Deep neural network goes lighter: A case study of deep compression techniques on automatic RF modulation recognition for Beyond 5G networks

†*Anu Jagannath, †Jithin Jagannath, *Yanzhi Wang, and *Tommaso Melodia

†Marconi-Rosenblatt AI/ML Innovation Lab, ANDRO, NY, USA
*Northeastern University, Boston, MA, USA

ABSTRACT

Automatic RF modulation recognition is a primary signal intelligence (SIGINT) technique that serves as a physical layer authentication enabler and automated signal processing scheme for the beyond 5G and military networks. Most existing works rely on adopting deep neural network architectures to enable RF modulation recognition. The application of deep compression for the wireless domain, especially automatic RF modulation classification, is still in its infancy. Lightweight neural networks are key to sustain edge computation capability on resource-constrained platforms. In this letter, we provide an in-depth view of the state-of-the-art deep compression and acceleration techniques with an emphasis on edge deployment for beyond 5G networks. Finally, we present an extensive analysis of the representative acceleration approaches as a case study on automatic radar modulation classification and evaluate them in terms of the computational metrics.

Keywords: CBRS radar classification, model compression, CNN pruning, lightweight neural networks, FLOPs, model speedup.

1. INTRODUCTION

Neural networks are gaining popularity in the radio frequency (RF) applications. Recently, neural networks have been employed to solve numerous challenging RF applications such as RF fingerprinting, blind modulation classification, wireless standard classification, direction of arrival (DoA) estimation, channel equalization, symbol detection, among others [1–6]. However, a notable trend is in designing wider and deeper neural networks. Among neural networks, the most popular are convolutional neural networks (CNNs) such as visual geometry group (VGG), AlexNet, GoogleNet, ResNet, DenseNet, among others which although designed for computer vision (CV) applications were shown to perform well for the RF applications [7–10]. However, unlike CV applications, the *deployability* of the model gains even more priority in the RF world. This can be attributed to the low-end radio platforms with strict computational, memory, and power constraints where the computational resources must host the neural networks in addition to the complex RF transceiver chains and similar intensive signal processing. The deployability of the model is a less investigated topic in the *neural networks for RF* realm except for a few works in this direction [5, 11, 12]. Deployability must carefully consider the number of layers, neurons, kernel sizes, etc., which significantly impacts the floating point operation (FLOPs), latency, and memory requirements. Designing neural networks is no trivial task especially considering the edge deployment factor. There is no direct way to estimate the number of layers, neurons, kernel sizes, number of kernels, etc., and must be determined empirically. This aspect was carefully and elaborately investigated in our previous work [3] to arrive at an efficient architecture for a modulation and protocol classification problem.

Neural network compression techniques are the next step in further reducing the redundancy imbibed in the neural network architectures. We define *redundant neuron or layer* as the network parameter which does not impart any additional feature extraction capability to the architecture and can be removed without degrading the performance. In the literature so far, the neural network compression techniques are broadly categorized into knowledge distillation [13,14], network pruning [15,16], parameter quantization [17], low-rank approximation [18], etc. Among these, low-rank approximation decomposes a tensor belonging to a trained neural network into smaller dimensions to achieve compression. However, low-rank methods can only decompose tensors one-by-one at each layer and cannot identify redundant parameters in the network. At present, network pruning has

gained significant traction whereby the redundant parameters of a trained network are determined and removed iteratively to achieve compression. These redundant parameters could be filters, neurons or channels that are determined by some specific criterion such as ℓ_1-norm, average percentage of zero activations (APoZ), etc.

Generally, network pruning can be categorized into structured and unstructured pruning. Unstructured pruning does not preserve the network geometry while removing the least significant parameters. On the contrary, structured pruning does not result in irregular and uneven network connections and would fit well in parallel computation platforms. Structured pruning helps achieve direct computational resource savings on embedded platforms and other hardware based systems. This consequently enables modern central processing units (CPUs) and graphical processing units (GPUs) to exploit the computational savings from structured pruning. Lightweight neural networks helps save the sparse parameters in on-chip memory resulting in significant energy savings by reducing the frequent DRAM accesses. In literature often large-sized networks have provided significant performance in solving challenging tasks while small sized networks can be limited in their learning capability. It is therefore appropriate to first design a reasonable sized network that achieves desirable performance followed by pruning them.

In literature, there are several works that closely evaluates the compressed network performance on challenging CV problems [13–15,17,19] while only a few remain for the RF domain [11,12]. This is the first work that analyzes the effect of different pruning strategies on CBRS radar waveform classification. We demonstrate the classification of CBRS radar waveforms by training a dense VGG16 network. The saliency of the different kernels within the various convolutional layers for the trained task under various pruning criteria are portrayed prior to pruning the network under diverse pruning strategies to achieve significant compression. Specifically, we analyze the effect of an iterative (Setup A) and greedy (Setup B) pruning strategies with different pruning algorithms such as ℓ_1-norm [16], APoZ [20], and k-means [19,21] on the radar classification with a pre-trained VGG16 network. We compress the model by 27.47% while achieving 84.79% with no accuracy loss compared to base model with APoZ Setup A. Similarly, under k-means Setup B, the 9.465% compressed model achieved a 85.17% accuracy demonstrating a slightly improved accuracy compared to base model. Most notably, we show that the trained base model can be significantly compressed to 99.74% to achieve a very lightweight model with only 0.04 Million trainable parameters with 80.2% accuracy and a speedup of 381.52×.

