TinyRadio: Tiny Neural Networks for Fingerprinting Radio Frequency Signals

Mabon Ninan†, Ryan Evans†, Logan Reichling†, Nirnimesh Ghose§, Boyang Wang†
†University of Cincinnati, §University of Nebraska-Lincoln
{*ninanmm, evans2ra, reichlln*}*@mail.uc.edu, nghose@unl.edu, boyang.wang@uc.edu*

***Abstract*—Research studies have shown that deep neural networks can effectively classify Radio Frequency signals for multiple applications, such as modulation classifications and radio fingerprinting, in wireless communications. However, neural networks investigated in the majority of existing studies are complex, which often consist of millions of parameters and consume high memory. This makes it challenging to deploy these complex neural networks on wireless devices. In this paper, we leverage an automatic pruning algorithm, which can automatically decide a customized pruning rate to remove less important filters on each layer in a neural network. We conduct comprehensive evaluations over multiple real-world datasets for both modulation classifications and radio fingerprinting. Our experimental results show that: (1) our automatic pruning can significantly reduce the number of parameters in a neural network for RF classifications (over 98.19% parameter reduction rate); (2) our automatic pruning can outperform pre-defined pruning algorithms in previous research and the ones used in existing industry tools (Xilinx Vitis-AI); (3) our pruning is compatible with other neural network compression technique (e.g., quantization), which can be leveraged to further reduce model size; (4) our pruned neural networks are extremely small (e.g., 0.25 MBs) and can effectively perform RF classifications on embedded devices (Nvidia Jetson) and FGPAs (Xilinx ZCU104 board) with real-time inference (e.g., 0.41 milliseconds per I/Q frame).**

I. Introduction

Recent research show that deep neural networks can outperform traditional methods in classifying Radio Frequency (RF) signals [1]. For instance, neural networks can perform *modulation classification*, which allows a receiver to identify the modulation used for wireless communication. In addition, neural networks can also carry out *radio fingerprinting*, which allows a receiver to authenticate a transmitter based on RF signals alone at the physical layer. These RF classification tasks service fundamental roles in maintaining the performance and security of current as well as future wireless transmissions.

Despite the promising progress in recent literature, one of the major challenges for deep learning RF classifications is that existing neural networks are often complex. For instance, even the lightweight ResNet used for modulation classification carries over 717,000 parameters [2]. These neural networks lead to high memory usage during RF classification on resource-constrained embedded devices.

In this paper, we leverage an automatic pruning algorithm [3], which can remove less important filters in a neural network. Given this algorithm, the pruning rate at each layer is automatically optimized based on filter scores at each layer. The total number of parameters in a neural network can be dramatically reduced with a minimal impact to the results of RF classification. The pruned neural network requires less memory usage and storage, which makes it more efficient for operating RF classification on embedded devices in real time. We conduct comprehensive evaluations over three existing large-scale datasets for two RF classification tasks, including modulation classification and radio fingerprinting. Our main findings are summarized below

- Given a baseline ResNet with 717,250 parameters for modulation classification, we can derive a pruned ResNet with only 12,921 parameters (i.e., 98.19% parameter reduction rate) and this pruned ResNet can still achieve 84.16% accuracy over RadioML2021 dataset [4].
- Given a baseline ResNet with 695,720 parameters for radio fingerprinting, we can obtain a pruned ResNet with only 7,710 parameters (i.e., 98.89% parameter reduction rate) and this pruned ResNet can still authenticate transmitters within 35 I/Q frames over NEU dataset [5].
- Our comparison demonstrates that, given a similar parameter reduction rate, our pruned neural networks achieve higher accuracy than the ones derived from existing studies, including pre-defined pruning [6] and pruning from Xilinx Vitis-AI framework [7]. For instance, given modulation classifications over RadioML2021 dataset, pre-defined pruning obtains a pruned ResNet with 22,519 parameters with 69.71% accuracy, Xilinx Vitis-AI framework derives a pruned ResNet with 9,005 parameters with 41.96%, and our algorithm generates a pruned ResNet with 12,921 parameters with 84.16% accuracy.
- We also demonstrate the effectiveness of our pruned neural networks on Nvidia Jetsons and Xilinx ZCU104 FPGA broads. For instance, given modulation classification, our pruned ResNet can process 6,205 I/Q frames per second and achieves 84.15% accuracy with model size of 0.38 MBs only on Jetson. By further applying quantization, our pruned-then-quantized ResNet can process 2,427 I/Q frames per second and achieves 79.84% accuracy with model size of 0.25 MBs on ZCU104 broad.

Reproducibility. Our source code and datasets can be found at `https://github.com/UCdasec/TinyRadio`

II. Background

System Model. The system model of an RF classification over I/Q frames can be generally described in Fig. 1. There are

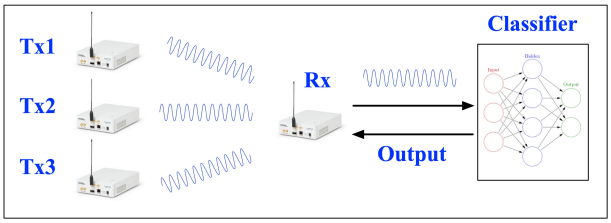

Fig. 1: The (general) system model of RF classification.

multiple transmitters and one receiver. Each transmitter sends RF signals, i.e., sequences of I/Q samples, to the receiver. The receiver performs a classification task over these I/Q samples. Two specific RF classification tasks, including modulation classification and radio fingerprinting, are investigated in this study. Modulation classification aims to decide the modulation used in wireless communication for downstream decoding. Radio fingerprinting aims to infer which transmitter it is for physical-layer authentication.

