Investigating Sparse Neural Networks for Radio Frequency Fingerprint Identification

Emma Bothereau, Alice Chillet, Robin Gerzaguet, Matthieu Gautier, Olivier Berder

Investigating Sparse Neural Networks for Radio Frequency Fingerprint Identification

Emma BOTHEREAU*, Alice CHILLET*, Robin GERZAGUET*, Matthieu GAUTIER*, Olivier BERDER*,

* Univ Rennes, CNRS, IRISA, firstname.name@irisa.fr

***Abstract*—Radio-Frequency Fingerprint Identification (RFFI) shows promise for enhancing wireless identification security through unique emitter imperfections. However, this method, which primarily relies on deep learning, faces challenges for a real-time deployment on edge devices. Both memory and computational resources are indeed presented as the major obstacle of such deployment. To address these issues, this paper proposes to investigate the behavior of compressed convolutional neural networks when using unstructured pruning in a data-free scenario, on several public datasets. Indeed, unstructured pruning exhibits significant compression capabilities while preserving performance with minimal computation. We show that state-of-the-art RFFI neural networks can be pruned by up to 70% of the weights, while maintaining F1-Scores above 99%, without retraining. Additionally, we advocate the use of data from a later day, referred to as resilience, as an additional indicator of network performance.**

***Index terms* - Radio Frequency Fingerprint Identification, Deep Learning, Pruning, Resilience**

I. Introduction

Radio-Frequency Fingerprint Identification (RFFI) is a technique designed to identify wireless devices by extracting the unique characteristics created by the imperfections of their components. This approach has gained considerable attention in recent decades due to its potential applications in device security. The use of the non-reproducible characteristics of the emitter would essentially make it impossible to compromise a device. Therefore, it could provide a stronger security approach compared to current wireless communication protocols [1]. In this context, the rise of machine learning has given a new impulse to RFFI, aiming to distinguish between different emitters based on the signals they emit.

Deep learning is often associated with high computational and memory requirements, which can be challenging for embedded systems. Therefore, alternative techniques have been developed, including the design of light machine learning algorithms [2] or the reduction of trained networks. To obtain low complexity models, various compression methods have been explored, such as bit quantization, transfer learning, and network pruning [3], [4]. Over the past 5 years, the use of pruning to reduce complexity has significantly increased. Pruning is a method introduced in 1989 by Yann LeCun in the article "Optimal Brain Damage" [5]. It is designed to decrease the complexity of learning models by selectively eliminating network elements with the least influence on the model performance. The ratio of neutralized weights is called *sparsity*. There are several network pruning techniques, including unstructured pruning [6], involving the removal of individual weights, and structured pruning [4], involving the removal of weights in groups that constitute a filter, a channel or a neuron. Another classic pruning method is to remove weights given a set of selected patterns [7].

The field of network pruning for the compression of RFFI networks is relatively new, with only few publications available to date [8], [9], [10]. The proposed methods, all structured pruning algorithms, involves retraining, demanding substantial data and extensive computational resources to produce the final networks. To address this issue, we apply data-free pruning, offering a solution to reduce the size of neural networks without requiring data while preserving the network identification capabilities without the need for additional computational and memory resources for data processing. Moreover, these articles do not explore the behavior of the networks in an ever-changing environment. Yet, the primary goal of RF fingerprint identification is to recognize devices in given scenarios, requiring the network to identify devices across various time periods and possibly different contexts. This implies that the network not only captures the initial channel conditions but also acquires knowledge of device characteristics, referred to as the RF fingerprint. We name this capacity *resilience*.

The key contributions of this paper include:

- A comparative study of unstructured pruning of two Convolutional Neural Networks (CNNs) over four RFFI datasets: these networks can be pruned from 35% up to 70% sparsity, depending on the network and dataset, without retraining, while maintaining a 99% F1-Score.
- An improved classification: with unstructured pruning, F1-Score can increase classification on all tested CNNs and datasets, up to 0.19%.
- A heightened comprehension of network resilience capacity: our findings reveal that resilience strongly relies on the network original structure, and that unstructured pruning does not seem to affect the ability to perform on different days.