2. STRUCTURED DEEP COMPRESSION: PRUNING CONVOLUTIONAL FILTERS

In this section, we introduce the readers to the various pruning strategies that are investigated in this work. The architectures of interest in this article are CNNs. Consequently, we present a systematic walkthrough of the convolutional filter pruning methods. CNNs superior performance in spatial feature extraction can be attributed to their varying sized kernels arranged in the layers to perform strided spatial convolutions to derive condensed feature maps. These feature maps hold the salient features of the input enabling their classification. However, often times the neural network is over-parameterized and would contain redundant filters (kernels) that are expendable. This redundancy opens the door for saliency estimation and redundancy removal without significant loss in accuracy which essentially compresses the network to a smaller size. For instance, well known deep architectures such as VGG16, ResNet50, DenseNet121, among others possess significant redundancy among the different filters and feature maps. Smaller, efficient networks with reduced memory footprint and power consumption promotes the use of CNNs in resource constrained edge platforms. The number of pruned filters will correlate directly with the computational speed up and memory footprint reduction as it reduces the number of trainable parameters.

We favor structured compression over unstructured owing to their ease of implementation on most widely available general purpose hardware. Structured compression involves the removal of structural elements of the CNN such as filter(s) and/or channel(s) that are well supported by various off-the-shelf deep learning libraries.

We present some preliminaries on feature maps from a FLOPs perspective in order to ease the reader into the various pruning strategies. Consider an input feature map $F_i \in \mathbb{R}^{N_i \times H_i \times W_i}$ to a convolutional layer i where N_i, H_i , W_i are the number of input channels, height, and width of the input feature map, respectively. Let the layer i contain N_{i+1} convolutional kernels of dimensions $h_k \times w_k \times W_i$ corresponding to a total number of

learnable parameters of $N_i h_k w_k W_i$. The convolutional layer maps the input feature maps to output tensor $F_{i+1} \in \mathbb{R}^{N_{i+1} \times H_{i+1} \times W_{i+1}}$ which serves as input for the next convolutional layer $i+1$ by the following transformation.

$$F_{i+1}(m_p, n_q) = \sum_{p=1}^{h_k} \sum_{q=1}^{w_k} \sum_{r=1}^{W_i} K_i(p, q, r) F_i(m, n) \tag{1}$$

where the spatial location of the output are $m_p = m - p + 1$ and $n_q = n - q + 1$ considering a unit stride without zero-padding. In other words, each convolutional kernel in layer i of size $h_k \times w_k \times W_i$ generates one feature map. The total number of FLOPs of layer i is $N_{i+1} H_{i+1} W_{i+1} h_k w_k W_i$. Pruning one filter from the layer i will therefore reduce $H_{i+1} W_{i+1} h_k w_k W_i$ FLOPs and $h_k w_k W_i$ trainable parameters. In a nutshell, pruning or removal of a filter will remove a feature map from the layer output which consequently eliminates a channel from the subsequent convolutional layer.

In a broad sense, we can categorize the structured pruning to the following steps:

1. Train a baseline neural network model $\mathcal{F}$ on the target classes.
2. Rank the filters of layer l as per some metric that determines saliency.
3. Prune the filter(s) with the least importance to achieve a target pruning rate in layer l.
4. Retrain and fine-tune the pruned network f to achieve the desired accuracy. Repeat step 3 for the target layers until desired compression ratio is achieved.

2.1 ℓ_1-norm

The authors of [16] proposed an ℓ_1-norm approach to prune the filters of a layer. According to this approach, the ℓ_1-norm of the filters are used as a saliency determination metric to physically prune them from the network. For a layer l, the ℓ_1-norm of a filter at index m, $K^{l,m}$, can be obtained by computing the sum of its absolute weights, as

$$||K^{l,m}||_1 = \sum |k_{i,j}| \tag{2}$$

Recall here that the number of input channels is the same across filters of a layer. In that sense, the ℓ_1-norm also represents the average magnitude of its kernel weights. Hence, this gives an estimate of the magnitude of the output feature maps. The filters are ranked or sorted based on their ℓ_1-norm values and the filters with the smallest values are removed from the layer to achieve desired pruning percentage.

We note here that determining the importance of the filters based on their numerical values would be an insufficient benchmark. For example, consider two 3×3 filters $A = [0.01, 0.005, 0.01; 0.01, 0.8, 0.01; 0.03, 0.001, 0.002]$ and $B = [0.5, 0.5, 0.5; 0.5, 0.5, 0.5; 0.5, 0.5, 0.5]$. The ℓ_1-norms of these would be 0.8 and 1.5 respectively, causing A to be ranked smaller than B. However, intuitively it can be seen that the filter A places more weightage to the center grid unlike B which considers all the grids equally. Hence, different filters activates different spatial locations of the input features. Consequently, considering only the filter coefficients might prove insufficient. Therefore, it becomes necessary to also consider the feature map activations.