Given a sequence of I/Q samples, the receiver first pre-processes it by dividing the sequence into multiple non-overlapping I/Q frames, where each frame consists of a fixed number of consecutive I/Q samples. An RF classification task based on neural networks requires two phases, including a training phase and a test phase. In the training phase, a neural network is trained based on labeled I/Q frames from multiple classes (e..g, transmitters or modulations). In the test phase, the receiver performs the RF classification over unlabeled I/Q frames by leveraging the trained neural network. The neural network can be trained in advance. The trained neural network needs to perform RF classification in real time on the receiver.

Metrics. *Accuracy* is a primary metric to measure the performance of neural network in a RF classification. Given n I/Q frames, if m I/Q frames are predicted correctly by a trained classifier, the accuracy is calculated as m/n. A higher accuracy indicates a stronger capability in RF classification.

To measure the complexity of a neural network, we adopt two metrics, including (1) the *number of parameters* and (2) the number of *Floating Point Operations*. A FLOP (Floating Point Operation) is an addition, subtraction, division, multiplication, or any other operation involving a floating point value. *Reduction Rate* is calculated as the ratio between the number of parameters/FLOPs removed after the pruning and the number of parameters/FLOPs in a baseline network before pruning.

III. Our Proposed Design

A. Neural Network Pruning

Neural Network Pruning [8], [9], or *pruning* for short, reduces the size of a neural network by pruning less significant parameters/weights. Structured pruning removes parameters in groups (e.g., channels or filters), which is more effective than unstructured pruning. We focus on structure pruning over filters in a neural network in this paper.

Typically, a *score algorithm* is used to measure the importance of a filter. We use l_2 norm [10] as the score algorithm in this paper. *Pruning rate* p of a layer is defined as the ratio between the number of filters that are pruned/removed and the total number of filters (before pruning) at this layer.

Given a set of filters $\mathbf{W}^i = \{F_{i,1}, ..., F_{i,N_i}\}$ at the i-th layer and an integer M_i, a pruning algorithm keeps a subset $\mathbf{W}^{i*}$ of M_i filters from all the N_i filters such that the M_i filters have the maximum sum of importance. In other words, the pruning algorithm finds a subset $\mathbf{W}^{i*}$ such that

$$\underset{\mathbf{W}^{i*} \subset \mathbf{W}^i}{\arg\max} = \sum_{F_{i,j} \in \mathbf{W}^{i*}} S(F_{i,j}) \quad (1)$$

where $|\mathbf{W}^{i*}| = M_i$ and $S(\cdot)$ is a score algorithm. Filters that are in set $\mathbf{W}^i$ but not in subset $\mathbf{W}^{i*}$ are removed. The pruning rate of this layer is defined as $1 - \frac{M_i}{N_i}$.

General Pruning Process. Given a score algorithm and a trained neural network, a pruning process in one iteration consists of includes two steps: (1) removing less important filters based on a score algorithm and a pruning rate, where the pruning rate is either pre-defined or automatically decided on-the-fly; (2) fine-tuning a pruned neural network to regain accuracy in predictions.

B. Our Pruning for RF Classification

In this paper, we leverage a recent pruning algorithm proposed in [3]. This algorithm, named MiniDrop, automatically decides a customized pruning rate for each layer based on the filter scores. Specifically, after assigning scores for each filter, all these filters at every layer are sorted based on filter scores in a descending order. After sorting, the filter index with the *minimal absolute value of the discrete derivative* over all the sorted filter scores is marked, and then all the filters after this index is considered less important and are pruned. Each layer ends up with a customized pruning rate based on filter scores at the corresponding layer. A pruning parameter h ($h \geq 1$) in MiniDrop defines the granularity of the pruning, where a lower value of h indicates a more aggressive pruning. More details of this pruning algorithm can be found in [3].

IV. Evaluation

We evaluate the effectiveness of the automatic pruning algorithm in the context of RF classifications, including modulation classification and radio fingerprinting, over three existing large-scale datasets in this section.

A. Baseline Model Architectures

We choose a lightweight ResNet model as our baseline architecture for both radio fingerprinting and modulation classification. Specifically, this ResNet consists of 5 ResNet stacks, where each stack includes 2 convolutional layers (using ReLU as the activation function) followed by 1 max pooling layer. One dense layer and a softmax layer are attached to the end of the last ResNet stack for classification. This ResNet was developed for modulation classification in [11] and is also included as a tutorial for modulation classification on FPGAs in Xlinix Vitis AI framework. The hyperparameters of the baseline architecture are summarized in Table I.

TABLE I: Hyperparameters of the Lightweight ResNet

Conv_1 BatchNorm_1 Activation_1	filters: 16; kernel: (5, 1); activation: None; Applied after Conv_1 Relu
ResNet_Block_1 ResNet_Block_2 ResNet_Block_3 ResNet_Block_4 ResNet_Block_5	in filters: 16; out filters: 32; kernel: (5, 1); Relu in filters: 32; out filters: 64; kernel: (5, 1); Relu in filters: 64; out filters: 128; kernel: (5, 1); Relu in filters: 128; out filters: 256; kernel: (5, 1); Relu in filters: 256; out filters: *No. of classes*; kernel: (5, 1); Relu
Flatten Dropout Dense Softmax	Applied after ResNet_Block_5 rate: 0.5 No. of neurons: *No. of classes*; Relu Applied after Dense layer