Section II introduces RF fingerprint, identification system and the different datasets. Section III analyzes two commonly state-of-the-art CNNs for RFFI. Finally, in Section IV, pruning methods are explored, and results are reported across multiple evaluation criteria.

II. RF Fingerprint Identification Principle

A. RF Fingerprint

The process of transmitting a digital signal consists of several essential steps, including modulation, division into In-phase (I) and Quadrature (Q) channels, amplification, filtering, and transmission via an antenna. At the receiving side, the signal is captured by a receiving antenna and subsequently undergoes amplification, filtering, and demodulation into I and Q channels before recovering the original data. Due to imperfections in the emitter components and the impact of the channel, the received signal contains a slight distortion called an RF fingerprint ($\mathcal{F}$). This received add qualificatif signal x_r, depending on the original signal x can be expressed as :

$$x_r(t) = \mathcal{F} \circ x(t) = \mathcal{F}_{Channel} \circ \mathcal{F}_{PA} \circ \mathcal{F}_{LO} \circ x(t) \quad (1)$$

with $\circ$ the composition function, $\mathcal{F}_{LO}$ as the Local Oscillator fingerprint, $\mathcal{F}_{PA}$ as the Power Amplifier fingerprint, and $\mathcal{F}_{Channel}$ as the signal distortion caused by the channel or environment. We will not consider the receiver fingerprint in the process, as it is the same for all signals received.

B. Pruning-based Identification System

In the studied scenarios, signals emitted by the different transmitters are received and collected to construct a database. This database is then used to train a classifier with the objective of identifying all transmitters independently.

We suggest an additional processing step after the training phase, involving the pruning of the network. Our approach is introduced in Figure 1. While various pruning techniques can be employed, we specifically examine data-free unstructured pruning. This method allows the creation of a sparse neural network with a higher level of sparsity compared to structured pruning, as removing weights individually is easier than removing them in groups, without affecting network capacity. Sparse networks, combined with the acceleration and memory optimization techniques [11], are promising. Nevertheless, it is worth noting that this article does not cover hardware acceleration for sparse networks.

Our goal is to attain the highest level of network sparsity, while preserving its F1-Score performance, without utilizing data for the pruning operation.

C. Datasets

The datasets used in this article are open access WiFi datasets for RFFI. For each dataset, two distinct scenarios are defined, designated as Scenario 1 (S1) and Scenario 2 (S2). The datasets are described in Table I. The signals from Scenario 1 are extracted, shuffled, and divided into training and testing datasets (90% assigned to training and 10% assigned to test). The signals from Scenario 2 are employed as a second test (test S2). We define *resilience* as the network ability to perform with data that is either recorded at a different time or in a different context than the data it was trained on, here the resilience is measured with S2.

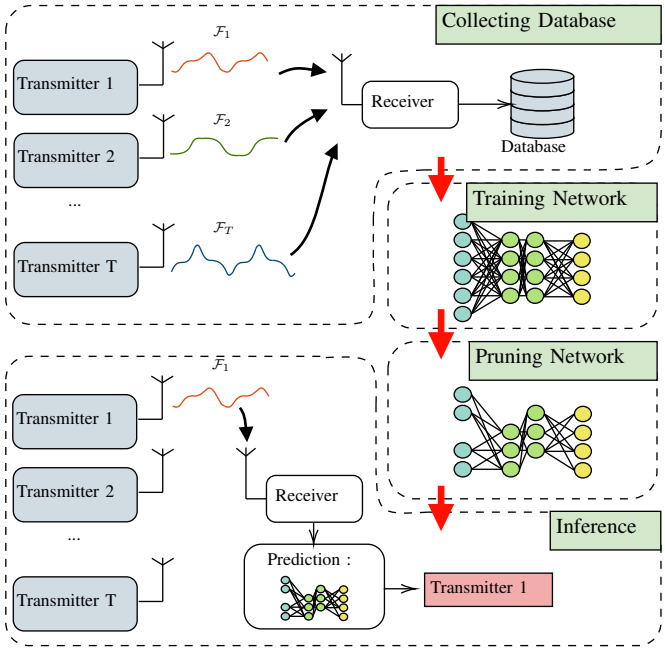

Fig. 1: Proposed system for RF fingerprint identification.

