IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 May 2024, accepted 30 May 2024, date of publication 3 June 2024, date of current version 13 June 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3409180*

RESEARCH ARTICLE

# Efficient Hardware Design of DNN for RF Signal Modulation Recognition Employing Ternary Weights

**JONGSEOK WOO, KUCHUL JUNG, AND SAIBAL MUKHOPADHYAY, (Fellow, IEEE)**
Georgia Institute of Technology, Atlanta, GA 30332, USA
Corresponding author: Saibal Mukhopadhyay (saibal.mukhopadhyay@ece.gatech.edu)

**ABSTRACT** This paper presents an efficient deep neural network (DNN) accelerator designed for radio frequency (RF) signal modulation recognition. A novel DNN design optimized for mobile applications is demonstrated by combining MobileNetV3-based DNN with a ternary weight quantization. We also propose a new training method called decaying weight training to overcome the performance degradation due to quantization. The effect of the ternary weight quantization is demonstrated with a co-analysis of the classification accuracy and the physical design. The physical design analysis is based on the Application Specific Integrated Circuit (ASIC), and the results show that the ternary weight quantization with the proposed training method minimizes the impact of the quantization while increasing the allowable clock frequency and reducing hardware cost significantly. We also implement the hardware design dedicated to the ternary weight networks to reduce the required number of the multiply and accumulate (MAC) engines. The hardware design is verified on FPGA and the ternary weight-based DNN shows the feasibility of reducing the hardware cost significantly.

**INDEX TERMS** Radio frequency, modulation recognition, machine learning, deep neural network (DNN), MobileNet, quantization, accelerator, ternary weight.

## I. INTRODUCTION

Cognitive radio is the emerging platform for multiband multimode communication systems [1], [2]. The spectrum sensing in the cognitive radio detects the primary/secondary user and utilizes available resources such as communication channels efficiently according to the user type [3]. Modulation recognition is the key spectrum sensing mechanism to determine the user type, therefore the modulation classification accuracy directly affects the performance of the cognitive radio. Traditional automatic modulation classification (AMC) can be mainly categorized into likelihood-based (LB) or feature-based (FB) methods, and these methods require prior information or expert knowledge to build the efficient classification algorithm [4], [5], [6], [7]. Considering the massive multiple-in multiple-out (MIMO) systems, the AMC mechanism employing deep neural networks (DNNs) has advantages over the traditional methods due to availability of the limited vision recognition, which does not rely on prior knowledge or information. The DNN-based RF signal modulation classification such as convolutional neural networks (CNNs) has shown promising performance [8]. In a typical receiver, the radio frequency machine learning (RFML) module which classifies the modulation of the received RF signal is the sequential module placed after down-conversion and digitization. Since the DNN processing block typically requires heavy computational demand, the DNN processing block design often determines the overall receiver performance such as sampling rate and allowable bandwidth. With the increased BW requirements of the incoming RF signals, the software-based deep learning implementation for modulation recognition is not enough for real-time operation, which arouses the need for hardware acceleration. Therefore, a novel comprehensive hardware

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

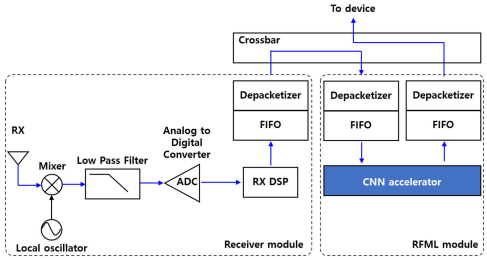



**FIGURE 1. An illustrative RF receiver architecture with an on-chip accelerator for detection of RF signal modulation.**

design with optimal DNN implementation should be further considered.

This paper presents a hardware design methodology for the acceleration of signal modulation detection to enable higher bandwidth with efficient hardware cost and excellent performance (Fig. 1). The key contributions of this paper are:

- We design a DNN optimized for mobile applications by combining MobileNetV3-based DNN with ternary weight quantization.
- We propose a new training method for the ternary weight quantization to minimize the impact of the quantization on the classification accuracy.
- We apply a Common Subexpression Elimination (CSE) algorithm to ternary weight networks to improve the efficiency of the design.
- We implement hardware design dedicated to ternary weight quantization to reduce hardware costs.

In the previous work, we first designed a computationally efficient low-complexity CNN model with quantized (half-precision floating-point to ternary) weights, and the physical hardware designs operating at different precision levels were synthesized [9]. Based on the previous work, we integrate the ternary weight quantization into the MobileNetV3-based DNN, which is known to be designed for mobile applications along with excellent performance. We also propose a new training algorithm for the ternary weight quantization to improve the performance of the quantized DNN model further. In order to design the proposed architecture efficiently, the CSE algorithm is applied prior to the hardware design and the number of MAC operations is reduced significantly. The ASIC design is evaluated by the co-analysis on the system-on-chip level considering throughput, area, power-efficiency of the hardware, the maximum allowable bandwidth of the RF signals, and the classification accuracy of the DNN model. Our analysis shows that the ternary weight quantization with the proposed training method minimizes the impact on classification accuracy while reducing the hardware design cost significantly. Also, the reduced hardware utilization and the improvement in performance is observed in the dedicated hardware design on FPGA.

## II. RELATED WORKS

Timothy and Nathan introduced an RML2016 dataset for RFML and first showed the basic performance of the visual tree convolution neural network (VT-CNN) based modulation recognition task [10]. Based on CNN, Various approaches were made to increase the performance of the DNN-based modulation classification. Jdid et al. [11] proposed a robust AMR leveraging the benefits of both contextual features (CFs) and hand-crafted features (HCFs) with a signal-to-noise ratio (SNR) splitter. In [12], the author built a deep hierarchical network (DHN) based CNN which combined the shallow features with high-level features. The pre-processing of the input signal was also considered, such as the logarithmic constellation mapping ([13]) and short-time Fourier transform (STFT) ([14]). In addition to CNN, the other types of DNN such as recurrent neural network (RNN) and long short-term memory (LSTM) have been introduced for modulation recognition [15], [16], [17], [18], [19], [20].

Since the DNN-based modulation recognition requires more computation resources than the traditional LB or FB method, several works have been implemented to reduce the computational complexity or the latency. Fu et al. [21] applied separable convolutional neural network (S-CNN) to reduce the complexity of the CNN layers. Also, the training efficiency is improved by applying joint training of the multiple edge devices, with a slight degradation of accuracy. Huynh-The et. al. [22] proposed the MCNet to improve the latency. The M-block in MCnet consists of 3 asymmetric convolutional kernels organized in parallel to analyze the multi-scale spatiotemporal signal correlations simultaneously. In [23], the author used a less complex DNN model while improving the classification accuracy by taking advantage of the STFT transform of the input signal. A lightweight AMC model employing exploiting the multi-layer perceptrons (MLP) has been also proposed [24]. The proposed model consists of three modules: the spatiotemporal feature segmentation module, the local multi-scale temporal feature fusion module, and the Patches-Mixer module. This model showed a relatively small number of parameters, but the small observation window by splitting the original sequence caused miss-classification of analog-modulated signal. Building a compressed model exploiting quantization or pruning has been the conventional method for hardware acceleration [25]. Tridgell et. al. [26] applied ternary weight quantization to the visual geometry group (VGG) net to reduce the hardware resources and the classification accuracy degradation from 90.9% to 82.1% was observed. Kumar et al. [27] proposed a QMCNet which quantized the input, weight, and activation to 4bit, 5bit, and 6bit respectively, and a RUNet which utilized the 6-bit quantization and the residual unit. Also, the iterative weight pruning is implemented to increase the sparsity of the network and showed the best accuracy of 90.59% (QMCNet) and 94.46% (RUNet).