B. Datasets

Dataset for Modulation Classification. We leverage one existing large-scale dataset, named RadioML 2021 [4], in the evaluation associated with modulation classification. This dataset consists of synthetic simulated channel effects from 27 different types of digital and analog modulations. Specifically, it includes RF signals from OOK, 4ASK, 8ASK, BPSK, QPSK, 8PSK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, AM-SSB-WC, AM-SSB-SC, AM-DSB-WC, AM-DSB-SC, FM, GMSK, OQPSK, BFSK, 4FSK, 8FSK. RF signals of each modulation are meticulously crafted to reflect realistic RF signal conditions, including various levels of noise, fading, and interference. This dataset and its previous versions (including RadioML 2018 and 2016) have been widely utilized by studies in deep learning modulation classification [11], [12]. The details and description of the dataset can be found in [11]. The I/Q frames in this dataset are pre-processed already and each frame consists of 1,024 I/Q samples. How the pre-processing was extracted is not explicitly mentioned in the dataset. The entire dataset consists of 2,875,392 I/Q frames and is balanced across all the modulation classes. The dataset consists of I/Q frames at 26 SNR (Signal-to-Noise Ratio) levels, ranging from -20 dB to 30 dB with increments of 2 dB. Similar to previous studies [12], [2], we only leverage I/Q frames with SNRs that are higher than or equal to 6 dB. This results in a dataset of 1,437,696 I/Q frames (i.e., 27 modulations and 53,248 frames per modulation). All the samples in frames are stored in `int8` format. We use 80% of frames for training, 10% of frames for validation, and 10% for testing in our experiments.

Datasets for Radio Fingerprinting. We leverage two datasets, referred to as NEU dataset and HackRF-10 dataset, collected respectively from two previous studies [13], [14]. NEU dataset consists of RF transmissions acquired from a lab testbed established by Al-Shawabka in [13]. The testbed consists of 21 USRPs, where 20 USRPs (12 NI N210 and 8 NI X310) serve as transmitters and 1 USRP serves as a receiver. We specifically utilize a subset of this dataset with the setup "Setup A-In-the-Wild, Different Antennas" described in their dataset document [5]. Samples are streamed at 2.432 GHz with a sampling rate of 20 million samples per second and BPSK 1/2 was used as modulation. I/Q samples were collected after WiFi Frame Equalizer in GNU Radio. RF signals of each transmitter consist of 10 transmissions and each transmission lasts for 30 seconds each day. The data collection lasted over 10 days. We only leverage RF signals from one day in our experiment. All the samples are stored in `floating-point` format.

HackRF-10 dataset compromises RF signals collected from a lab testbed with 10 HackRF Ones serving as transmitters and 1 HackRF One serving as a receiver [14]. Each HackRF One is equipped with 1 ANT500 antenna. I/Q samples were collected at 2.45 GHz center frequency with 2 MHz bandwidth and 2 MHz sampling rate running BPSK 1/2 modulation. The data collection run for two days with 3 transmissions (30 seconds per transmission) from every transmitter on each day. Samples were recorded before FFT, after FFT, and after WiFi Frame Equalizer on the receiver side. We only utilize samples after WiFi Frame Equalizer from Day 1 in our experiments. All the samples are stored in `floating-point` format.

Pre-processing for I/Q frames in Radio Fingerprinting. Studies [13], [14] addressing radio fingerprinting often apply sliding windows as a way of pre-processing raw I/Q samples and preparing inputs (i.e., I/Q frames) for neural networks. Given the entire sequence of I/Q samples from a transmission, which consists of thousands (or millions) of I/Q samples, a sliding window with a length L starts from the beginning of the entire sequence and extracts the L I/Q samples within the window to form one I/Q frame. Next, the window slides towards the end of the sequence with a stride s and extracts the next I/Q frame with L I/Q samples. The process repeats until a certain number of I/Q frames has been prepared or the window reaches the end of the entire sequence. Typically, window stride s should be greater than window size L to avoid overlaps across I/Q frames per previous studies [14]. Otherwise, a neural network tends to learn/remember the content of I/Q samples rather than RF shifts caused by transmitter fingerprints.

We choose stride $s = 864$ and window length $L = 864$ in the pre-processing for both NEU dataset and HackRF-10 dataset. From each dataset, we randomly select and form a set of 40,000 I/Q frames per device, i.e., 800,000 I/Q frames for NEU dataset and 400,000 I/Q frames for HackRF-10 dataset. We use 80% for training, 10% for validation, and 10% for testing.

C. Experiment Setting: Software and Hardware

For all the experiments requiring GPUs, we run them on a server with Ubuntu 22.04, Intel i9 CPU, 128 GB memory, and a NVIDIA GeForce RTX Titan GPU. For each neural network, we train it for 150 epochs and apply an early stop with 10 epochs (i.e., the training stops before it reaches 150 epochs if the validation loss has not been changed for the last 10 epochs). We use a learning rate of 0.001 with `RMSprop` as the optimizer. We use TensorFlow (version 2.4.1) and Python (version 3.9.19) to implement neural networks and associated functions. For the fine-tuning phase involved in pruning, we use the same set of training traces and fine-tune each neural network for 150 epochs with an early stop of 10 epochs.