III. Deep Learning for RF Fingerprint Identification

A. Definition

Let $\mathcal{N} = \{(W^k, B^k), k \in [\![1, K]\!]\}$ be a convolutional neural network consisting of K layers, with each layer k defined by a weight matrix W^k representing the weights and a bias vector B^k representing the biases. For simplicity in this paper, the individual weights are referred to as w_j. We define N as the total number of parameters present in network $\mathcal{N}$.

B. Convolutional Neural Networks

Multiple neural networks are used for RF fingerprint classification. There are numerous variations of neural networks, each with its own unique description. Thus, we chose to observe the performance of a reference architecture [15] (CNN_L), and a particularly lightweight network [13] (CNN_S) presented with the WiSig dataset. It illustrate that networks of fundamentally different complexity can follow a similar methodology. We give them the same raw IQ signals in the time domain as input with dimensions of 256×2, representing 256 time samples across two channels (I and Q channels). With theses inputs, CNN_L and CNN_S respectively have 1 232 774 and 39 778 parameters for the classification of six transmitters.

C. Methodology

For reproducibility and representativeness, all experiments are performed on 5 fixed seeds, fixed values used for the pseudorandom number generator. Each seed has an impact on both the train-test split and the weight initialization. All networks are trained on data from S1 (training dataset) and tested on S1 (testing dataset) and S2. Networks are evaluated on their macro F1-Score (expressed as a percentage).

Database	POWDER - WiFi [12]	WiSig - ManySig [13]	Stable WiFi RF Fingerprints (SWRFF) - Random [14]	ORACLE [15]
Description	Recordings on two distincts days.	Recordings on 4 days over a month.	*Enrollment phase*: All devices placed 1 meter away from the receiver. *Deployment phase*: Randomly moved within a 3-meter radius.	Signals recorded from 2ft to 62ft with two runs per location.
Transmitters	4 - USRP X310	6 - Atheros AR5212/AR5213	15 - 10 FiPy & 5 LoPy	16 - USRP X310
Signals	≈11700/day	1000/day/TX	Enrollement: ≈5400/Tx Deployement: ≈4300/Tx	4000/Tx/location/run
Scenario 1	Day 1	Day 1	Enrollement phase (morning)	Run 1 - 2ft
Scenario 2	Day 2	Day 2-3-4	Deployement phase (evening)	Run 2 - 2ft

TABLE I: Summary of RFFI datasets and scenarios chosen.

D. Training behaviour

In this section, we examine the performance of the networks during training, using the Adam optimizer with the best performing learning rate, a loss scheduler, diminishing of 10% the loss value every 10 epochs, and cross-entropy as the loss function. Networks are trained for 200 epochs with an early stopping if the loss does not decrease for 10 epochs. At the end of each training epoch, we evaluate the F1-Score of the networks on two different sets of datatests: Scenario 1 (S1) and Scenario 2 (S2).

The mean, minimum, and maximum F1-Scores of both CNNs for the different databases are shown in Table II. The best results between CNN_L and CNN_S, for each scenario and dataset, are underlined.

