019. arXiv: 1810.05723 [cs.CV].

[27] Laurent Sifre and Stéphane Mallat. "Rotation, Scaling and Deformation Invariant Scattering for Texture Discrimination". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 1233–1240. DOI: 10.1109/CVPR.2013.163.

[28] Laurent SIfre and Stéphane Mallat. *Rigid-Motion Scattering for Texture Classification*. 2014. arXiv: 1403.1687 [cs.CV].

[29] Min Lin, Qiang Chen, and Shuicheng Yan. *Network In Network*. 2014. arXiv: 1312.4400 [cs.NE].

[30] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV].

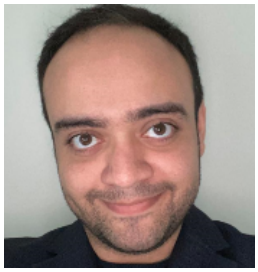
[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV].

[33] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2017. arXiv: 1610.02357 [cs.CV].

[34] Alessandro Pappalardo. *Xilinx/brevitas*. 2021. DOI: 10.5281/zenodo.3333552. URL: https://doi.org/10.5281/zenodo.3333552.

AUTHORS

Mohammad Chegini is currently pursuing a B.Sc. degree in electrical engineering from Shahid Beheshti University. Meanwhile, he has been a research associate at the University of Victoria. He has three publications so far and his primary research interest include applied deep learning and 6G.



Pouya Shiri received his B.Sc. degree in electrical engineering from Shahid Beheshti University and his M.Sc. degree in computer architecture from University of Tehran in 2015 and 2018 respectively. He is currently a Ph.D. student studying deep learning at the University of Victoria. His research interests include designing and optimizing neural networks, and high-performance computing.

Amirali Baniasadi received his B.Sc. degree in electronic engineering from Tehran University and his M.Sc. degree in digital electronics engineering from Sharif University of Technology in 1992, and 1995, respectively. He received his Ph.D. degree in computer engineering from Northwestern University, Illinois, in 2002. He joined the Department of Electrical and Computer Engineering, University of Victoria (U.Vic.) in 2002 where he is currently a professor. He was visiting professor at Sharif University of Technology (2007)and Stanford University, California (2016, 2017). His fields of interest are in computer architecture, neural networks and quantum computing. His research has been funded by Natural Sciences and Engineering Research Council of Canada(NSERC), Consortium for Aerospace Research and Innovation in Canada (CARIC) and MITACS (a Canadian network of centres of excellence). He has over 100 publications including refereed journals, conference papers and technical reports.