2.2 k-means

In this section, we will analyze the effect of clustering on the achievable compression and efficacy in preserving the accuracy. Here, we adopt k-means CNN filter pruning proposed in [**?**] whereby the filters of a layer are subject to k-means clustering. The intuition behind this pruning method is to identify the filters that are located far away from the centroid of the clusters and keep only the ones closest to the center. The outlier filters along with their feature maps and channels of the subsequent convolutional layer will be subject to pruning.

2.3 Zero activation analysis

The two methods discussed above determined the significance of filters based on their coefficients. Recall the numerical example in section 2.1, considering the filter coefficients alone may not correspond to their activations. Therefore, here we will analyze the filter activation-based saliency determination to identify prunable filter(s). The approach in [22] proposes to consider the average percentage of zero (APoZ) activations of the filter for a sample of the examples after the ReLU mapping. The intuition here is that if a filter is rarely activated, then they do not contribute much to the feature extraction. The APoZ metric is computed for each filter f in a layer l over M examples of the validation set as follows,

$$APoZ^{f,l,c} = \frac{\sum_{i=0}^{M}\sum_{j=0}^{N}\mathfrak{F}\left(a_{ij}^{f,l,c} == 0\right)}{MN} \tag{3}$$

where $a^{f,l,c}$ is the activation map of the filter f in layer l, M is the the number of examples in the batch, N is the dimension of the filter along channel c, and $\mathfrak{F}\left(bool\right)$ is 1 if $bool$ condition is true and 0 otherwise. The filters with high mean APoZ are subject to pruning to achieve the desired compression ratio.

3. CASE STUDY: CBRS RADAR WAVEFORM CLASSIFICATION

A 150 MHz bandwidth of the 3.5 GHz Citizens Broadband Radio Services (CBRS) spectrum has been commissioned for shared use by Federal incumbent access users and commercial users. This band was exclusively used by the US Federal government for Navy radar and aircraft communication. Dynamic spectrum access strategies which involve detection of the incumbents in order to free up the spectrum by the secondary commercial users are therefore inevitable to promote a harmonious coexistence between the two entities. This capability also referred to in literature as environmental sensing capability (ESC) currently involves a network of sensors which samples the spectrum to detect the presence of incumbent radar signals. The National Telecommunications and Information Administration (NTIA) has published in [23] radar signal parameter bounds to facilitate a device's ESC performance. The literature from the past few years has only looked at radar detection in the CBRS band [24–26] which identifies whether a radar signal is present or not (binary classification). However, future ESC systems would require radar identification for comprehensive dynamic spectrum access such that the spectrum reallocation can be performed in a more informed manner. Therefore, in this work for the first time, to the best of the authors' knowledge, we investigate the CBRS radar identification problem which is a multi-label classification. We exploit the CBRS radar waveforms dataset in [27] released by National Institute of Standards and Technology (NIST) which follows the radar signal parameter bounds for ESC compliance testing released by NTIA.

The NIST-CBRS dataset is comprised of five radar modulations: P0N#1, P0N#2, Q3N#1, Q3N#2, and Q3N#3 each different from each other in terms of the pulse width, pulses per burst, chirp width (as applicable), and pulse repetition rate as mentioned in [23]. The dataset in addition also contains noise examples which corresponds to the scenario where there are no active radar emissions in the spectrum. Hence, the class labels in the radar identification problem are: P0N#1, P0N#2, Q3N#1, Q3N#2, Q3N#3, and Noise.

The dataset holds a total of 459 GB worth of samples. We have empirically determined that the distribution of each radar waveform in the dataset is uneven. This stems from the original intention of this dataset - radar detection in the CBRS band. However, to aid the convergence and learning of the neural networks for the radar identification task, for our application we selected same number of examples for each class label to prevent any bias during training. The whole dataset is split into training, validation, and test sets with 4080, 1800 , and 2400 examples respectively. Each example in the dataset contains 800k complex inphase-quadrature (IQ) samples. An example snapshot of a 800k samples time-domain view of all the CBRS radar waveforms of signal-to-noise-ratio (SNR) 20 dB is shown in Figure 1. Here, we have marked the radar pulses to show the active radar emission. For training the neural networks we mapped the 800k IQ samples into the corresponding time-frequency (TF) domain of dimensions $128 \times 128 \times 3$. The time domain to TF transformation is obtained by taking a 128-point short time Fourier transform (STFT) spectrogram of the 800k IQ samples. Note here that the TF map contains three channels each with the spectrogram magnitude, power spectrum density (PSD), and phase mapping. This three channel TF map serves as the input to the VGG16 architecture.