Several attempts to reduce the hardware cost often suffer from the degradation of the performance or additional iterative steps to increase the sparsity of the networks are required. The motivation of this study is to overcome the challenges of acceleration of the DNN-based modulation recognition while minimizing the impact on the perfor-

mance and eliminating additional pre-processing or iterative methods.

## III. DNN DESIGN FOR MODULATION CLASSIFICATION

### A. WEIGHT QUANTIZATION

The major operation of each hidden layer in neural networks is to calculate the weighted sum of the input. This operation is implemented by the array of the MAC engines in hardware. Mostly the bottleneck of the MAC engine is multiplier, and the low-precision parameters can simplify the multiplication. If the parameters can be binarized, it can simplify the multiplication dramatically since the real multiplier circuit is no longer required [28], [29]. In this work, we consider ternary weight quantization instead of binary weight to improve expressive ability while still not requiring a multiplier circuit [30].

#### 1) TERNARY WEIGHT QUANTIZATION

Ternary weight networks are similar to Binary weight networks. However, the weights are constrained to $-1$,0, and $+1$, whereas the binary weight networks use $-1/+1$ weights. As shown in Fig. 2, the weight conversion to ternary weights is computed to minimize the Euclidean distance between the original weight and the ternary-valued weight (1). ternary weight conversion with scaling factor is also used to minimize the Euclidean distance further. In this case, the weight values after conversion are $-\alpha$, 0, and $+\alpha$ (2).

$$W^t = \text{argmin}(|| W - W^t ||)_2^2 \tag{1}$$

$$W^t = \text{argmin}(|| W - \alpha W^t ||)_2^2 \tag{2}$$

The ternary weight conversion is the function with the $\Delta$ parameter, which is the boundary condition for the conversion(Fig. 2b). The original weights are compared with $\Delta$, $-\Delta$ to determine the value after the conversion (3).

$$W_i^t = f_t(W_i \mid \Delta) = \begin{cases} +1(\alpha), & \text{if } W_i > \Delta \\ 0, & \text{if } \mid W_i \mid \leq \Delta \\ -1(-\alpha), & \text{if } W_i < -\Delta \end{cases} \tag{3}$$

Based on the conversion above, (2) can be represented as:

$$\text{argmin}_{\Delta>0}(\sum_{i=1}^{n} \mid W_i \mid^2 -2\alpha \sum_{i\in I_\Delta} \mid W_i \mid +\alpha^2 \mid I_\Delta \mid) \tag{4}$$

n is the number of the weights for the current channel and $\mid I_\Delta \mid$ denotes the number of $I_\Delta$ elements where $I_\Delta = \{i \mid\mid W_i \mid > \Delta\}$. The optimal scaling factor $\alpha$ is the expected magnitude of the weights which are larger than $\Delta$, as shown in (5). By substituting $\alpha$ into 4, $\Delta$ is computed by (6).

$$\alpha_\Delta^* = \frac{1}{\mid I_\Delta \mid} \sum_{i\in I_\Delta} \mid W_i \mid \tag{5}$$

$$\Delta^* = \text{argmax}_{\Delta>0} \frac{1}{\mid I_\Delta \mid} (\sum_{i\in I_\Delta} \mid W_i \mid)^2 \tag{6}$$

It is hard to apply these computation methods to the software or hardware implementation since there is no straightforward solution. If there is an assumption in the distribution of the weights, it can simplify the computation. If the weights follow the normal distribution, the delta can be simply computed using the expectation of the absolute value of the weight (7).

$$\Delta^* \approx \frac{0.7}{n} \sum_{i=1}^{n} \mid W_i \mid \tag{7}$$

In this task, we convert the ternary weights based on the assumption above and exclude the scaling factor in the conversion to maximize the simplicity of the hardware design.

#### 2) TRAINING WITH TERNARY WEIGHT QUANTIZATION

The software-based DNN model trains the model and inferences with the 32-bit floating-point type of the weight [31]. For the application of the ternary weight quantization to the training, DNN should implement forward propagation with ternary weight but the gradient descent from backward propagation updates the weights as floating point type. Therefore, an additional conversion step is required. The ternary weight conversion step is located between the forward propagation and the backward propagation (Fig. 3). In the loop, the weights are updated by gradient at backward propagation. These updated weights are used to update the $\Delta$, and weights are also updated by the ternary weight conversion with updated $\Delta$.

The gradient descent updates the weights in the neural network by computing the derivative from the calculated loss. (8) shows the gradient of the weight $W_i$ in the convolution layer used for the update.

$$\frac{\partial L}{\partial W_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial W_i} \tag{8}$$

where L is the calculated loss and O is the output of the current layer. Since the partial derivative of the current layer output with respect to the weight is equal to the input, the gradient is computed as shown in (9).

$$\frac{\partial L}{\partial W_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * X_k \tag{9}$$

If the weight used in the forward propagation is zero, the network blocks the propagation of that neuron and the partial derivative of the current layer output with respect to the weight is not equal to the input. Therefore, the gradient can not be computed properly. Conventional ternary weight quantization in neural networks converts a large number of weights ($>$ 50% of the total weights) to zero. The large number of the zero weights from the beginning of the training can cause performance degradation during the training.

#### 3) PROPOSED TRAINING METHOD

In order to overcome the training performance degradation due to the ternary weight quantization, we propose the

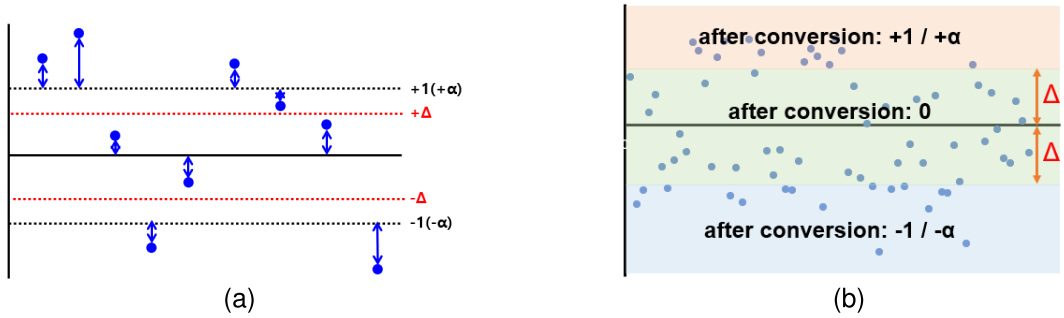



**FIGURE 2.** Ternary weight networks (a) The Euclidean distance between the original weight and the ternary weight. (b) Ternary weight conversion.