TABLE II: Performance of Baseline Neural Networks for RF Classifications

	Dataset	Training Time (sec) in Total	Training Time (sec) per Epoch	No. of Parameters	FLOPs (million)	Model Size (MBs)	Accuracy	Random Guess
Modulation	RadioML	3,591	104.41	717,250	241.75	5.95	97.76%	3.7%
Radio	NEU	2,065	51.04	695,720	203.78	5.78	64.74%	5%
	HackRF-10	1,830	26.09	674,760	203.60	5.61	64.28%	10%

TABLE III: Performance of Pruned Neural Networks for Modulation Classification (RadioML 2021, Random Guess: 3.7%, RR: Reduction Rate, $\perp$ indicates failing to obtain a pruned neural network)

Pruning Param. h	No. of Parameter	Parameters RR	FLOPS (million)	FLOP RR	Accuracy	Pruning Time (sec)	Fine-Tuning Time (sec) per Epoch
Baseline	717,250	0%	241.75	0%	**97.76%**	NA	NA
64	712,685	0.63%	233.24	3.52%	94.20%	0.157	98.95
32	681,405	4.99%	211.40	12.55%	97.69%	0.156	88.01
16	461,565	35.64%	147.33	39.05%	93.92%	0.159	71.76
8	120,180	83.24%	58.98	75.60%	93.90%	0.158	51.40
4	30,009	95.81%	22.40	90.73%	87.46%	0.157	29.45
2	**12,921**	**98.19%**	**11.43**	**95.27%**	84.16%	0.159	**24.69**
1	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

TABLE IV: Performance of Pruned Neural Networks for Radio Fingerprinting (NEU dataset, Random Guess: 5%, RR: Reduction Rate, $\perp$ indicates failing to obtain a pruned neural network)

Pruning Param. h	Number of Parameter	Parameters RR	FLOPS (million)	FLOP RR	Accuracy	Pruning Time (sec)	Fine-Tuning Time (sec) per Epoch
Baseline	695,720	0%	203.78	0%	64.28%	NA	NA
64	685,825	1.42%	192.75	5.41%	**64.43%**	0.242	45.95
32	489,405	29.65%	147.49	27.62%	55.89%	0.230	41.88
16	266,360	61.71%	100.32	50.76 %	46.30%	0.229	36.08
8	58,735	91.55%	25.52	87.47%	31.55%	0.228	24.22
4	26,440	96.19%	11.25	94.48%	25.23%	0.223	15.49
2	**7,710**	**98.89%**	**2.37**	**98.83%**	17.04%	0.224	**10.53**
1	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

TABLE V: Performance of Pruned Neural Networks for Radio Fingerprinting (HackRF-10 dataset, Random Guess: 10%, RR: Reduction Rate, $\perp$ indicates failing to obtain a pruned neural network)

Pruning Param. h	Number of Parameter	Parameters RR	FLOPS (million)	FLOP RR	Accuracy	Pruning Time (sec)	Fine-Tuning Time (sec) per Epoch
Baseline	674,760	0%	203.60	0%	**64.74%**	NA	NA
64	670,095	0.69%	203.57	0.014%	61.95%	0.218	24.81
32	610,730	9.49%	193.48	4.97%	63.00%	0.215	22.67
16	382,740	43.28%	123.75	39.22 %	56.30%	0.217	19.12
8	102,225	84.85%	48.81	76.03%	65.27%	0.219	14.12
4	13,760	97.96%	10.36	94.91%	62.14%	0.225	7.76
2	**5,305**	**99.21%**	**2.19**	**98.92%**	54.83%	0.219	**7.06**
1	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

D. Experiments

Experiment 1: Performance of Baseline Neural Networks. We first report the performance of the baseline neural network in both modulation classification and radio fingerprinting. As presented in Table. II, the baseline neural network, i.e., the lightweight ResNet, derives high accuracy but requires a large number of parameters for each RF classification. Specifically, the baseline neural network achieves 97.76% accuracy across 27 classes in modulation classification over the RadioML 2021 dataset ($\geq$ 6 dB only), which is consistent with the accuracy reported in previous studies using the same or similar architecture [12]. Similarly, the accuracy obtained in radio fingerprinting over the two datasets is over 64%, which is much higher than random guess and is also consistent with results listed in recent studies [13], [14]. On the other hand, the baseline neural network requires around 700,000 parameters across the three datasets and two RF classifications we examine.

Experiment 2: Performance of Pruned Neural Networks. We evaluate the performance of pruned neural networks obtained by the pruning algorithm (MiniDrop). Specifically, we show the trade-off between accuracy and the number of parameters for both RF classifications when we adjust the pruning parameter h in the pruning algorithm. Given a baseline neural network, we apply MiniDrop by selecting the pruning parameter h as a value from the set $\{64, 32, 16, 8, 4, 2, 1\}$ each time, where a smaller value generally indicates the pruning is more aggressive and a greater number of parameters in a neural network will be removed.

As illustrated in Table III, we have two major observations. First, the pruning can effectively reduce the number of parameters in a neural network while still maintaining a relatively high accuracy. Second, when we reduce the pruning parameter h, both the number of parameters and accuracy decrease. For

instance, given $h = 2$, we can obtain a pruned neural network with only 12,921 parameters (i.e., 98.19% reduction) but can still achieve 84.16% accuracy (only 13.6% accuracy drop) compared to the baseline network. We also observe that the fine-tuning time decreases significantly given a smaller pruning parameter h as a smaller neural network is used in fine-tuning. On the other hand, the pruning time (the time that is needed to obtain filter scores and prune less important filters) remains the same regardless of the selection of the pruning parameter. It is worth mentioning that when we reduce h to 1, some layers no longer contain the minimal number of filters (i.e., 1) and we fail to derive a pruned network. In other words, the pruned network we obtained from $h = 2$ is the smallest neural network we can derive from the baseline over RadioML 2021 dataset.

In Table IV and V, we obtain similar observations from radio fingerprinting over NEU dataset and HackRF-10 dataset. For instance, when $h = 2$, we can derive the smallest pruned neural network with only 5,305 parameters (99.21% parameter reduction rate) over HackRF-10 dataset while still achieving 55.83% accuracy (only 9.91% accuracy drop) given our pruning algorithm. For NEU dataset, we can also achieve a 98.89% parameter reduction rate, which reduces the total number of parameters to only 7,710. As a tradeoff, the accuracy drops to 17.04% (over 46% accuracy drop). While the accuracy drop seems to be significant, we would like to emphasize that the pruned neural network can still authenticate transmitters correctly and efficiently.