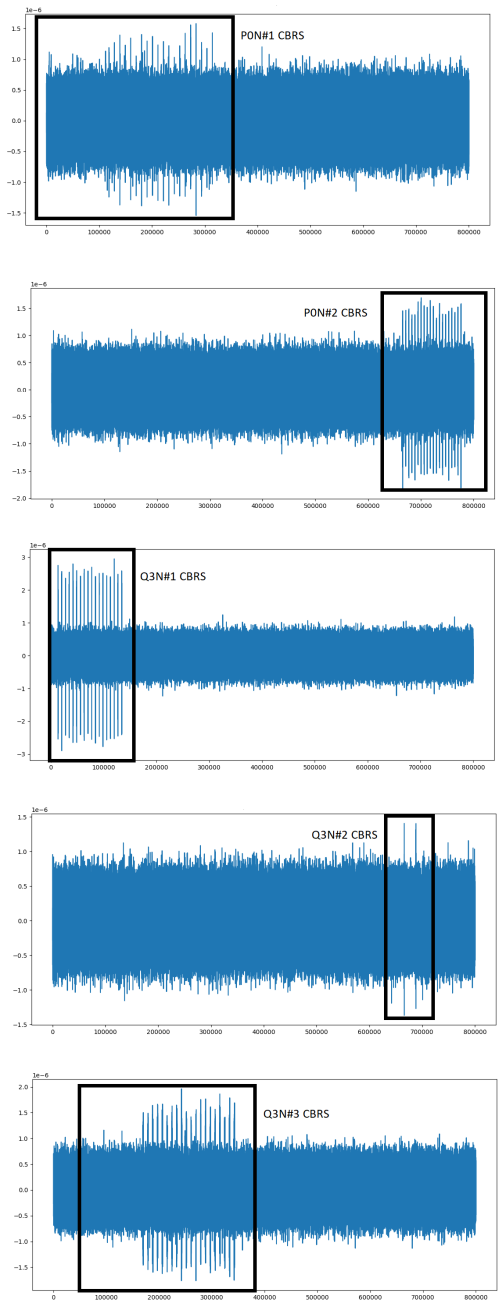

Figure 1. CBRS Radar Waveforms - Time Domain representation at SNR=20dB

4. EXPERIMENTS

4.1 Saliency Analysis: A magnified look

In this section, we take a closer look at the chosen architectures in terms of the filter saliency. We define saliency as the significance of the filter or layer or neuron in extracting the features and contributing to the subsequent layers. Therefore, a layer with more number of salient filters will have a low compression potential. We determine the saliency of the filters in terms of the ℓ_1-norm and APoZ metrics to understand the pruning potential of the chosen neural network architectures with these pruning metrics. We choose VGG16 to evaluate the effects of

various pruning approaches on the computational and performance fronts.

Figure 2 shows the ℓ_1-norm distribution of all the convolutional layers in the VGG16 model (trained on the NIST CBRS dataset) respectively. As per the ℓ_1-norm pruning approach, a higher ℓ_1-norm metric would indicate highly salient filters and vice-versa. Therefore, if any of the layer has a higher percentage of its filters with a low ℓ_1-norm value, it would imply that the layer has several redundant and non-salient filters that can be pruned away without much accuracy loss. The Figure 2 implies that a majority of the filters in the convolutional layers possess a low ℓ_1-norm suggesting a higher pruning potential as per the ℓ_1-norm pruning approach.

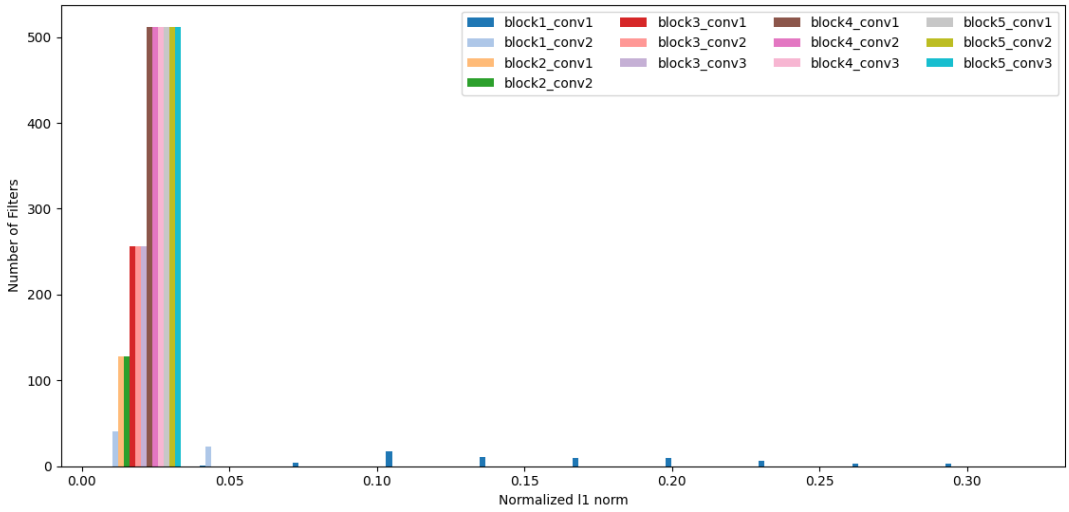

Figure 2. ℓ_1-norm distribution of filters of the VGG16 convolutional layers.