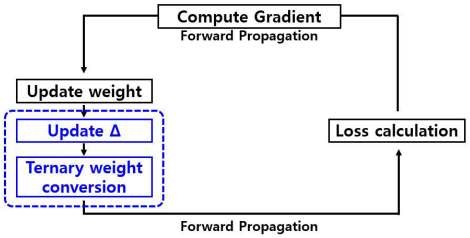



**FIGURE 3.** Training process with ternary weight quantization.

training with the decaying weight. During training, this method converts the weights to $\{-1, -\frac{1}{2^j}, \frac{1}{2^j}, 1\}$ as shown in (10), instead of $\{-1, 0, 1\}$.

$$W_i^t = f_t(W_i \mid \Delta) = \begin{cases} +1, & \text{if } W_i > \Delta \\ \frac{1}{2^j}, & \text{if } 0 < W_i \leq \Delta \\ -\frac{1}{2^j}, & \text{if } -\Delta \leq W_i \leq 0 \\ -1, & \text{if } W_i < -\Delta \end{cases} \tag{10}$$

The parameter j increases as the iteration increases, making $\{-\frac{1}{2^j}, \frac{1}{2^j}\}$ close to zero. Algorithm 1 shows the process of the proposed method. The proposed method has the following characteristics.

- All the weights have non-zero values so that the DNN can compute gradient descent properly.
- By decaying mechanism, $\frac{1}{2^j}$ becomes a very small value at the end of the training, enabling the conventional ternary weight conversion during the inference.
- The proposed method has 4 types of weight. Therefore, the bit width for the weight remains the same as the ternary weight(2-bit) and the additional memory for only one global parameter (j) is required.
- The multiplier inside the MAC engine remains simple since the multiplication/division by a power of 2 can be easily implemented by adding/subtracting the exponent bit.

### B. DNN MODEL FOR MODULATION RECOGNITION

The modulation recognition task is implemented by using the MobileNetV3-Small architecture, considering the mobile

**Algorithm 1** Training With Decaying Weight

**Parameter** j:power of the initial weight value
**Parameter** k:weight decaying period

```
n=1
for Training do
    if mod(n,k) == 0 then
        j=j+1
    end if
    convert weights to {−1, −1/2^j, 1/2^j, 1}
    Implement forward propagation
    Compute Gradient Descent
    Update Weights
    n=n+1
end for
convert weights to −1, 0, 1
```

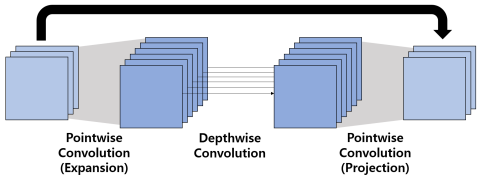



**FIGURE 4.** Inverted residual bottleneck.

application. MobileNetV3 was introduced by Google as an efficient DNN architecture with excellent performance [32]. MobilenetV1 utilizes depth-wise separable convolution which implements simple convolution operation through the depth-wise convolution, followed by point-wise convolution to transform the channel [33]. MobilenetV2 proposed a bottleneck depth-separable convolution block (Fig. 4) with residuals that transforms channels with point-wise convolution, performs depth-wise convolution with the expanded channels, and adds input (residual block) [34]. The inverted residual bottleneck layers enable memory-efficient implementation, which makes MobileNetV2 faster than V1. MobileNetV3 includes a squeeze and excite (SE) block (Fig. 5) between the depth-wise convolution

**TABLE 1.** DNN for modulation recognition.

| Input | Operator | expansion | output channel | Activation |
|---|---|---|---|---|
| 2x1024x1 | conv2d, filter size 3, stride 2 | | 16 | Hard Swish |
| 2x512x16 | bottleneck(SE), filter size 3, stride 2 | 16 | 16 | ReLU6 |
| 2x256x16 | bottleneck, filter size 3, stride 2 | 72 | 24 | ReLU6 |
| 2x128x24 | bottleneck, filter size 3, stride 1 | 88 | 24 | ReLU6 |
| 2x128x24 | bottleneck(SE), filter size 5, stride 2 | 96 | 40 | Hard Swish |
| 2x128x40 | bottleneck(SE), filter size 5, stride 1 | 240 | 40 | Hard Swish |
| 2x128x40 | bottleneck(SE), filter size 5, stride 1 | 240 | 40 | Hard Swish |
| 2x128x40 | bottleneck(SE), filter size 5, stride 1 | 120 | 48 | Hard Swish |
| 2x128x48 | bottleneck(SE), filter size 5, stride 1 | 144 | 48 | Hard Swish |
| 2x128x48 | bottleneck(SE), filter size 5, stride 2 | 288 | 56 | Hard Swish |
| 2x64x48 | bottleneck(SE), filter size 5, stride 1 | 288 | 56 | Hard Swish |
| 2x64x48 | bottleneck(SE), filter size 5, stride 1 | 288 | 56 | Hard Swish |
| 2x64x48 | conv2d, filter size 1, stride 1 | | 288 | Hard Swish |
| 2x64x288 | pool | | 288 | |
| 1x1x288 | conv2d, filter size 1, stride 1 | | 1024 | Hard Swish |
| 1x1x1024 | conv2d, filter size 1, stride 1 | | 24 | |

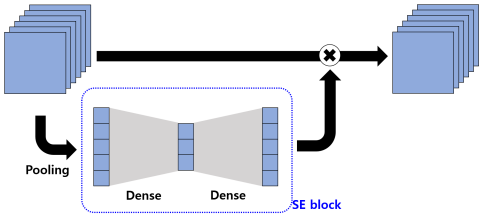



**FIGURE 5.** Squeeze and excite block.

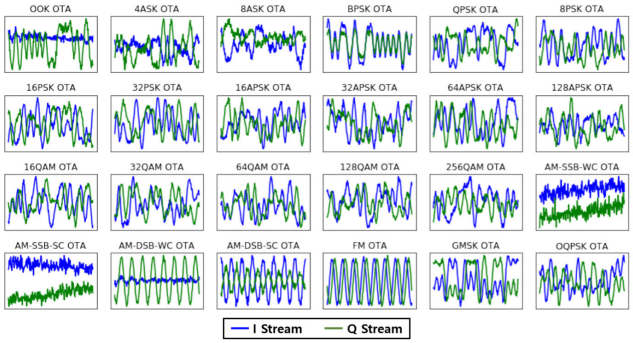



**FIGURE 6.** RadioML2018.01a dataset.

layer and the projection layer to squeeze from the largest representation inside the bottleneck block. This SE block helps MobileNetV3 to provide faster and more accurate operation. MobileNetV3 also uses hard-swish non-linearity for certain bottleneck layers, which is a modified version of swish non-linearity by replacing the sigmoid function with the equation including ReLU6 non-linearity to reduce computational cost (11). From the baseline MobileNetV3 architecture, we modified some channels to reduce the hardware cost with negligible effect on the classification accuracy. Table 1 shows the DNN model used for this modulation recognition task.

$$Hard - swish(x) = x\frac{ReLU6(x+3)}{6} \tag{11}$$

## IV. MODULATION CLASSIFICATION RESULTS

### A. DATASET

The RadioML2018.01a dataset created by O'Shea et al. is used to train and evaluate the DNN model [18]. The samples are captured through over-the-air transmission channels. Each sample is an I/Q pair complex time-domain 1,024 data vector. The number of modulation classes is 24 and each modulated signal was recorded at various signal-to-noise ratio (SNR) levels. The total size of the dataset is 2,555,904 labeled samples with 4,096 samples for each modulation and SNR pair. The 24 modulation classes of the signals are: OOK, 4ASK, 8ASK, BPSK, QPSK, 8PSK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, AM-SSBWC, AM-SSB-SC, AM-DSB-WC, AM-DSB-SC, FM, GMSK, and OQPSK. Fig 6 shows the I/Q stream of the dataset at 10dB SNR.