More specifically, we report the *device rank* of pruned neural networks over NEU dataset, where the confidence scores of all the transmitters over multiple test I/Q frames are aggregated and ranked. A device rank of 1 given N I/Q frames indicates a classifier can rank the correct transmitter as the top candidate within N I/Q frames. A recent work suggests that device rank is a more reliable metric compared to accuracy in the context of radio fingerprinting [14]. We can observe in Fig. 2 that, even though the accuracy is only 17.04%, our pruned neural network can achieve device rank of 1 given 35 I/Q frames (i.e., the device rank converges to 1 within 35 I/Q frames). Given a sampling rate of 20 million samples per second and 864 samples per trace in NEU dataset, it suggests that our pruned neural network can authenticate a transmitter correctly with only 1.51 milliseconds (i.e., $\frac{35\times 864}{20\times 10^6}$ seconds).

Experiment 3: Comparison between Single-Iteration Pruning and Multi-Iteration Pruning. In this experiment, we compare the performance of pruned neural networks derived from single-iteration pruning and multi-iteration pruning in RF classifications. Specifically, in addition to running the pruning with a single iteration, we also run the pruning for 2, 3, or 4 iterations, where the input neural network for one iteration is the output of the previous iteration. We highlight the results of modulation classification over the RadioML dataset in Table VI.

We have two main findings. First, given the same pruning parameter h, if we apply a greater number of pruning iterations, it is feasible to further reduce the number of parameters in a neural network. However, the accuracy will drop significantly

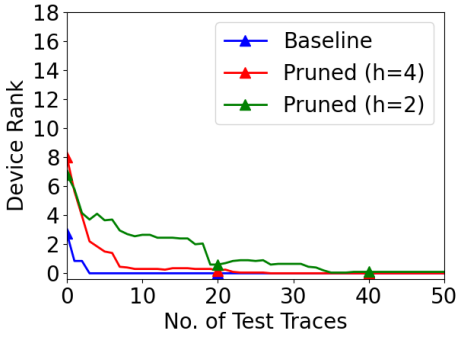

Fig. 2: Device rank over NEU dataset given our pruned neural networks ($h = 2$, No. of parameters, 7,710; accuracy 17.04%; $h = 4$, No. of parameters, 26,440, accuracy: 25.23%)

TABLE VI: Comparison between Single-Iteration Pruning and Multi-Iteration Pruning for Modulation Classification (RadioML 2021).

No. of Iterations	Pruning Param. h	No. of Parameter	Pruning & Fine-Tuning Time (sec)	Accuracy
1	2	**12,921**	**650**	**84.16%**
2	2	$\perp$	$\perp$	$\perp$
1	4	30,009	980	87.46%
2	4	16,242	1,350	65.86%
3	4	$\perp$	$\perp$	$\perp$
3	8	29,073	2,150	82.83%
4	8	18,126	2,750	55.27%

and the overall pruning time will increase. At some point, the pruning cannot be applied further and no pruned neural networks will be obtained. Second, we find that applying pruning gradually through multiple iterations with a less aggressive pruning parameter (i.e., a greater value of h) *derives* a greater pruned neural network with lower accuracy than the one obtained from pruning with a single iteration with a more aggressive pruning parameter. For instance, given a pruning parameter $h = 2$ with a single iteration, we can obtain a neural network with only 12,921 parameters and 84.16% accuracy for modulation classification. On the other hand, if we prune with 4 iterations but using pruning parameter $h = 8$, we generate a neural network that still has 18,126 parameters but with an accuracy of 55.27%. *In other words, given MiniDrop as the pruning algorithm, one should apply single-iteration pruning with an aggressive pruning parameter rather than running multi-iteration pruning with a conservative pruning parameter in the context of RF classification.*

Experiment 4: Performance of Pruned Neural Networks on Embedded Devices. We demonstrate that our pruned neural networks can still run efficiently and effectively on embedded devices, including Nvidia Jetson and FPGAs (as shown in Fig. 3). In addition, our evaluations on FPGAs also show that our pruning is compatible with other neural network compression techniques, such as quantization, and can be integrated into existing industry AI development frameworks (e.g., Xilinx Vitis AI).

For the evaluations associated with Nvidia Jetson, we train

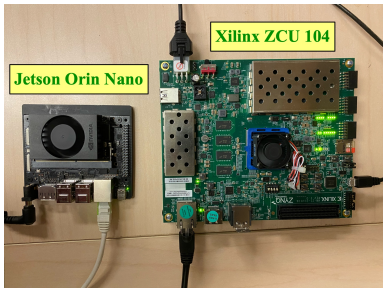

Fig. 3: Embedded Systems Used in Our Evaluation.

and prune a neural network in advance on our server with GPU. Next, we run testing on a NVIDIA Jetson Orin Nano. Jetson Orin Nano consists of a NVIDIA Ampere GPU, a 6-core ARM CPU, and 8GB memory. We still use the same number of I/Q frames for testing on Jetson.

For the evaluations associated with FPGAs, we perform training and pruning of a neural network on our server with GPU. Next, we run quantization and testing of a neural network on an AMD/Xilinx Zynq UltraScale+ MPSoC ZCU104 Evaluation Board. It is equipped with a quad-core ARM Cortex-A53 applications processor, dual-core Cortex-R5 real-time processor, Mali-400 MP2 graphics processing unit, and 16nm FinFET+ programmable logic.