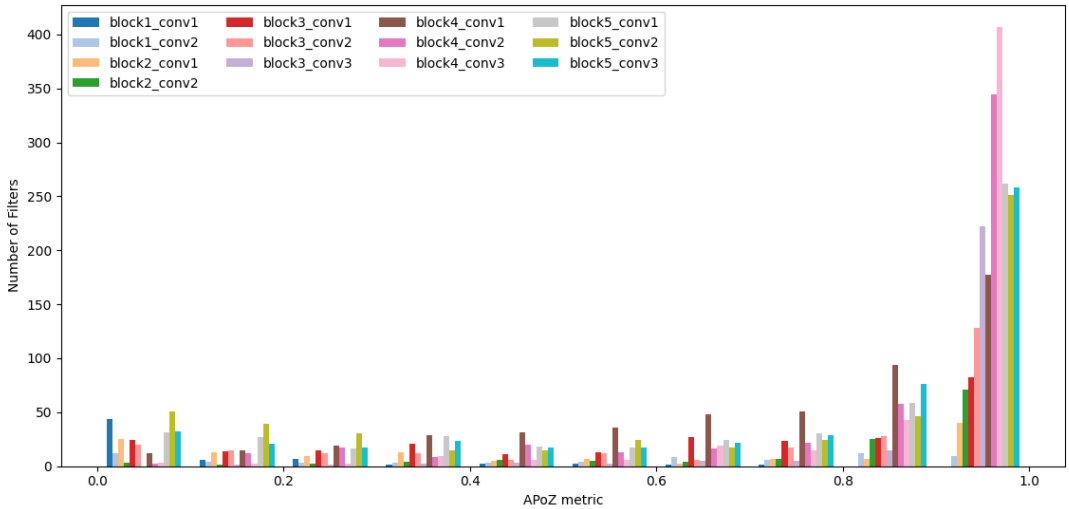

Figure 3. APoZ distribution of filters of the VGG16 convolutional layers.

On the contrary, the APoZ approach measures saliency in terms of the percentage of zero activations as in Equation 3. Accordingly, a higher APoZ indicates the filter is rarely activated implying less significant feature

propagation to the subsequent layer. Hence, a layer with low APoZ for most of its filters will be highly salient and relevant in the feature extraction. The APoZ distribution of all convolutional layers of the VGG16 architecture is depicted in Figure 3. Figure 3 indicates that the later convolutional layers of the VGG16 architecture has a higher percentage of filters that are rarely activated implying immense compression potential. Next, we will closely evaluate the accuracy and computational tradeoffs incurred by pruning these architectures following different pruning strategies in section 4.2.

4.2 Diverse pruning strategies

Typically, pruning is performed iteratively one filter at a time and in a layer-by-layer manner. For deep networks, such as VGG16, ResNet50, etc., such an iterative process could take weeks or months of pruning and retraining. The pruning-and-retraining cost in terms of time and computation is one major tradeoff that comes with a target compressed networks. However, iterative pruning process is often suggested and linked to higher classification accuracy. We pose the approaches in two different settings and critically evaluate how the different strategies affect the accuracy as follows,

1. Setup A: Candidate filters are selected and iteratively removed from multiple layers simultaneously.
2. Setup B: Candidate filters are selected and iteratively removed sequentially from the layers on a layer-by-layer basis.

The prune-and-retrain cost as well as the accuracy vary significantly across the different pruning strategies. We will take a closer look at each of these approaches.

4.2.1 Setup A: Pruning filters iteratively from multiple layers simultaneously

In this setup, the candidate filters are chosen from each convolutional layer and pruned in an iterative manner from multiple layers at once. The algorithm for Setup A is detailed in Algorithm 1. This approach can be considered a one-shot strategy from a layers perspective while iterative in the number of filters being pruned at once from the target layers. This strategy saves significant prune-and-retrain cost by pruning from all target layers at once. We will evaluate how such a pruned model would compare in terms of accuracy to other strategies.

Algorithm 1 Prune-Retrain Setup A

Train a baseline neural network model $\mathcal{F}$
Choose pruning strategy $\mathcal{S}$.
Determine desired pruning percentage p for each layer.
for `iter = 1 to MAX_ITERS` **do**
  if `iter == 1` **then**
    Given model $\mathcal{M} = \mathcal{F}$.
  end if
  Identify prunable filters $\mathbb{F}_l$ $\forall$ target convolutional layers $l \in \mathbb{L} = \{l_1, l_2, \cdots, l_L\}$ of $\mathcal{M}$ based on $\mathcal{S}$ and p.
  Pruning step size $\delta = \min\left(\mathbb{F}_l\right)$
  while $\mathbb{F}_l \neq \emptyset$ **do**
    Prune: From all target layers l, remove δ filters from $\mathbb{F}_l$ to obtain pruned model $\mathcal{F}^{iter}$.
    Update model $\mathcal{M} = \mathcal{F}^{iter}$
    Recompute filter saliency based on $\mathcal{S}$ and update $\mathbb{F}_l$.
    Retrain the pruned model $\mathcal{F}^{iter}$ for N epochs.
  end while
end for
Output: Compressed and fine-tuned model $\mathcal{F}^*$

Here, the `MAX_ITERS` determine the maximum iterations to achieve the desired pruning percentage for each layer. This will is arbitrarily set such that it is large enough to achieve the desired compression for each layer.

4.2.2 Setup B: One-shot filter pruning from multiple layers simultaneously

This approach can be perceived as a greedy approach whereby the candidate prunable filters are removed at once from all the target layers. Unlike, the other two approaches, here the filters do not get to mitigate any feature extraction loss that may arise with the pruning. Rather, here the model is subject to a single-shot pruning and retraining. This pruning strategy denoted as setup B is elaborated in Algorithm 2.