### B. CLASSIFICATION ACCURACY WITH PROPOSED METHOD

Fig. 7 shows the classification accuracy of the baseline and quantized models. The proposed work uses a 16-bit floating point as the input and the activation. The training epoch is 200 and the learning rate is 0.001. The proposed decaying weight training starts with the initial value of $1/2^2$ and reduces the weight magnitude by half after 20 iterations, making the magnitude $1/2^{11}$ at the end of the training. The MobileNetV3-based modulation task shows excellent performance with the signal over 6dB SNR, showing 92.1% average accuracy. The quantization to a 16-bit floating point barely affects the performance of the DNN showing identical classification accuracy. When the parameters are quantized to ternary weights, the average classification accuracy drops to 87.8%(4.3% lower than the reference) due to the limited capability of weight representation and training. When the decaying weight training was applied, the DNN is fitted to the designated modulation classification task better than

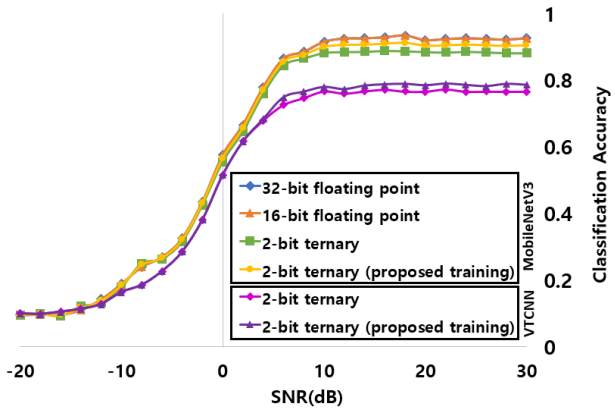



**FIGURE 7.** Classification accuracy.

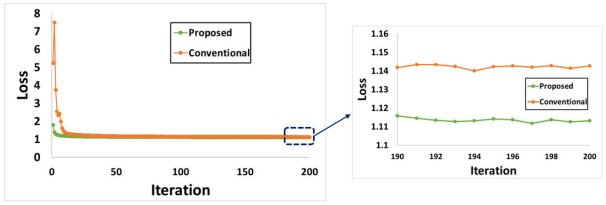



**FIGURE 8.** Loss curve with validation data.

the conventional method, showing the reduced loss with the validation set (Fig. 8). As a result, the proposed training method improved the classification accuracy to 90.1% even with the ternary weight representation. The improvement of the classification accuracy with the proposed training method is also observed in VTCNN model, which is used in previous work (76.2% → 78.0%).

Fig. 9 and Fig. 10 show the confusion matrix with the received signal above 6dB SNR with the conventional and the proposed training method, respectively. The distribution in the confusion matrix shows that the modulation classification mechanism has difficulty in discriminating between the high-order modulation schemes. The dataset used in this task includes M-ary PSK and QAM modulations and most of the miss-classification comes from the confusion between them. Also, the confusion between the Suppressed Carrier (SC) and the Without Carrier (WC) signals in AM modulation is observed due to their similarity. The proposed training method helps the model to classify these difficult cases better than the conventional training.

### C. COMPARISON WITH PRIOR WORKS

We also compared the classification accuracy of the proposed method with the prior work including our previous work. The models named T-CNN [9], STFT-CNN [23], CLDNN [19], MCLDNN [20], MCNet [22], VGG10 [26], and QMC-Net/RUNet [27] are selected for the performance comparison. Fig. 11 shows the classification accuracy comparison and the Table 2 summarizes the complexity, quantization type, and classification accuracy at 30dB SNR. Since [9], [23],

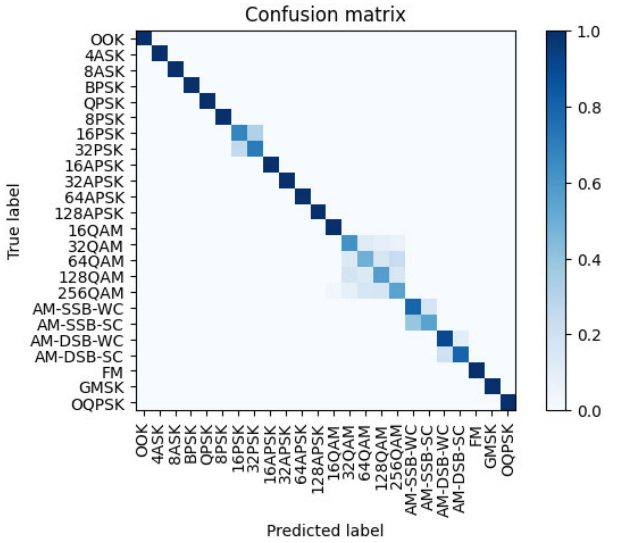



**FIGURE 9.** Confusion matrix of ternary weight model.

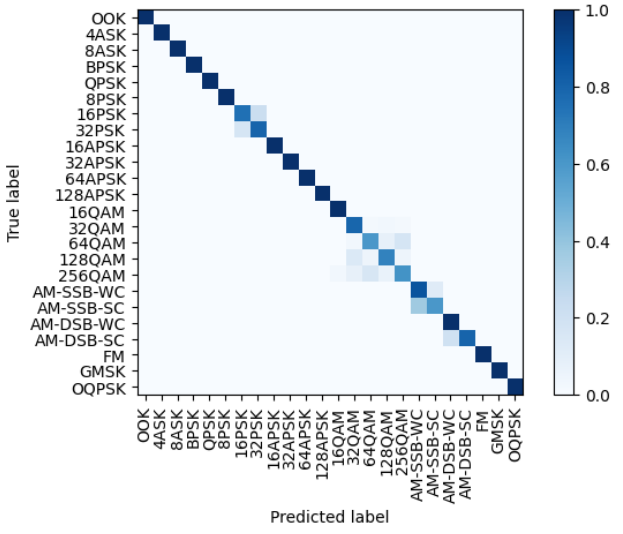



**FIGURE 10.** Confusion matrix of ternary weight model with proposed training.

[26], [27] and our work use the lower precision type than 32-bit floating point, the bit-level operation per second (BOPs) which means the number of bit-level arithmetic operations per second is also calculated to estimate the real circuit size required for each model. Since the multiplication with the ternary weights reduces the required number of the logic circuits significantly, the proposed method shows the excellent performance even with the lowest number of bit-level arithmetic operations per second.

