Adversarial Robustness of Distilled and Pruned Deep Learning-based Wireless Classifiers

Nayan Moni Baishya and B. R. Manoj
Department of Electronics & Electrical Engineering, Indian Institute of Technology Guwahati, India.
Emails: {nmb94, manojbr}@iitg.ac.in

***Abstract*—Data-driven deep learning (DL) techniques developed for automatic modulation classification (AMC) of wireless signals are vulnerable to adversarial attacks. This poses a severe security threat to the DL-based wireless systems, specifically for edge applications of AMC. In this work, we address the joint problem of developing optimized DL models that are also robust against adversarial attacks. This enables efficient and reliable deployment of DL-based AMC on edge devices. We first propose two optimized models using knowledge distillation and network pruning, followed by a computationally efficient adversarial training process to improve the robustness. Experimental results on five white-box attacks show that the proposed optimized and adversarially trained models can achieve better robustness than the standard (unoptimized) model. The two optimized models also achieve higher accuracy on clean (unattacked) samples, which is essential for the reliability of DL-based solutions at edge applications.**

***Index Terms*—Adversarial attacks, adversarial training, deep learning, modulation classification, knowledge distillation, pruning, wireless security.**

I. Introduction

Deep learning (DL), the cornerstone of modern artificial intelligence systems, has empowered researchers to effectively solve some of the most challenging problems in diverse fields, such as healthcare, natural language processing, and computer vision [1]. Inspired by their tremendous success, DL-based approaches are adopted in wireless communication domain for both classification-based [2]–[4] and regression-based applications [5]. Compared to the conventional probabilistic decision theory-based methods, the DL-based approaches achieve better performance and provide significant computational advantages, such as extracting superior features directly from a large corpus of wireless signal data to develop more complex models and scalability to multiple use cases.

In practice, to accomplish the potential of DL-based solutions for wireless communication applications, it is of paramount importance to design these solutions for successful deployment on edge devices. By combining the advent of the next-generation communication technology, the DL-based approaches can empower edge devices, such as drones and IoT systems, to perform intelligent wireless communication tasks efficiently and autonomously [6]. Edge devices equipped with DL capabilities can process data locally, reducing the need for transmitting large volumes of data to centralized servers, thus conserving bandwidth and enhancing privacy. However, edge devices generally have power constraints and limited computational resources, and the complex, over-parametrized DL models must be optimized before deployment to run efficiently with faster inference, less compute requirement, and lower power and memory consumption. The most commonly used methods to achieve model optimization are knowledge distillation (KD), network pruning, and model quantization [7]. KD is a powerful method to transfer the rich knowledge learned by a complex, large deep neural network (DNN) to a lightweight network to achieve comparable performance. In network pruning, the less important neurons or weights of a DNN are identified and removed to make the network sparse. The sparsity will benefit a model to achieve faster inference and lesser storage requirements. Model quantization also aims to achieve model storage optimization by representing the weights of a DNN at a reduced precision.

Although the ability to deploy optimized DL models on edge devices is beneficial for enhancing privacy through local data handling and computation, a critical security threat to such applications is the vulnerability of DL models to various attacks, namely, adversarial attacks, data poisoning, model extraction [8], etc. This work primarily focuses on the threat towards optimized DL models against adversarial attacks, where a malicious adversary generates an adversarial example by adding a well-crafted perturbation to the input signal, which could lead to an incorrect prediction by the DL model. The adversarial perturbations are low-power signals that are hard to detect. The adversarial attacks can be classified as: a) white-box (WB) and b) black-box (BB) attacks. In a WB attack, the adversary has access to the trained DL model parameters and the training data to generate the adversarial perturbations, e.g., fast gradient method (FGM), fast gradient sign method (FGSM), projected gradient descent (PGD) [9], etc. On the other hand, the adversary lacks model-related information in a BB attack, such as the universal adversarial perturbation (UAP) [10].