Algorithm 2 Prune-Retrain Setup B

Train a baseline neural network model $\mathcal{F}$
Choose pruning strategy $\mathcal{S}$.
Determine desired pruning percentage p for each layer.
Determine candidate filters $\mathbb{F}_l$ $\forall$ target convolutional layers $l \in \mathbb{L} = \{l_1, l_2, \cdots, l_L\}$ based on $\mathcal{S}$ and p.
Prune: From all target layers l, remove $\mathbb{F}_l$ filters to obtain pruned model $\mathcal{F}^{'}$.
Retrain the pruned model $\mathcal{F}^{'}$ for N epochs.
Output: Compressed and fine-tuned model $\mathcal{F}^{*}$

We would like to point out to the reader that the number of retraining epochs plays a significant role in maintaining the model accuracy. We have empirically determined that greater pruning percentages (p) benefit from higher number of retraining epochs. Accordingly, in all of the evaluations we have set the retraining epochs as 10 for $p < 50$ and as 30 for $p \geq 50$.

5. DISCUSSION

We conduct two broad set of experiments (Setup A and Setup B) each with three pruning algorithms - ℓ_1-norm, APoZ, and k-means - at six pruning ratios yielding a total of thirty six experiments. Having analyzed the saliency distribution, we now look at the pruning potential of the trained architecture. We impose pruning percentage for each convolutional layer to achieve significant model compression. We assess the pruned model in terms of percentage model compression, FLOPs, number of trainable parameters, and model speedup. Below, we define each of these metrics,

1. Percentage model compression - ratio of total number of trainable parameters in pruned model to the total number of trainable parameters in the base model.
2. FLOPs - number of FLOPs in the pruned model.
3. Trainable parameters - total number of trainable parameters in the pruned model.
4. Speedup - is a measure of FLOPs reduction and hence the subsequent computational speedup achieved with pruning. It is estimated as the ratio of number of FLOPs in the base model to the number of FLOPs in the pruned model.

Figure 4 shows the top-1 accuracy of pruned model at different per layer pruning ratios using the different pruning algorithms. It can be seen that for moderate pruning percentages, APoZ serves as a slightly better pruning algorithm while significant pruning percentages above 50% for each convolutional layer diminishes its accuracy. Interestingly, at the maximum pruning rates of 95% per convolutional layer, ℓ_1-norm yields a slightly higher accuracy. We note here that k-means exhibits stable performance across the various pruning rates under setup A.

Now lets investigate how a greedy strategy (Setup B) will affect the compressed model performance in Figure 5. At lower pruning rates ($\leq 30\%$), all the pruning algorithms perform reasonably well under the greedy strategy which makes them ideal to save prune-and-retrain time for lower compression ratios. However, the accuracy takes a steep decline as the pruning rates increase implying for higher pruning rates setup A serves as a good pruning strategy.

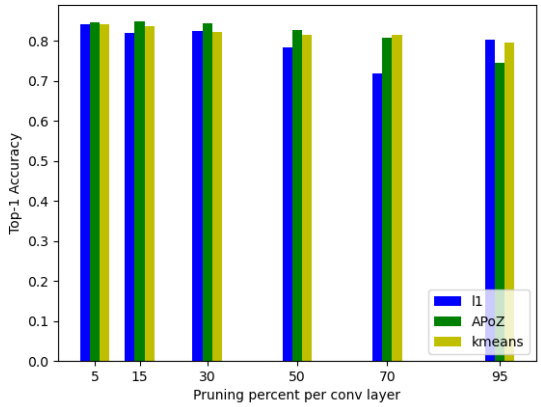

Figure 4. Setup A

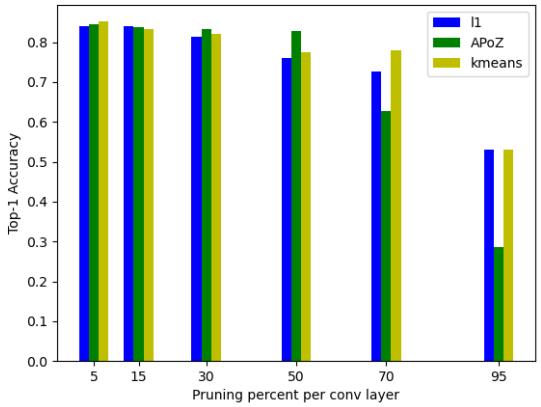

Figure 5. Setup B

We elaborate the in depth comparison of the performance metrics with setup A in Table 1. The 5% pruning rate from each convolutional layer results in a compression percentage of 9.465. At this low compression rate, the APoZ scheme which identifies and removes filters with zero activations outperform the other two pruning algorithms. This trend remains for all compression rates up to 50%. Out of these experiments, the most notable one which yields significant speedup and compression is the k-means pruning at 95% per layer pruning rate. Here, the setup A iterative pruning strategy yields a *very lightweight model that is 99.74% smaller than the base model with a computational speedup of 381.52×*.