Scenario	Mean	Min	Max	Mean	Min	Max
	CNN_L			CNN_S		
	POWDER [12]					
S1	99.99	99.98	100.0	99.78	99.24	99.96
S2	91.14	86.09	95.67	79.41	66.59	98.89
	WiSig [13]					
S1	99.73	99.34	100.0	99.57	99.02	99.85
S2	58.55	51.47	69.32	56.98	49.23	72.75
	SWRFF [14]					
S1	99.63	99.36	99.80	99.61	99.15	99.85
S2	6.74	3.64	9.40	15.06	13.21	16.88
	ORACLE [15]					
S1	99.92	99.67	100.0	98.35	97.36	99.55
S2	29.70	26.26	32.89	26.40	25.10	27.67

TABLE II: Networks F1-Scores on the different datasets.

1) Disparity between Datasets: While all networks perform similarly on S1, their performance on S2 varies significantly. It is important to recognize that the second scenario may differ greatly from the first. For example, the SWRFF dataset has poor F1-Score on S2 due to changing locations, while the POWDER dataset performs well on both S1 and S2, with consistent locations and having less transmitters to identify.

2) Disparity between networks: CNN_L outperforms CNN_S across all databases, on S1, due to its greater depth and parameter count, allowing it to handle more complex data ,thus performing better on data from the same scenario used for training. Despite CNN_S having significantly fewer parameters, its classification performance remains strong, with minor F1-Score reductions ranging from 0.02% to 1.57%. However, CNN_S with it slightly lower F1-Score on the ORACLE database highlights the difficulty of training on this dataset.

3) Resilience: On S2, it is not clear why CNN_S performs better than CNN_L on some seeds. A possible explanation would be that CNN_S being smaller tends to generalize more by not being able to fit the data as close as CNN_L.

4) Variability and Resilience: CNN_S exhibits greater variability than CNN_L across different seed values, particularly in resilience testing with S2, as can be seen with the minimum-maximum intervals.. While CNN_S often shows lower minimum scores compared to CNN_L, some CNN_S realisations outperform any achieved by CNN_L. This variability mainly originates from the initial network weight values rather than the test/train data distribution, indicating that CNN_S higher dependence on initialization is due to its fewer parameters. Consequently, some CNN_S implementations may exhibit superior resilience compared to any CNN_L networks. However, this advantage may be offset by reduced performance on S1, and it is entirely dependent on the seed.

The choice of network architecture is crucial for RF fingerprint identification and should be made carefully. While larger networks may offer greater reliability and adaptability, smaller networks can enhance resilience at the expense of increased instability and sightly lower performance on the first scenario. In the next section, we will delve into pruning as a method to reduce network complexity while maintaining performance.

IV. Pruning

A. Metrics

Pruning can be defined by the removal or zeroing of selected weights within a network. In unstructured pruning, weights are typically individually set to zero, effectively considering them as non-existent components of the network. We call the ratio of weights set to zero sparsity. CNN_L, with a 0.97 sparsity, would be equivalent to CNN_S in terms of remaining parameters.

B. Definition

In the context of pruning, let's define r as the desired sparsity. A mask $\mathcal{M} = \{(M_{W^k}, k \in [\![1, K]\!]\}$ is considered for the network $\mathcal{N}$. The matrices M_{W^k} have the same dimensions as W^k. m_{w_j} refers to the mask value for a given weight w_j,

they take the value 1 if the weight of the network is to be kept and 0 if it is to be pruned. We define the pruned network as:

$$\mathcal{N}_r = \{(W^k \odot M_{W^k}), k \in [\![1, K]\!]\}, \tag{2}$$

with $\odot$ for element-wise multiplication. We then express the sparsity r as:

$$r = 1 - \frac{1}{N} \times \sum_{w_j \in \mathcal{N}} m_{w_j}. \tag{3}$$

The number of biases in a neural network is negligible compared to the number of weights. Therefore, they are ignored during pruning.

C. Criteria for unstructured pruning

To be able to prune, it is necessary to define a criterion. This criterion consists in defining a score S or a rule to select the weights to be removed. Several data-independent criteria can be found in the literature:

1) Random: This criterion involves randomly removing weights across the network.