This work focuses on the development of DL models that are optimized for edge devices as well as robust against adversarial attacks, with automatic modulation classification (AMC) of radio-frequency (RF) signals as the wireless application of interest. AMC is a safety-critical task with different applications, such as cognitive radio, signal detection and demodulation, and spectrum monitoring as well as management [11]. In recent years, DL-based methods for AMC have been proposed based on convolutional neural networks (CNN) [2], [3], long short-term memory (LSTM) [12], and transformer architectures [13]. These are generally complex networks with millions of trainable parameters that have to be optimized for

This work was supported in part by SERB Start-Up Research Grant (SRG) Scheme, Department of Science and Technology (DST), Govt. of India under Grant SRG/2022/001214 and in part by Start-Up Grant of Indian Institute of Technology Guwahati.

deployment in edge devices. It has also been shown in the literature that the DL models for AMC are highly susceptible to both WB and BB adversarial attacks [10]. Thus, several defense techniques are proposed based on adversarial training (AT) [14]–[16], randomized smoothing [15], GAN [17], etc. Amongst the defense methods, AT is found to provide superior performance for improving the robustness of the DL models. In the AT method, adversarial examples are augmented to the training dataset so that the DL model learns features from the adversarial input space. In the literature, the effectiveness of the AT method has been demonstrated for AMC [14]–[16]. To the best of the author's knowledge, there has not been any work in the investigation of the robustness of optimized DL models against adversarial attacks. Moreover, a key drawback of applying AT on large, complex models is the computational complexity associated with the generation of adversarial examples and the training process. The computational cost is significantly higher for examples generated using an iterative adversarial attack having a better attack success rate, such as PGD, momentum iterative method [18], etc. This results in a great challenge for performing on-device AT at the edge applications where computational resources are scarce. Thus, now more than ever, there is a great need to develop optimized, computationally efficient DL models for AMC that are also robust to adversarial attacks.

Specifically, the main contributions of this work are as follows: (a) We propose to implement KD and network pruning to develop the optimized and lightweight DL models for AMC. Through optimization, the computational cost of AT can also be significantly reduced to enable enhanced security of DL models with edge computing. (b) We show that both distillation and pruning are effective for developing optimized and robust DL models for AMC with a computationally efficient AT process. (c) For AT, we propose to utilize a combination of single-step and multi-step attacks, i.e., FGSM and PGD samples, and show that it also helps in achieving significant robustness against other unseen attacks (FGM, Deepfool, and UAP). (d) To further ensure secure deployment, the classification accuracy of all the adversarially trained models for clean samples should not be affected significantly. We have observed that the optimized models developed in this work perform much better than the standard model after AT.

II. Optimized DL models for AMC

Two optimized DL models for AMC are developed to investigate the robustness against adversarial attacks, and in this work, they are named *distilled* and *distill-pruned* models.

A. Distilled model

To develop this model, we have implemented the vanilla KD method proposed in [19]. KD has been widely adopted for optimizing the DNN for deployment in resource-constrained edge devices [20]. The main idea is to transfer the knowledge from the learned representations of a large, complex model (teacher model) to a smaller, less complex model (student model). The student model will provide several computational advantages for edge devices, such as faster inference and less storage and power requirements. Moreover, the student model can also achieve performance comparable to or even better than the teacher model, which is an additional important benefit of KD. Algorithm 1 presents the development of the distilled model using the vanilla KD method [19]. In the algorithm, the pre-trained teacher model is denoted by $f_{\mathcal{T}}(.;\boldsymbol{\theta}_{\mathcal{T}})$, where $\boldsymbol{\theta}_{\mathcal{T}}$ are the trained parameters and the untrained student model is denoted by $f_{\mathcal{D}}(.;\boldsymbol{\theta}_{\mathcal{D}})$, where $\boldsymbol{\theta}_{\mathcal{D}}$ are trainable parameters. The trained student model obtained after the distillation process is called the distilled model. The knowledge transfer process takes place by minimizing the distance between the output probability distributions of the teacher and student models, i.e., $\hat{\mathbf{Y}}_{\mathcal{T},B}$ and $\hat{\mathbf{Y}}_{\mathcal{D},B}$, respectively. The output probabilities are computed by applying $\mathrm{Softmax}_T(\cdot)$ on the logit values, where the temperature parameter T regulates the softness of the output probability distributions. During distillation, a high value of T is used, and T is set to 1 during inference. In Algorithm 1, we have used the Kullback-Leibler (KL) divergence, $KL_{Div.}(\cdot)$ to compute the distillation loss, $\mathcal{L}_d$ between $\hat{\mathbf{Y}}_{\mathcal{T},B}$ and $\hat{\mathbf{Y}}_{\mathcal{D},B}$. For the classification loss of the student, i.e., $\mathcal{L}_c$, we have used the cross-entropy loss. The total loss for the student model, $\mathcal{L}_t$, is the weighted sum of $\mathcal{L}_d$ and $\mathcal{L}_c$, with α being the weight of the distillation loss.