### D. APPLICATION TO IMAGE CLASSIFICATION

The proposed training method is also applied to the image classification task to evaluate the proposed model with the image dataset. The CIFAR-100 and tiny-ImageNet are used as a dataset for image classification tasks. The proposed

**TABLE 2.** Accuracy comparison with prior works.

| Model | Parameter | Capacity(bit) | FLOPs | BOPs | Quantization | Accuracy(30dB SNR, %) |
|---|---|---|---|---|---|---|
| CLDNN [19] | 0.3M | 8.1M | 3.6M | 753.6M | - | 77.7 |
| MCLDNN [20] | 0.4M | 13.1M | 6.7M | 1,274.3M | - | 90.9 |
| MCNet [22] | 0.1M | 4.5M | 2.2M | 511.8M | - | 77.9 |
| T-CNN [9] (Previous Work) | 0.5M | 1.1M | 6.6M | 135.8M | 2bit | 76.6 |
| STFT-CNN [23] | 0.4M | 6.4M | 6.5M | 242.7M | 16bit | 79.8 |
| VGG10 [26] | 0.6M | 1.3M | 9.9M | 187.5M | 2bit | 82.1 |
| QMCNet [27] | 0.7M | 3.3M | 9.5M | 104.9M | 4/5/6 bit | 90.6 |
| RUNet [27] | 0.4M | 2.2M | 5.6M | 89.2M | 6bit | 94.5 |
| **This Work** | 0.4M | 0.8M | 4.7M | 42.8M | 16/2 bit | 91.3 |

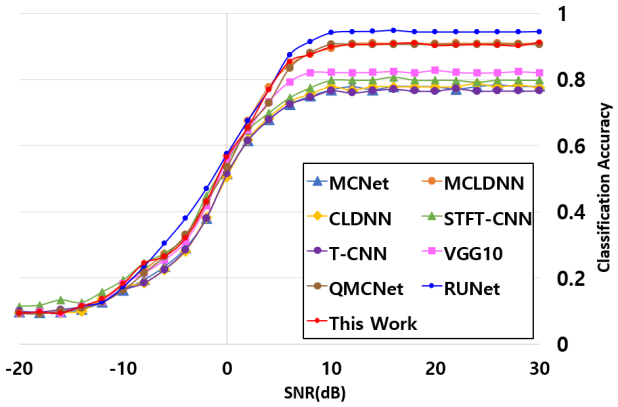



**FIGURE 11.** Accuracy comparison with prior works.

**TABLE 3.** Image classification results.

| Dataset | Type | Top-1 Accuracy(%) | Top-5 Accuracy(%) |
|---|---|---|---|
| CIFAR-100 | Ternary | 67.78 | 85.76 |
| | Ternary+Proposed Training | 67.79 | 85.74 |
| tiny-ImageNet | Ternary | 55.67 | 77.41 |
| | Ternary+Proposed Training | 55.65 | 77.42 |

training method slightly reduces the validation loss during the training, but the improvement in the classification accuracy is negligible (Table 3). The proposed training method is suitable for the modulation classification of the RF signal, which incorporates the data with the various noise levels including low SNR.

## V. PHYSICAL DESIGN OF THE ACCELERATOR

### A. DNN ACCELERATOR DESIGN

The neural network (NN) such as CNN and dense connection consume the most computations in the entire DNN operation, and the output of each NN is the weighted sum of the input. Therefore the DNN accelerator architecture mainly consists of the processing element (PE) array, and each PE implements the MAC operation [35], [36], [37]. The MAC engine includes a complex multiplication that utilizes element-wise multiplication of the real and imaginary parts (Fig. 12) and accumulation. Fig. 13 shows the DNN accelerator structure. At the beginning of the operation, all weights are stored in memory. During the process, the input streams and the weights are transferred to the global buffer and assigned to PE according to the network structure. The results of the PE are then transferred to the global buffer to store the partial sums. Since this process continues until the PE array completes all the MAC operations of DNN, the number of required MAC operations mainly affects the hardware design such as the PE array size and the throughput.

**TABLE 4.** Number of MAC operations.

| Module | number of MAC | | | |
|---|---|---|---|---|
| | Expansion | D-wise convolution | SE | Projection |
| conv2d | 96 | | | |
| bottleneck(SE) | 256 | 48 | 256 | 256 |
| bottleneck | 576 | 108 | - | 864 |
| bottleneck | 1,056 | 132 | - | 1,056 |
| bottleneck(SE) | 1,152 | 240 | 1,536 | 1,920 |
| bottleneck(SE) | 4,800 | 600 | 7,680 | 4,800 |
| bottleneck(SE) | 4,800 | 600 | 7,680 | 4,800 |
| bottleneck(SE) | 2,400 | 300 | 1,920 | 2,880 |
| bottleneck(SE) | 3,456 | 360 | 3,456 | 3,456 |
| bottleneck(SE) | 6,912 | 720 | 11,520 | 6,912 |
| bottleneck(SE) | 13,824 | 1,440 | 41,472 | 13,824 |
| bottleneck(SE) | 13,824 | 1,440 | 41,472 | 13,824 |
| conv2d | 13,824 | | | |
| conv2d | 294,912 | | | |
| conv2d | 24,576 | | | |

**TABLE 5.** Number of MAC operations before/after CSE.

| | number of MAC(before CSE) | number of MAC(after CSE) | ratio |
|---|---|---|---|
| conv2d | 20 | 7 | 0.35 |
| bottleneck(SE) | 425 | 259 | 0.61 |
| bottleneck | 792 | 442 | 0.56 |
| bottleneck | 1,164 | 638 | 0.55 |
| bottleneck(SE) | 2,512 | 1,320 | 0.53 |
| bottleneck(SE) | 9,087 | 4,463 | 0.49 |
| bottleneck(SE) | 9,075 | 4,486 | 0.49 |
| bottleneck(SE) | 3,752 | 1,970 | 0.53 |
| bottleneck(SE) | 5,414 | 2,783 | 0.51 |
| bottleneck(SE) | 13,385 | 6,366 | 0.48 |
| bottleneck(SE) | 35,745 | 15,883 | 0.44 |
| bottleneck(SE) | 35,865 | 15,852 | 0.44 |
| conv2d | 6,855 | 2,883 | 0.42 |
| conv2d | 140,942 | 69,340 | 0.49 |
| conv2d | 11,745 | 5,778 | 0.49 |
| ∗zero weight computations are excluded. | | | |

### B. COMMON SUBEXPRESSION ELIMINATION

Even though the MobileNetV3-based architecture shows an efficient network by taking advantage of the depth-wise separable convolution, it still suffers from the extensive computation implementing the point-wise convolution in the