2) L1 norm: Also known as the magnitude norm, this norm focuses on removing weights with the smallest magnitude [3].

$$S(w_j) = |w_j|. \tag{4}$$

3) LAMP: This criterion is a magnitude-based criterion that can be applied globally to the network [6]. It takes into account the relative magnitude of each weight within a given layer W^k. For $w_j \in W^k$:

$$S(w_j, W^k) = \frac{(w_j)^2}{\sum_{w_i \geq w_j, w_i \in W^k} w_i^2}. \tag{5}$$

4) SynFlow: This criterion [16] is sensitivity-based. It evaluates the weights based on their sensitivity when subjected to a loss function with input data consisting of ones.

$$S(w_j, g(w_j)) = |w_j \times g(w_j)|. \tag{6}$$

Here, $g(w_j)$ represents the gradient value obtained when a loss function is calculated as the sum of all the outputs, given an input data consisting entirely of ones, for the network under study with all its weights considered in absolute value.

All criteria outlined here are meant to be applied with global pruning, implying that weight removal is determined by their scores across the entire network, regardless of their specific locations within the network. However, some norms can also be used for local pruning, which means that we prune each layer with the desired sparsity ratio separately. Local pruning means that all layers will undergo pruning with the same ratio.

L1 is the only criterion presented here applied to local pruning, therefore we will study the L1 criterion for both local and global pruning. Other criteria are only used for global pruning as they are shown to be more effective globally.

D. Pruning algorithm

In local pruning, a mask is first created with the same dimensions as the weights for each dense or convolutional layer, with all weights initially set to one. For each layer, scores are computed for each weight. To achieve the desired sparsity level, we calculate the number of weights that should be set to zero. This requires selecting an appropriate threshold and then setting the masks for weights with scores lower than the threshold to zero. After applying pruning to all layers, an element-wise multiplication of the weight matrices with their respective mask matrices is performed.

In global pruning, the methodology is similar. However, the weights are not removed on a per-layer basis, instead, the specified number of weights with the lowest scores are eliminated, regardless of their position in the network.

E. Experiments

The previous networks (see Table II) undergo pruning at various sparsity levels from 0.05 to 0.95 in 0.05 increments, considering all previously mentioned criteria. These pruned networks are then assessed on both S1 and S2 across all 5 seeds.

The behavior of CNN_L and CNN_S, pruned at different sparsity with the LAMP criterion, are shown in Figure 2 and Figure 3 on S1 and Figure 4 and Figure 5 on S2, respectively.

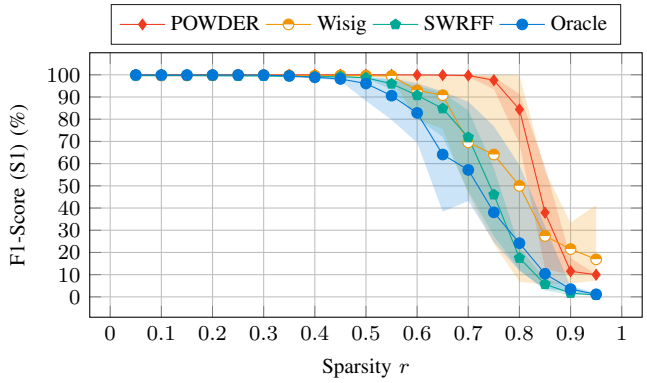

Fig. 2: LAMP pruning on different datasets for CNN_L for S1.

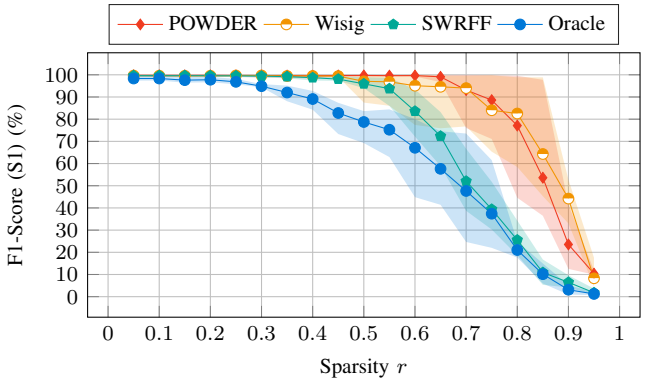

Fig. 3: LAMP pruning on different datasets for CNN_S for S1.