Table 1. Setup A

Approaches	**Layer pruning** %	**Model compression** %	**FLOPs (Million)**	**Trainable Parameters (Million)**	**Speedup**	**Top-1 Acc.** %
Baseline	0	0	**29.32**	**140.8**	-	**84.6**
ℓ_1-norm	5	9.465	26.65	13.33	1.1×	84.16
	15	27.47	21.35	10.68	1.38×	82.04
	30	50.78	14.49	7.24	2.03×	82.38
	50	74.98	7.36	3.68	3.99×	78.33
	70	90.93	2.67	1.34	11.03×	71.92
	95	**99.74**	**0.077**	**0.04**	**381.52**×	**80.2**
APoZ	5	9.465	26.65	13.33	1.1×	**84.63**
	15	27.47	21.35	10.68	1.38×	**84.79**
	30	50.78	14.49	7.24	2.03×	**84.42**
	50	74.98	7.36	3.68	3.99×	**82.79**
	70	90.93	2.67	1.34	11.03×	80.79
	95	99.74	0.077	0.04	381.52×	74.58
k-means	5	9.465	26.65	13.33	1.1×	84.16
	15	27.47	21.35	10.68	1.38×	83.75
	30	50.78	14.49	7.24	2.03×	82.29
	50	74.98	7.36	3.68	3.99×	81.54
	70	**90.93**	**2.67**	**1.34**	**11.03**×	**81.38**
	95	99.74	0.077	0.04	381.52×	79.46

Table 2. Setup B

Approaches	**Layer pruning** %	**Model compression** %	**FLOPs (Million)**	**Trainable Parameters (Million)**	**Speedup**	**Top-1 Acc.** %
Baseline	0	0	**29.32**	**140.8**	-	**84.6**
ℓ_1-norm	5	9.465	26.65	13.33	1.1×	84.08
	15	**27.47**	**21.35**	**10.68**	**1.38**×	**84**
	30	50.78	14.49	7.24	2.03×	81.38
	50	74.98	7.36	3.68	3.99×	76.04
	70	90.93	2.67	1.34	11.03×	72.71
	95	99.74	0.077	0.04	381.52×	52.92
APoZ	5	9.465	26.65	13.33	1.1×	84.42
	15	27.47	21.35	10.68	1.38×	83.67
	30	50.78	14.49	7.24	2.03×	**83.17**
	50	74.98	7.36	3.68	3.99×	**82.79**
	70	90.93	2.67	1.34	11.03×	62.75
	95	99.74	0.077	0.04	381.52×	28.5
k-means	5	**9.465**	**26.65**	**13.33**	**1.1**×	**85.17**
	15	27.47	21.35	10.68	1.38×	83.38
	30	50.78	14.49	7.24	2.03×	81.96
	50	74.98	7.36	3.68	3.99×	77.58
	70	90.93	2.67	1.34	11.03×	**78.04**
	95	99.74	0.077	0.04	381.52×	**53.13**

Similarly, Table 2 demonstrates the performance metrics of the compressed model under different pruning rates and approaches. It is evident here that such a greedy strategy will only suit lower compression rates with the peak performance at 5% per layer pruning which yields a 9.465% model compression and a 1.1× speedup using k-means. Similarly, ℓ_1-norm achieves an 84% accuracy for a 27.47% model compression and speedup of 1.38×. We note here that beyond 50% pruning rate for each convolutional layer, the greedy setup B is not a suitable pruning strategy. The overall diminished performance at higher pruning rates with setup B in contrast to setup A can be attributed to the lack of iterative prune-and-retrain which facilitates the model to relearn and improve its generalization capability. Intuitively, the advantage of setup B is faster pruning since it does not involve iterative step-by-step retraining.

6. CONCLUSION AND FUTURE WORK

In this paper, we investigated diverse pruning strategies and their effect on the classification accuracy with an application on CBRS radar waveform classification. This is the first work that presents an elaborate case study of the saliency analysis and different compression approaches in terms of effective model compression percentage, model speedup, number of FLOPs and trainable parameters, and classification accuracy. We demonstrate that the setup A iterative pruning strategy can yield a significantly compressed model by 99.74% with a computational speedup of 381.52× and only 0.04M trainable parameters preserving an accuracy of 80.2%. We demonstrate that the greedy and faster pruning setup B will serve as a good alternative for lower compression rates. As part of our future work, we plan to analyze more dense architectures and incorporate additional pruning algorithms to test the bounds of compression. We note here that reducing the pruning and retraining time to achieve faster compression is still an open challenge requiring significant investigation.

REFERENCES

[1] Jagannath, J., Polosky, N., Jagannath, A., Restuccia, F., and Melodia, T., "Machine learning for wireless communications in the internet of things: A comprehensive survey," *Ad Hoc Networks (Elseier)* **93**, 101913 (2019).

[2] Jagannath, J., Polosky, N., O'Connor, D., Theagarajan, L., Sheaffer, B., Foulke, S., and Varshney, P., "Artificial neural network based automatic modulation classifier for software defined radios," in [*Proc. of IEEE International Conference on Communications (ICC)*], (May 2018).

[3] Jagannath, A. and Jagannath, J., "Multi-task Learning Approach for Automatic Modulation and Wireless Signal Classification," in [*Proc. of IEEE International Conference on Communications (ICC)*], (June 2021).

[4] Jagannath, A., Jagannath, J., and Kumar, P. S. P. V., "A comprehensive survey on radio frequency (rf) fingerprinting: Traditional approaches, deep learning, and open challenges," (2022).