**TABLE 6.** Synthesis results.

| | Input | 32-bit | 16-bit | |
|---|---|---|---|---|
| | Multiplicand | 32-bit | 2-bit ternary | 2-bit ternary (Decaying Weight Training) |
| | Output | 32-bit | 16-bit | |
| MAC | Frequency | 1.82GHz | 3.43GHz | 3.43GHz |
| | Area | 4829.3$um^2$ | 815.6$um^2$ | 837.2$um^2$ |
| | Power | 6.22mW | 1.47mW | 1.52mW |
| SRAM | Capacity(Mbit) | 13.05 | 0.82 | 0.82 |
| | Area($mm^2$) | 3.59 | 0.25 | 0.25 |
| | Power(W) | 6.34 | 0.44 | 0.44 |
| | Efficiency (Gparams/W/s) | 8.15 | 130.47 | 130.47 |

**TABLE 7.** DNN design comparison.

| | | T-CNN [9] | STFT-CNN [23] | This Work |
|---|---|---|---|---|
| accuracy(%, 30dB SNR) | | 76.6 | 79.8 | 91.3 |
| Quantization(bit) | Input | 32 | 16 | 16 |
| | Weight | 2 | 16 | 2 |
| | Activation | 32 | 16 | 16 |
| Parameter | Frequency | 300MHz | | |
| | Area | 3.18$mm^2$ | 4.25$mm^2$ | 1.87$mm^2$ |
| | Power | 3.62W | 4.36W | 2.19W |
| Performance | TOPS/W | 30.4 | 24.5 | 35.8 |
| | TOPS/$mm^2$ | 34.7 | 25.5 | 42.0 |

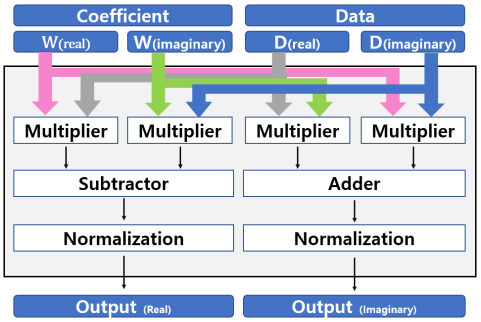



**FIGURE 12.** Complex multiplier architecture.

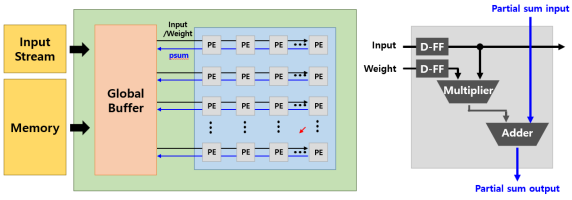



**FIGURE 13.** The CNN accelerator.

expansion/projection layer and the dense connection in SE block (Table 4). In order to take advantage of the ternary weight quantization, we applied CSE to reduce the required number of PEs further. The CSE was introduced by Hsiao et al. [38] to reduce area cost in bit-level equations. This CSE also can be applied to the ternary weight networks if the flipped sign bit value due to the $-1$ weight is also considered [39]. Fig. 14 and Algorithm 2 show how the CSE is implemented in the ternary weight networks. This algorithm finds the most frequent subexpression and changes it to a new variable repeatedly. In each iteration,

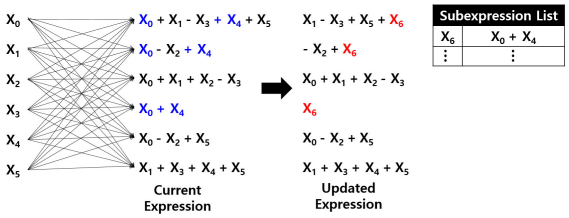



**FIGURE 14.** CSE description.

the most frequent subexpression is removed and only one adder/subtractor is added to make the new variable.

**Algorithm 2** CSE

```
while do
    Build N(i) array which lists two-term subexpressions.
    Store the frequency of the i^th expression to N[i].
    if allN[i] == 1 then
        break
    end if
    Find the most frequent subexpression N[i*]
    Replace all subexpression N[i*] with new variable x*
end while
```

The sparsity through CSE increases significantly as the number of expressions increases. Considering this, we applied CSE algorithm to the design of the point-wise convolution layers in the inverted residual bottleneck block and the dense layers in the squeeze and excite block.

## VI. HARDWARE IMPLEMENTATION RESULTS

### A. APPLICATION OF CSE TO TERNARY WEIGHT DNN

In order to reduce the number of MAC engines, we export the parameters from the trained model and apply the CSE algorithm before building Register Transfer Level (RTL). Table 5 shows the number of MAC operations of each block after excluding the operations with zero weight. The CSE algorithm reduced the required number of MAC operations in each block to 40 $\sim$ 60%, making the total number of MAC operations reduced to 48.7%.

**TABLE 8.** FPGA implementation comparison.

| | | Tridgell et. al. [26] | Kumar et. al. [27] | | Jongseok et. al. [9] | Kuchul et. al. [23] | This Work |
|---|---|---|---|---|---|---|---|
| Model | | VGG10 | QMCNet | RUNet | T-CNN | STFT-CNN | MobileNetV3 |
| Quantization(bit) | Input | 32 | 4 | 6 | 32 | 16 | 16 |
| | Weight | 2 | 5 | 6 | 2 | 16 | 2 |
| | Activation | 32 | 6 | 6 | 32 | 16 | 16 |
| Accuracy(%, 30dB SNR) | | 82.1 | 90.6 | 94.5 | 76.6 | 79.8 | 91.3 |
| Utilization | LUT(Logic) | 121k | 61.4k | 34.6k | 61.2k | 97.9k | 31.2k |
| | BRAM | 162 | 57 | 40 | 72 | 133 | 22 |
| | FF | 198.0k | 40.5k | 21.4k | 92.3k | 139.2k | 25.8k |
| | DSPs | 710 | - | - | 391 | 578 | 162 |
| Frequency(MHz) | | 250 | | | 250 | 250 | 250 |
| Power(W) | | | | | 8.7 | 10.5 | 4.2 |
| Performance | GOPS | | | | 185.7 | 178.7 | 263.4 |
| | GOPS/W | | | | 21.3 | 17.0 | 62.7 |
| | GOPS/DSP | | | | 0.5 | 0.3 | 1.6 |

### B. ASIC DESIGN RESULTS

The RTL for the DNN implementation is generated using Verilog language. The ASIC design is synthesized with TSMC 28nm standard cell library and Synopsis Design Compiler tool (Version 2022.12).

#### 1) PERFORMANCE OF MAC ENGINE WITH QUANTIZED WEIGHT

Table 6 shows the synthesis results. In the synthesis, the primary bottleneck of the MAC engine is the multiplier. Since the converted ternary weights have only 1-bit magnitude ($-1$, 0, and $+1$), the multiplication with the ternary weight can be implemented simply by flipping the sign bit or returning the 0 value. The ternary weight quantization makes the multiplication simpler, and the maximum frequency is increased by 88% compared to the reference MAC unit (32-bit floating-point as multiplicand). Other than the training with the conventional ternary weight, the training with the decaying weight uses 1/(power of 2) value as weight. The division by power of 2 can be simply implemented by increasing or decreasing the exponent in floating point representation. Therefore, the area/power increase due to the decaying weight is negligible.