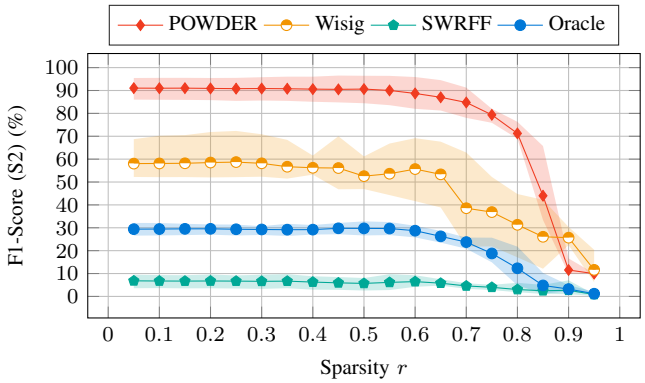

Fig. 4: LAMP pruning for CNN_L for S2.

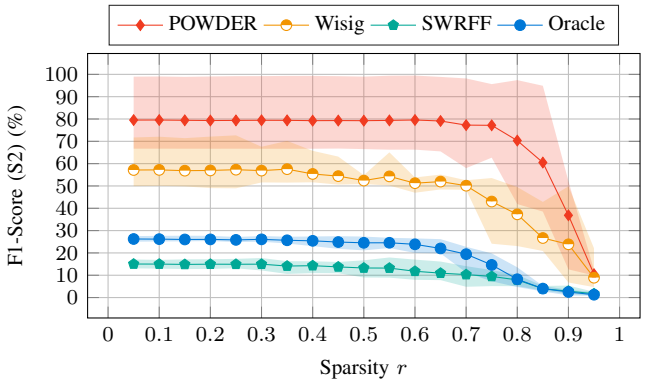

Fig. 5: LAMP pruning for CNN_S for S2.

For the sake of brevity, only this criterion is displayed, as demonstrated later in this section. It is the best performing criterion. The average of the five seeds was plotted for all pruned networks, with the minimum and maximum F1-Scores achieved for each sparsity displayed in transparency. The observations for the LAMP criterion remains valid for the other criteria.

The two CNNs trained on all datasets exhibit different behaviors. Indeed, as the dataset becomes more complex, involving numerous transmitters across various locations and time points, the less the network is able to be sparse with unstructured pruning, as can be observed in the SWRFF and ORACLE datasets. Moreover, it can be observed in Figure 2 and Figure 3 that CNN_L is capable of handling a higher degree of pruning than CNN_S. This can be attributed to the fact that CNN_L has a greater depth and parameter count. Finally, Figure 4 and Figure 5 demonstrate that unstructured pruning, when maintaining the F1-Score on S1, preserves both the original resilience and variability on S2, as was found in Table II.

Performance across all criteria can be then discussed considering three distinct experimentations:

i. Improving F1-Score through pruning,
ii. Achieving the highest pruning sparsity while maintaining over 99% F1-Score,
iii. Achieving the highest pruning sparsity while maintaining over 95% F1-Score.

For each scenario, we calculated the average value across all seeds for all criteria to determine the highest level of sparsity possible. The results are shown in Table III.

1) LAMP vs SynFlow: Local criteria are less effective than global pruning. This is because it uniformly removes weights across all layers. As the pruning ratio increases, layers with fewer parameters are quickly affected, leading to a shortage of connections. Meanwhile, other layers still have many weights that can be pruned without much impact on performance. Regarding global criteria, global L1 is less effective than both SynFlow and LAMP. SynFlow performs better on CNN_S than CNN_L, likely due to the smaller size of the network paired with the efficiency of gradient-based pruning, which seems to be particularly effective on small networks like CNN_S. But, overall, LAMP proves to be the most effective criterion for both networks, allowing at least 35% pruning while preserving 99% of F1-Score.