Given a pruned neural network, we leverage Xlinix Vitis-AI framework (version 3.5) to further quantize a neural network. Specifically, we use *power of 2 scale quantization* (`pos2`), which is a post-training quantization. This quantization transforms weights in a neural network from floating points (`FP32`) to integers (`int8`) except the ones on the last softmax layer. Once completing quantization, we use Xlinix Vitis-AI framework to compile the compressed neural network and deploy it to the ZCU104 board. During the compilation, the Vitis-AI framework automatically splits the operations of a compressed model into two components, where the majority of layers are implemented through DPUs (Deep-Learning Processing Units) on the FPGA, and the last softmax layer is run on CPU. A DPU is a programmable engine optimized for neural networks and consists of a set of parameterizable IP cores pre-implemented on the hardware without the need for place and route. To perform RF classification with a compressed neural network from Vitis-AI, the I/Q samples are also transformed to `int8` format if the original format is `FP32`.

As shown in Table VII, we can observe that the pruned neural network can still run effectively on embedded systems with similar accuracy as on GPUs. Specifically, for modulation classification, the pruned neural network on Jetson carries the same number of parameters and almost the same accuracy (84.15%) as the pruned one on GPU. Compared to processing over 125 thousand frames per second on a NVIDIA Titan, Jetson can process 6,205 frames per second, which is still highly efficient. The testing performance decrease is expected for embedded systems. We have similar observations about radio fingerprinting on Jetson as presented in Table VIII.

On the FPGA, there is a slight accuracy drop (4.29%) for

TABLE VII: Performance of Pruned Neural Networks on Embedded Systems for Modulation Classification (RadioML, $h = 2$, Single-Iteration Pruning).

	No. of Parameters	Testing (No. of frames per sec)	Model Size (MBs)	Accuracy
Titan	12,921	169,067	0.377	84.13%
Jetson	12,921	6,205	0.377	84.15%
ZCU104	12,505	2,427	0.246	79.84%

TABLE VIII: Performance of Pruned Neural Networks on Embedded Systems for Radio Fingerprinting (NEU Dataset, $h = 2$, Single-Iteration Pruning).

	No. of Parameters	Testing (No. of frames per sec)	Model Size (MBs)	Accuracy
Titan	7,710	4,837	0.317	17.29%
Jetson	7,710	607	0.317	16.68%
ZCU104	7,481	2,584	0.230	5%

modulation classification due to the additional quantization step applied after pruning. On the other hand, the size of the pruned neural network on the FPGA is further reduced due to quantization (from 0.377 MBs to 0.246 MBs). For radio fingerprinting, unfortunately, the accuracy drops to random guess (5%) after further applying quantization with Vitis-AI on our pruned neural network, which makes it unable to authenticate transmitters correctly. This is likely because the samples of NEU dataset are in `FP32` format and transforming samples from `FP32` to `int8` dramatically affects accuracy in addition to the accuracy drop due to quantization.

Experiment 5: Comparison between Our Work and Previous Studies on Pruning. We compare the performance of our pruned neural networks with two methods, including pre-defined pruning and the pruning offered by current industry tools (e.g., Vitis-AI). Pre-defined pruning is a common pruning method that removes a certain percentage of less important filters on every layer in a neural network given a pre-defined pruning rate. For instance, given a pruning rate of 0.6, pre-defined pruning will rank the filters at each layer based on scores (e.g. l_2 norm) and remove 60% of filters that are less important (i.e., with lower scores) on each layer.

A recent paper [6] first investigated pre-defined pruning in the context of radio fingerprinting. Specifically, the authors apply the pruning through multiple rounds, where Alternating Direction Method of Multipliers is used to identify the more important filters given a pre-defined pruning ratio at each layer and masked retraining is utilized to regain accuracy in each round. Although the source code and pruning parameters of this previous study are publicly available, they are highly integrated with their own neural network architecture and private datasets, which prevents us from reproducing results or extending it to other neural networks and datasets (including ours). To still perform comparisons (at least to some fair degree), we implement the pre-defined pruning method by ourselves using l_2 norm as the scoring algorithm and apply the same pruning rate to each layer.

For Vitis AI, we leverage *coarse-grained pruning* (also referred to as *channel pruning*) to prune a baseline neural

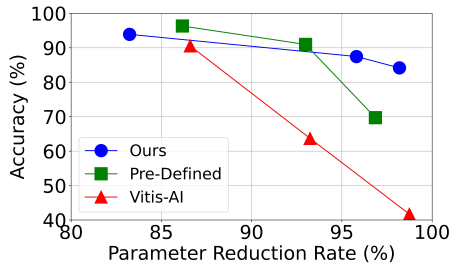

Fig. 4: Comparison among ours, pre-defined pruning, and Vitis-AI's pruning on modulation classification over RadioML dataset (frames with SNR $\geq$ 6dB).

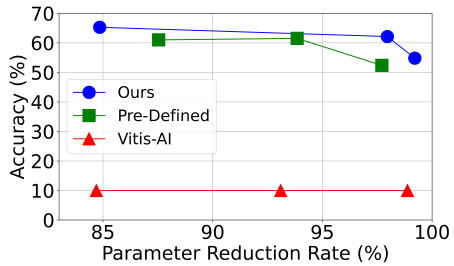

Fig. 5: Comparison among ours, pre-defined pruning, and Vitis-AI's pruning on radio fingerprinting over HackRF dataset

network. In essence, it is a pre-defined pruning in which the pruning process prunes a baseline neural network given a global pruning rate α. To perform the pruning with Vitis AI, we choose the global pruning rate α, set the number of pruning iterations as 1, pass a baseline neural network, and provide a subset of training data to the Vitis Analyzer. The Vitis Analyzer leverages the subset of training data to automatically generate a pruning strategy for the baseline neural network. In the case of modulation classification, we provide 500,000 training frames to the Vitis Analyzer. In the case of Radio Fingerprinting, we provide 500,000 training frames for the NEU dataset and 320,000 training frames for HackerRF dataset.