#### 2) MEMORY USAGE

The impact of the quantization on the design of the on-chip SRAM for the CNN layers is estimated with the 28nm SRAM compiler to generate the total SRAM array using a 16KB sub-bank operating at 1.96GHz maximum frequency. The total SRAM power is estimated assuming all sub-banks are active. Since the SRAM is mainly used to store the weights, the required capacity, area, and power are proportional to the bit width of the weight, showing ∼15x decrease in the ternary weight case. Since the decaying weight training only requires one more parameter compared with the conventional ternary weight, which is the decaying parameter, memory usage is almost the same as the conventional ternary weight case.

#### 3) DNN DESIGN

From the CSE algorithm result and the MAC engine with quantized weight, we build RTL for the entire DNN. Table 7 shows the hardware design results and the comparison with prior works about on-chip acceleration. The baseline model used in [9], and [23] is VTCNN and our MobileNetV3-based DNN with the proposed training method shows better accuracy even with the 16-bit quantized input/activation and the ternary weight. Therefore, our work shows better performance along with the lower area and power than the previous work, showing 35.8 TOPS/W and 42.0 TOPS/$mm^2$.

### C. FPGA IMPLEMENTATION

We implemented the DNN model on the Xilinx ZCU102 evaluation board for verification. We use Vivado 2022.1 to synthesize designs, and the PYNQ framework to verify the functionality of the design. From the RTL generated after CSE, we generate the bitstream to implement this DNN on ZCU102 FPGA with a clocking frequency of 250MHz, 497k throughput, and 7us latency. Table 8 shows the resource utilization of the proposed work and the comparison with the prior work which includes quantization. The comparison with the VTCNN-based previous work ([9], [23]) shows better resource utilization and the performance (62.7 GOPS/W and 1.6 GOPS/DSP), which are similar to the ASIC design results. Tridgell et al. [26] implemented a modulation classification task with VGG10 [18]-based ternary weight network and achieved the best accuracy of 82.1 % which is better than the VTCNN-based modulation recognition, but the accuracy is still lower than our work (91.3% at 30dB SNR) and resource utilization is larger. Kumar et al. [27] proposed quantized QMCnet and RUnet with iterative pruning-based training to increase the sparsity of the network. The RUNet in this work showed superior classification accuracy (94.5% at 30dB SNR), but the iterative pruning method is applied to increase the sparsity of the network and this makes the training take significantly longer than our work (25 Hrs. and 27 min vs 2 Hrs. and 18 min).

## VII. CONCLUSION

In this paper, we present the efficient DNN-based modulation classification by combining the MobileNetV3 architecture with ternary weight quantization. The ternary weight quantization in the MobileNetV3 architecture shows significant improvement in hardware design such as operating frequency, power, and area with a 4.3% accuracy degradation of the signal above 6dB SNR. We also propose a new training method called decaying weight training to compensate for the limited model-fitting capability under ternary weight networks. The proposed method can improve the classification accuracy by 2.3% with negligible impact on the memory or the computational demand. The ASIC design results show that the MobileNetV3-based DNN with the proposed training method has lower power/area and better performance(35.8 TOPS/W, 42.0 TOPS/$mm^2$) than the previous work, along with much better accuracy. The FPGA implementation with the dedicated hardware design also shows excellent performance (62.7 GOPS/W and 1.6 GOPS/DSP) and proves that the proposed design has the advantage over the prior works considering the classification accuracy, the resource utilization, the performance, and the intensity of the training.

## REFERENCES

[1] J. Mitola and G. Q. Maguire, "Cognitive radio: Making software radios more personal," *IEEE Pers. Commun.*, vol. 6, no. 4, pp. 13–18, Aug. 1999, doi: 10.1109/98.788210.

[2] B. Wang and K. J. R. Liu, "Advances in cognitive radio networks: A survey," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 1, pp. 5–23, Feb. 2011, doi: 10.1109/JSTSP.2010.2093210.

[3] E. Axell, G. Leus, E. G. Larsson, and H. V. Poor, "Spectrum sensing for cognitive radio : State-of-the-Art and recent advances," *IEEE Signal Process. Mag.*, vol. 29, no. 3, pp. 101–116, May 2012, doi: 10.1109/MSP.2012.2183771.

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

[5] M. Turan, M. Öner, and H. A. Çirpan, "Joint modulation classification and antenna number detection for MIMO systems," *IEEE Commun. Lett.*, vol. 20, no. 1, pp. 193–196, Jan. 2016.

[6] D. Grimaldi, S. Rapuano, and L. De Vito, "An automatic digital modulation classifier for measurement on telecommunication networks," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 5, pp. 1711–1720, Oct. 2007.

[7] S. Aer, M. Diao, X. Zhang, and X. Zhang, "Automatic modulation classification strategy based on novel feature processing," in *Proc. 7th Int. Conf. Signal Image Process. (ICSIP)*, Jul. 2022, pp. 74–80.

[8] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Proc. Int. Conf. Eng. Appl. Neural Netw. (EANN)*, Aberdeen, U.K., 2016, pp. 213–226.

[9] J. Woo, K. Jung, and S. Mukhopadhyay, "Efficient on-chip acceleration of machine learning models for detection of RF signal modulation," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Atlanta, GA, USA, Jun. 2021, pp. 74–77, doi: 10.1109/IMS19712.2021.9574932.

[10] O. Timothy and W. Nathan, "Radio machine learning dataset generation with GNU radio," in *Proc. GNU Radio Conf.*, Sep. 2016, pp. 1–6. [Online]. Available: https://pubs.gnuradio.org/index.php/grcon/article/view/11

[11] B. Jdid, W. H. Lim, I. Dayoub, K. Hassan, and M. R. B. Mohamed Juhari, "Robust automatic modulation recognition through joint contribution of hand-crafted and contextual features," *IEEE Access*, vol. 9, pp. 104530–104546, 2021, doi: 10.1109/ACCESS.2021.3099222.

[12] J. Nie, Y. Zhang, Z. He, S. Chen, S. Gong, and W. Zhang, "Deep hierarchical network for automatic modulation classification," *IEEE Access*, vol. 7, pp. 94604–94613, 2019, doi: 10.1109/ACCESS.2019.2928463.

[13] M. Ma, Z. Li, Y. Lin, L. Chen, and S. Wang, "Modulation classification method based on deep learning under non-Gaussian noise," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, Antwerp, Belgium, May 2020, pp. 1–5, doi: 10.1109/VTC2020-Spring48590.2020.9129576.

[14] Y. Zeng, M. Zhang, F. Han, Y. Gong, and J. Zhang, "Spectrum analysis and convolutional neural network for automatic modulation recognition," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 929–932, Jun. 2019, doi: 10.1109/LWC.2019.2900247.

[15] D. Hong, Z. Zhang, and X. Xu, "Automatic modulation classification using recurrent neural networks," in *Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, Chengdu, China, Dec. 2017, pp. 695–700, doi: 10.1109/COMPCOMM.2017.8322633.