To enhance the classification process (Experimentation 1), SynFlow appears to be a better choice, as it enhances the F1-Score. Nevertheless, to reduce the number of active parameters, i.e., to achieve higher sparsity (Experimentation 2 and 3),

Experimentation	Criterion	F1-Score S1	F1-Score S2	Sparsity r
CNN_L				
i	SynFlow	100.0	91.1	0.40
ii	LAMP	99.68	84.77	0.70
iii	LAMP	97.56	79.36	0.75
CNN_S				
i	Local L1	99.80	79.42	0.10
ii	LAMP	99.09	79.11	0.65
iii	LAMP	99.09	79.11	0.65

(a) POWDER

Experimentation	Criterion	F1-Score S1	F1-Score S2	Sparsity r
CNN_L				
i	LAMP	99.92	58.17	0.30
ii	LAMP	99.50	53.63	0.55
iii	LAMP	99.50	53.63	0.55
CNN_S				
i	Global L1	99.72	57.01	0.05
ii	LAMP	99.41	54.41	0.45
iii	LAMP	95.09	51.24	0.60

(b) WiSig

Experimentation	Criterion	F1-Score S1	F1-Score S2	Sparsity r
CNN_L				
i	SynFlow	99.64	6.79	0.15
ii	LAMP	99.19	5.93	0.45
iii	LAMP	95.96	6.14	0.55
CNN_S				
i	LAMP	99.62	15.03	0.05
ii	SynFlow	99.23	14.80	0.35
iii	LAMP	95.94	13.25	0.50

(c) SWRFF

Experimentation	Criterion	F1-Score S1	F1-Score S2	Sparsity r
CNN_L				
i	LAMP	99.93	29.35	0.25
ii	LAMP	99.49	29.17	0.35
iii	LAMP	96.08	29.73	0.50
CNN_S				
i	SynFlow	98.37	26.30	0.05
ii	-	-	-	-
iii	SynFlow	95.21	24.75	0.35

(d) ORACLE

TABLE III: Mean F1-Score and Sparsity for Different Models and Criteria.

LAMP appears to be the most suitable candidate for RFFI.

2) Pruning improving F1-Score (S1): On average, for CNN_L, we have a F1-Score increase on S1 of 0.01% to 0.19%, while for CNN_S, the increase is between 0.01% to 0.15% (compared to unpruned networks, Table II). Nevertheless, these gains are observed at low sparsity levels of only 0.05 to 0.10 for CNN_S and 0.15 to 0.30 for CNN_L.

3) CNN_L vs CNN_S: Despite CNN_L having more parameters than CNN_S, the sparsity difference to validate the same experimentation is only 0.05 to 0.15, with CNN_S often achieving smaller sparsity levels. Despite CNN_L being pruned up to 0.70 sparsity on the POWDER dataset in both experimentations ii and iii, it still retains nine times more active parameters than the original CNN_S.

4) Impact of initial design and pruning process: The initial design of the neural network is crucial for achieving the desired F1-Score and network size as all results are strongly correlated to the F1-Score of the original network for both S1 and S2.

Unstructured pruning methods, without retraining, are consistent across all tested datasets, showing that unstructured pruning is a viable method for reducing the number of active parameters of RFFI neural networks.

V. Conclusions

This article proposes the application of unstructured pruning to compress convolutional neural networks (CNNs) without retraining, resulting in sparse networks for Radio Frequency Fingerprint Identification. A comparison of two state-of-the-art networks, CNN_L and the lightweight CNN_S, across four different databases reveals notable differences. CNN_L, with over nine times the parameters of CNN_S, exhibits greater stability across various seeds, while CNN_S instability leads to high-performing realizations only on some seeds.