Given the baseline neural network from modulation classification, we compare the accuracy and parameter reduction rate of pruned neural networks obtained from the three methods, including ours, pre-defined pruning, and Vitis AI's pruning, in Fig. 4. For our method, we report the results from $h = \{8, 4, 2\}$, where $h = 2$ is the most aggressive pruning parameter we can apply over this dataset. For pre-defined pruning, we report results from $p = \{0.5, 0.6, 0.7\}$, where $p = 0.7$ is the most aggressive pruning parameter we can apply over this dataset. For Vitis AI's pruning, we present results from $\alpha = \{0.79, 0.89, 0.99\}$, where $\alpha = 0.99$ is the most aggressive pruning parameter we can apply over this dataset.

We find that pre-defined pruning can achieve a higher accuracy with a lower parameter reduction rate initially than ours, but is eventually outperformed by our automatic pruning. While Vitis AI's pruning algorithm can obtain an even smaller neural network with only 9,005 parameters (v.s., 12,921 parameters from the smallest by ours), it takes a significant tradeoff in accuracy (i.e., 41.96% accuracy), which is much lower than 84.16% accuracy from the smallest pruned neural network obtained by our automatic pruning.

For radio fingerprinting over the HackRF dataset, we have similar observations regarding the comparison between our pruning and pre-defined pruning as shown in Fig 5. On the other hand, Vitis-AI pruning results in a pruned neural network with only random guess.

V. Related Work

Modulation Classification. Several recent studies [12], [15], [2], [16], [17] in modulation classification focus on optimize the size and performance of neural networks by applying quantization and deploying compressed neural networks on FPGAs. Specifically, Tridgell et al. [12] utilize ternary-weight neural networks for modulation classification, where weights are {1, 0, -1} only (i.e., each weight has only 4 bits). They show that a compressed VGG10 with ternary weights can achieve up to 82% accuracy given traces of 30 dB over RadioML 2018 dataset and demonstrate the performance of the compressed neural network on Xilinx ZCU111. The quantized VGG10 consists of 490,000 parameters and is about 123 Kbs. Woo et al. [17] also quantize weights to ternary weights in a lightweight neural network (MobileNetV3) and implement compressed neural networks on Xilinx ZCU102 evaluation board. Their quantized MobileNetV3 includes 0.4 million parameters with a size of 0.8 million bits in total. Both studies also leverage Common Subexpression Elimination to improve the efficiency of hardware designs. Jentzsch et al. [2] leverage FINN, Xilinx's open-source compiler framework, to quantize a neural network for modulation classification. The authors show that they are able to quantize the weights and activations of a VGG10 to the ones with 4 bits using FINN and still achieve 94% accuracy given traces of 30 dB in RadioML 2018 dataset. The quantized VGG10 consists of 72,000 parameters. Den Boer et al. [15] leverage High-Level Synthesis to optimize the implementation of neural networks on FPGAs for modulation classification. Their model carries 13,840 parameters and can achieve 71.49% accuracy at 30dB SNR. *These studies mainly focus on quantization. On the other hand, our method can significantly reduce the number of parameters in a neural network, which can complement these quantization techniques and further optimize the size of neural networks.*

Several recent work [18], [19], [20] also investigate neural network pruning in the context of modulation classification. For instance, Chen et al. [20] applies single-short pre-defined structured pruning, which measures the score of channels based on the cosine similarity of weights on convolutional layers and batch normalization layers. *However, these work are all based on pre-defined pruning, which applies the same pruning rate to each layer and is less effective than our automatic pruning. Besides, none of them has shown that the pruning is compatible with quantization as ours*

Besides reducing the size of neural networks, improving robustness of modulation classification, especially against unseen data with domain shifts is critical. For instance, Jung et al. [21] leverage Short-Time Fourier Transform to transform data from

the time domain to frequency-time domain to improve the accuracy of a neural network for modulation classification. Liu et al. [22] transform IQ samples into three representations, including phase, angular, and frequency, and pass data in representations to a single lightweight neural network to enhance the robustness of modulation classification against unseen data. Nasr et al. [23] examine domain shifts caused by adversarial examples. More comprehensive surveys on modulation classification can be found in [24].

Radio Fingerprinting. Chen et al. [6] first investigate predefined pruning in the context of radio fingerprinting. Specifically, the authors apply the pruning through multiple rounds, where Alternating Direction Method of Multipliers is used to identify the more important filters given a pre-defined pruning ratio at each layer and masked retraining is utilized to regain accuracy in each round. The authors demonstrate the effectiveness of their pruning over multiple edge devices, including smart phones, Nvidia Jetsons, and Xilinx ZCU104 FPGA boards, over two private datasets. The pruning ratio of each layer is defined manually in their evaluation. Their results indicate that the proposed pruning can achieve a pruning rate of 27.2 over ResNet50-1D (equivalent to 96.32% parameter reduction rate in our study), where the total number of parameters in the neural network is reduced from 16.06 million to 0.74 million. Additional hardware optimizations on FPGAs (e.g., parallel processing elements) are also explored to further improve the performance.

The majority of existing studies [13], [14], [25], [26], [27], [28] in radio fingerprinting investigates domain shifts caused by different days, locations, receivers, and adversarial attacks as well as methods that can overcome these domain shifts.

VI. Conclusion

We apply automatic pruning to optimize the size of neural networks with a minimal impact to the results of RF classification tasks. We also demonstrate that it is feasible to run our pruned neural networks on embedded devices with extremely small storage and real-time inference time.

References

[1] T. Erpek, T. J. O'Shea, Y. E. Sagduyu, Y. Shi, and T. C. Clancy, "Deep Learning for Wireless Communications," https://arxiv.org/pdf/2005.06068.