[16] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Trans. Cognit. Commun. Netw.*, vol. 4, no. 3, pp. 433–445, Sep. 2018, doi: 10.1109/TCCN.2018.2835460.

[17] Z. Ke and H. Vikalo, "Real-time radio modulation classification with an LSTM auto-encoder," in *Proc. ICASSP - IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Toronto, ON, Canada, Jun. 2021, pp. 4935–4939, doi: 10.1109/ICASSP39728.2021.9414351.

[18] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 168–179, Feb. 2018, doi: 10.1109/JSTSP.2018.2797022.

[19] X. Liu, D. Yang, and A. E. Gamal, "Deep neural network architectures for modulation classification," in *Proc. 51st Asilomar Conf. Signals, Syst., Comput.*, Oct. 2017, pp. 915–919.

[20] J. Xu, C. Luo, G. Parr, and Y. Luo, "A spatiotemporal multi-channel learning framework for automatic modulation recognition," *IEEE Wireless Commun. Lett.*, vol. 9, no. 10, pp. 1629–1632, Oct. 2020, doi: 10.1109/LWC.2020.2999453.

[21] X. Fu, G. Gui, Y. Wang, T. Ohtsuki, B. Adebisi, H. Gacanin, and F. Adachi, "Lightweight automatic modulation classification based on decentralized learning," *IEEE Trans. Cognit. Commun. Netw.*, vol. 8, no. 1, pp. 57–70, Mar. 2022, doi: 10.1109/TCCN.2021.3089178.

[22] T. Huynh-The, C.-H. Hua, Q.-V. Pham, and D.-S. Kim, "MCNet: An efficient CNN architecture for robust automatic modulation classification," *IEEE Commun. Lett.*, vol. 24, no. 4, pp. 811–815, Apr. 2020, doi: 10.1109/LCOMM.2020.2968030.

[23] K. Jung, J. Woo, and S. Mukhopadhyay, "An on-chip accelerator with hybrid machine learning for modulation classification of radio frequency signals," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Denver, CO, USA, Jun. 2022, pp. 487–490, doi: 10.1109/IMS37962.2022.9865378.

[24] J. Wang, C. Wang, H. Zhang, W. Zhang, and D. W. Kwan Ng, "SigMixer: Lightweight automatic modulation classification via multi-layer perceptrons neural network," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Kuala Lumpur, Malaysia, Dec. 2023, pp. 7447–7452, doi: 10.1109/GLOBECOM54140.2023.10437948.

[25] B. Rokh, A. Azarpeyvand, and A. Khanteymoori, "A comprehensive survey on model quantization for deep neural networks in image classification," *ACM Trans. Intell. Syst. Technol.*, vol. 14, no. 6, pp. 1–50, Dec. 2023, doi: 10.1145/3623402.

[26] S. Tridgell, D. Boland, P. H. W. Leong, R. Kastner, and A. Khodamoradi, "Real-time automatic modulation classification using RFSoC," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, New Orleans, LA, USA, May 2020, pp. 82–89, doi: 10.1109/IPDPSW50202.2020.00021.

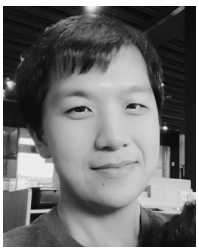
[27] S. Kumar, R. Mahapatra, and A. Singh, "Automatic modulation recognition: An FPGA implementation," *IEEE Commun. Lett.*, vol. 26, no. 9, pp. 2062–2066, Sep. 2022, doi: 10.1109/LCOMM.2022.3184771.

[28] M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 2. Cambridge, MA, USA: MIT Press, 2015, pp. 3123–3131.

[29] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* (Lecture Notes in Computer Science), vol. 9908. Cham, Switzerland: Springer, 2016, pp. 525–542, doi: 10.1007/978-3-319-46493-0_32.

[30] B. Liu, F. Li, X. Wang, B. Zhang, and J. Yan, "Ternary weight networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Rhodes Island, Greece, Jun. 2023, pp. 1–5, doi: 10.1109/ICASSP49357.2023.10094626.

[31] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter, "End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications," *IEEE Access*, vol. 6, pp. 18484–18501, 2018, doi: 10.1109/ACCESS.2018.2818794.
[32] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324, doi: 10.1109/ICCV.2019.00140.
[33] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[34] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.
[35] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8 $\mu$ J/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, Jan. 2019, doi: 10.1109/JSSC.2018.2869150.
[36] J. Zhang, P. Raj, S. Zarar, A. Ambardekar, and S. Garg, "CompAct: On-chip compression of activations for low power systolic array based CNN acceleration," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, p. 47, Oct. 2019, doi: 10.1145/3358178.
[37] X. Wei, C. Hao Yu, P. Zhang, Y. Chen, Y. Wang, H. Hu, Y. Liang, and J. Cong, "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, Jun. 2017, pp. 1–6, doi: 10.1145/3061639.3062207.
[38] S. Hsiao, M. Chen, and C. Tu, "Memory-free low-cost designs of advanced encryption standard using common subexpression elimination for subfunctions in transformations," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 3, pp. 615–626, Mar. 2006, doi: 10.1109/TCSI.2005.859052.
[39] S. Tridgell, M. Kumm, M. Hardieck, D. Boland, D. Moss, P. Zipf, and P. H. W. Leong, "Unrolling ternary neural networks," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, no. 4, pp. 1–23, Dec. 2019, doi: 10.1145/3359983.



**JONGSEOK WOO** received the B.S. and M.S. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2012. He is currently pursuing the Ph.D. degree with Georgia Institute of Technology, Atlanta, GA, USA. He has been a Staff Engineer with the Mobile Display Design Team, Samsung Display Company Ltd. His current research interests include the acceleration of the radio frequency machine learning and the hardware design of the accelerator.

**KUCHUL JUNG** was born in Pohang, South Korea, in 1980. He received the B.S. degree in electronics engineering from Korea University, in 2005. He is currently pursuing the Ph.D. degree in electrical computer engineering with Georgia Institute of Technology. His research interests include radio frequency of machine learning, cognitive radio, and the emerging technologies of 5G wireless communication networks.

**SAIBAL MUKHOPADHYAY** (Fellow, IEEE) received the B.E. degree in electronics and telecommunication engineering from Jadavpur University, Kolkata, India, in 2000, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2006. He was a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, from August 2007 to September 2007. He is currently a Joseph M. Pettit Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. He has authored or coauthored more than 200 articles in refereed journals and conferences and holds five U.S. patents. His research interests include the design of energy efficient, intelligent, and secure systems in nanometer technologies. He was a recipient of the IBM Ph.D. Fellowship Award, from 2004 to 2005; the SRC Technical Excellence Award, in 2005; the SRC Inventor Recognition Award, in 2008; the IBM Faculty Partnership Award, in 2009 and 2010; the National Science Foundation CAREER Award, in 2011; and the Office of Naval Research Young Investigator Award, in 2012.