The study demonstrates that unstructured pruning can achieve sparsity levels ranging from 0.35 to 0.60 without compromising F1-Score or resilience. However, the choice of the initial network significantly impacts the pruned performance.

Furthermore, it was found that the selection of the best criterion depends on the pruning objective: for maximizing sparsity, the LAMP criterion is preferred, whereas SynFlow emerges as an interesting option for enhancing F1-Score.

Further investigation into iterative pruning may prove beneficial in enhancing both F1-Score and sparsity levels. While this method necessitates training data, it mitigates the criticality of the initial network structure and enhances classification accuracy.

Acknowledgement

This work is funded by the French National Research Agency (ANR) under the grant number ANR-22-CE25-0007-01 (RedInBlack project)

References

[1] N. Miguélez-Gómez and E. A. Rojas-Nastrucci, "RF fingerprinting: Hardware-trustworthiness enhancement in the hardware trojan era: RF fingerprinting-based countermeasures," *IEEE Microwave Magazine*, vol. 24, pp. 35–52, 2023.

[2] A. Chillet, B. Boyer, R. Gerzaguet, K. Desnos, and M. Gautier, "Tangled Program Graph for Radio-Frequency Fingerprint Identification," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2023.

[3] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks." *CoRR*, vol. abs/1506.02626, 2015.

[4] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4335–4344, 2019.

[5] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 2. Morgan-Kaufmann, 1989.

[6] J. Lee, S. Park, S. Mo, S. Ahn, and J. Shin, "Layer-adaptive sparsity for the magnitude-based pruning," in *International Conference on Learning Representations (ICLR)*, 2020.

[7] W.-C. Chou, C.-W. Huang, and J.-D. Huang, "Hardware-friendly progressive pruning framework for cnn model compression using universal pattern sets," *International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–4, 2022.

[8] Y. Wang, G. Gui, H. Gacanin, T. Ohtsuki, O. A. Dobre, and H. V. Poor, "An efficient specific emitter identification method based on complex-valued neural networks and network compression," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2305–2317, 2021.

[9] T. Jian, Y. Gong, Z. Zhan, R. Shi, N. Y. Soltani, Z. Wang, J. G. Dy, K. R. Chowdhury, Y. Wang, and S. Ioannidis, "Radio frequency fingerprinting on the edge," *IEEE Transactions on Mobile Computing*, vol. 21, pp. 4078–4093, 2022.

[10] Y. Lin, H. Zha, Y. Tu, S. Zhang, W. Yan, and C. Xu, "GLR-SEI: Green and low resource specific emitter identification based on complex networks and fisher pruning," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–12, 2023.

[11] S. Singh and A. Bhatele, "Exploiting sparsity in pruned neural networks to optimize large model training," *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 245–255, 2023.

[12] G. Reus-Muns, D. Jaisinghani, K. Sankhe, and K. Chowdhury, "Trust in 5g open rans through machine learning: Rf fingerprinting on the powder pawr platform," in *IEEE Globecom 2020-IEEE Global Communications Conference*. IEEE, 2020.

[13] S. S. Hanna, S. Karunaratne, and D. Cabric, "Wisig: A large-scale wifi signal dataset for receiver and channel agnostic rf fingerprinting," *IEEE Access*, vol. 10, pp. 22 808–22 818, 2021.

[14] A. Elmaghbub and B. Hamdaoui, "Eps: Distinguishable iq data representation for domain-adaptation learning of device fingerprints," *arXiv preprint arXiv:2308.04467*, 2023.

[15] K. Sankhe, M. Belgiovine, F. Zhou, L. Angioloni, F. Restuccia, S. D'oro, T. Melodia, S. Ioannidis, and K. R. Chowdhury, "No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, pp. 165–178, 2020.

[16] H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," *ArXiv*, vol. abs/2006.05467, 2020.