[5] Fu, X., Gui, G., Wang, Y., Ohtsuki, T., Adebisi, B., Gacanin, H., and Adachi, F., "Lightweight automatic modulation classification based on decentralized learning," *IEEE Transactions on Cognitive Communications and Networking* **8**(1), 57–70 (2022).

[6] Jagannath, A. and Jagannath, J., "Dataset for modulation classification and signal type classification for multi-task and single task learning," *Computer Networks* **199**, 108441 (2021).

[7] Peng, S., Jiang, H., Wang, H., Alwageed, H., Zhou, Y., Sebdani, M. M., and Yao, Y., "Modulation classification based on signal constellation diagrams and deep learning," *IEEE Transactions on Neural Networks and Learning Systems* **30**(3), 718–727 (2019).

[8] Al-Shawabka, A., Restuccia, F., D'Oro, S., Jian, T., Costa Rendon, B., Soltani, N., Dy, J., Ioannidis, S., Chowdhury, K., and Melodia, T., "Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting," in [*IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*], 646–655 (2020).

[9] Sankhe, K., Belgiovine, M., Zhou, F., Angioloni, L., Restuccia, F., D'Oro, S., Melodia, T., Ioannidis, S., and Chowdhury, K., "No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments," *IEEE Transactions on Cognitive Communications and Networking* **6**(1), 165–178 (2020).

[10] Ramjee, S., Ju, S., Yang, D., Liu, X., Gamal, A. E., and Eldar, Y. C., "Fast deep learning for automatic modulation classification," *ArXiv* **abs/1901.05850** (2019).

[11] Jian, T., Gong, Y., Zhan, Z., Shi, R., Soltani, N., Wang, Z., Dy, J. G., Chowdhury, K. R., Wang, Y., and Ioannidis, S., "Radio frequency fingerprinting on the edge," *IEEE Transactions on Mobile Computing* , 1–1 (2021).

[12] Jagannath, A. and Jagannath, J., "Multi-task learning approach for modulation and wireless signal classification for 5g and beyond: Edge deployment via model compression," (2022).

[13] Gong, Y., Liu, L., Yang, M., and Bourdev, L. D., "Compressing deep convolutional networks using vector quantization," *ArXiv* **abs/1412.6115** (2014).

[14] Liu, R., Fusi, N., and Mackey, L., "Model compression with generative adversarial networks," *CoRR* **abs/1812.02271** (2018).

[15] Cun, Y. L., Denker, J. S., and Solla, S. A., "Optimal brain damage," in [*Proc. of Advances in Neural Information Processing Systems*], 598–605, Morgan Kaufmann (1990).

[16] Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P., "Pruning filters for efficient convnets," in [*Proc. of International Conference on Learning Representations*], Loew, M. H., ed., *Proc. ICLR* (2016).

[17] Liang, T., Glossner, C. J., Wang, L., and Shi, S., "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing* **461**, 370–403 (2021).

[18] Tai, C., Xiao, T., Wang, X., and Weinan, E., "Convolutional neural networks with low-rank regularization," *arXiv: Learning* (2016).

[19] Li, L., Xu, Y., and Zhu, J., "Filter level pruning based on similar feature extraction for convolutional neural networks," in [*IEICE Transactions on Information and Systems*], **E101.D**, 1203–1206 (2018).

[20] Hu, H., Peng, R., Tai, Y.-W., and Tang, C.-K., "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *ArXiv* **abs/1607.03250** (2016).

[21] Hartigan, J. A. and Wong, M. A., "A k-means clustering algorithm," *JSTOR: Applied Statistics* **28**(1), 100–108 (1979).

[22] Hu, H., Peng, R., Tai, Y.-W., and Tang, C.-K., "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *ArXiv* **abs/1607.03250** (2016).

[23] Sanders, F. H., Carroll, J. E., Sanders, G. A., Sole, R. L., Devereux, J. S., and Drocella, E. F., "Procedures for laboratory testing of environmental sensing capability sensor devices ." National Telecommunications and Information Administration, Boulder, CO. Available: http://www.its.bldrdoc.gov/publications/3184.aspx.

[24] Troglia, M., Melcher, J., Zheng, Y., Anthony, D., Yang, A., and Yang, T., "Fair: Federated incumbent detection in cbrs band," in [*Proc. of IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*], 1–6 (2019).

[25] Caromi, R. and Souryal, M., "Detection of incumbent radar in the 3.5 ghz cbrs band using support vector machines," in [*Proc. of Sensor Signal Processing for Defence Conference (SSPD)*], 1–5 (2019).

[26] Sarkar, S., Buddhikot, M., Baset, A., and Kasera, S. K., "Deepradar: A deep-learning-based environmental sensing capability sensor design for cbrs," in [*Proc. of the 27th Annual International Conference on Mobile Computing and Networking*], *MobiCom '21*, 56–68, Association for Computing Machinery, New York, NY, USA (2021).

[27] Caromi, R., Souryal, M., and Hall, T., "F Dataset of Incumbent Radar Signals in the 3.5 GHz CBRS Band." Journal of Research (NIST JRES), National Institute of Standards and Technology, Gaithersburg, MD. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=929268 (2019). (Accessed March 2, 2022).