[2] F. Jentzsch, Y. Unuroglu, A. Pappalardo, M. Blott, and M. Platzner, "RadioML Meets FINN: Enabling Future RF Applications with FPGA Streaming Architectures," *IEEE Micro*, vol. 42, no. 6, 2022.

[3] H. Li, M. Ninan, B. Wang, and J. M. Emmert, "Tinypower: Side-channel attacks with tiny neural networks," in *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2024.

[4] [Online]. Available: https://opendata.deepsig.io/datasets/2021.07/RADIOML_2021_07_INT8.tar.gz

[5] [Online]. Available: https://wiot.northeastern.edu/wp-content/uploads/2020/07/dataset_release.pdf

[6] T. Jian, Y. Gong, Z. Zhan, R. Shi, N. Soltani, Z. Wang, J. Dy, K. Chowdhury, Y. Wang, and S. Ioannidis, "Radio frequency fingerprinting on the edge," *IEEE Transactions on Mobile Computing*, 2022.

[7] [Online]. Available: https://github.com/Xilinx/Vitis-AI

[8] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the State of Neural Network Pruning?" in *Proc. of the 3rd MLSys Conference*, 2020.

[9] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and Quantization for Deep Neural Network Acceleration: A Survey," in *Neurocomputing*, vol. 461. Elsevier, 2021.

[10] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[11] T. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING*, vol. 12, no. 1, 2018.

[12] S. Tridgell, D. Boland, P. H. Leong, R. Kastner, A. Khodamoradi, and Siddhartha, "Real-Time Automatic Modulation Classification using RF-SoC," in *Proc. of IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW 2020)*, 2020.

[13] A. Al-Shawabka, F. Restuccia, S. D'Oro, T. Jian, B. C. Rendon, N. Soltani, J. Dy, S. Ioannidis, K. Chowdhury, and T. Melodia, "Exposing the Fingerprint: Dissecting the Impact of the Wireless Channel on Radio Fingerprinting," in *Proc. of IEEE INFOCOM'20*, 2020.

[14] H. Li, K. Gupta, C. Wang, N. Ghose, and B. Wang, "RadioNet: Robust Deep-Learning Based Radio Fingerprinting," in *Proc. of IEEE Conference on Communication and Network Security (CNS 2022)*, 2022.

[15] H. den Boer, R. W. D. Muller, S. Wong, and V. Voogt, "FPGA-based Deep Learning Accelerator for RF Applications," in *Proc. of Milcom'21*, 2021.

[16] J. Rosa, D. Granhao, G. Carvalho, T. Goncalves, M. Figueiredo, L. C. Bento, N. Paulino, and L. M. Pessoa, "BACALHAUNET: A Tiny CNN for Lightning-Fast Modulation Classification," *ITU Journal on Future and Evoling Technologies*, vol. 3, no. 2, 2022.

[17] J. Woo, K. Jung, and S. Mukhopadhyay, "Efficient Hardware Design of DNN for RF Signal Modulation Recognition Employing Ternary Weights," *IEEE Access*, vol. 12, 2024.

[18] Y. Lin, Y. Tu, and Z. Dou, "An Improved Neural Network Pruning Technology for Automatic Modulation Classification in Edge Devices," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, 2020.

[19] Y. Wang, J. Yang, M. Liu, and G. Gui, "LightAMC: Lightweight Automatic Modulation Classification via Deep Learning and Compressive Sensing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, 2020.

[20] Z. Chen, Z. Wang, X. Gao, J. Zhou, D. Xu, S. Zheng, Q. Xuan, and X. Yang, "Channel Pruning MEthod for Signal Modulation Recognition Deep Learning Models," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 2, 2024.

[21] K. Jung, J. Woo, and S. Mukhopadhyay, "On-Chip Acceleration of RF Signal Modulation Classification with Short-Time Fourier Transform and Convolutional Neural Network," *IEEE Access*, vol. 11, 2023.

[22] D. Liu, K. Ergun, and T. S. Rosing, "Torwards A Robust and Efficient Classifier for Real World Radio Signal Modulation Classification," in *Proc. of 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'23)*, 2023.

[23] M. Nasr, P. Araujo-Filho, G. Kaddoum, and A. Mourad, "Projected Natural Gradient Method: Unveiling Low-Power Perturbation Vulnerabilities in Deep Learning-Based Automatic Modulation Classification," *IEEE Internet of Things Journal*, 2024.

[24] B. Jdid, K. Hassan, I. Dayoub, W. H. Lim, and M. Mokayef, "Machine Learning Based Automatic Modulation Recognition for Wireless Communications: A Comprehensive Survey," *IEEE Access*, 2021.

[25] S. AlHazbi, S. Sciancalepore, and G. Oligeri, "The Day-After-Tomorrow: On the Performance of Radio Fingerprinting over Time," in *Annual Computer Security Applications Conference (ACSAC'23)*, 2023.

[26] G. Shen, J. Zhang, A. Marshall, R. Woods, J. Cavallaro, and L. Chen, "Towards Receiver-Agnostic and Collaborative Radio Frenqeucy Fingerprint Identification," *IEEE Transactions on Mobile Computing*, 2024.

[27] T. Zhao, X. Wang, J. Zhang, and S. Mao, "Explanation-Guided Backdoor Attacks on Model-Agnostic RF Fingerprinting," in *Proc. of IEEE INFOCOM'24*, 2024.

[28] F. Afrin, H. Li, B. Wang, and N. Ghose, "Sarp: Spatial agnostic radio fingerprinting with pseudo-labeling," in *Proc. of 2025 IEEE 22nd Consumer Communications Networking Conference (CCNC)*, 2025.