Algorithm 1: Distilled model

Input: Student $f_{\mathcal{D}}(.;\boldsymbol{\theta}_{\mathcal{D}})$, teacher $f_{\mathcal{T}}(.;\boldsymbol{\theta}_{\mathcal{T}})$, clean training data $\mathbf{X} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, $\mathcal{L}_d$, $\mathcal{L}_c$, temperature T, weight α, epochs E, batch size B

Initialize: Model parameters $\boldsymbol{\theta}_{\mathcal{D}}$

Number of batches per epoch: $N_B = \text{ceil}(N/B)$
for m in $\{1, 2, \ldots, E\}$ **do**
  for n in $\{1, 2, \ldots, N_B\}$ **do**
    Randomly sample a batch: $(\mathbf{X}_B, \mathbf{Y}_B)$
    Teacher Predictions for the batch: $\hat{\mathbf{Y}}_{\mathcal{T},B} = \mathrm{Softmax}_T(f_{\mathcal{T}}(\mathbf{X}_B, \boldsymbol{\theta}_{\mathcal{T}}))$
    Student Predictions for the batch: $\hat{\mathbf{Y}}_{\mathcal{D},B} = \mathrm{Softmax}_T(f_{\mathcal{D}}(\mathbf{X}_B, \boldsymbol{\theta}_{\mathcal{D}}))$
    Distillation loss: $\mathcal{L}_d = KL_{Div.}(\hat{\mathbf{Y}}_{\mathcal{T},B}, \hat{\mathbf{Y}}_{\mathcal{D},B})$
    Classification loss: $\mathcal{L}_c = CE(\hat{\mathbf{Y}}_{\mathcal{D},B}, \mathbf{Y}_B)$
    Total loss: $\mathcal{L}_t = \alpha \cdot \mathcal{L}_d + (1-\alpha) \cdot \mathcal{L}_c$
    // Compute gradient of $\mathcal{L}_t$ and update $\boldsymbol{\theta}_{\mathcal{D}}$
    $\boldsymbol{\theta}_{\mathcal{D}} \leftarrow \boldsymbol{\theta}_{\mathcal{D}} - \nabla_{\boldsymbol{\theta}_{\mathcal{D}}} \mathcal{L}_t$
  endfor
endfor

Output: Distilled model $f_{\mathcal{D}}(.;\boldsymbol{\theta}_{\mathcal{D}})$

For demonstration purposes, this work considers the VTCNN2 model in [2] as the student model and the InceptionNet model in [3] as the teacher model. The VTCNN2 and the InceptionNet models have 2.83M and 10.07M parameters, respectively. Both architectures are publicly available, which is beneficial for the reproducibility of our results. The parameters $T = 10$ and $\alpha = 0.1$ are chosen for this work. For the remainder of the manuscript, $f_{\mathcal{D}}$ will refer to the distilled VTCNN2 model, where knowledge is distilled from the InceptionNet teacher model. Also, the original VTCNN2 model trained specifically for AMC as in [2] (without KD) is referred to as the standard model $f_{\mathcal{S}}$ in this work.

Algorithm 2: Distill-pruned model

Input: Distilled model $f_{\mathcal{D}}(.;\boldsymbol{\theta}_{\mathcal{D}})$, normalized weight matrices $\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_L$, L layers, prune-layer index k, data matrix $\mathbf{U} \subset \mathbf{X}$, $\eta > 0$

// Calculate layer-wise activations with original weights
$\mathbf{Y}_0 = \mathbf{U}$ // Input data
for l in $\{1, 2, \ldots, L\}$ **do**
  $\mathbf{Y}_l \leftarrow \max(\mathbf{W}_l^\top \mathbf{Y}_{l-1}, \mathbf{0})$ // Activations before pruning
endfor
$\hat{\mathbf{W}}_k \leftarrow \mathrm{TRIM}(\mathbf{Y}_{k-1}, \mathbf{Y}_k, \mathbf{0}, \eta)$ // Apply TRIM on layer k
Update: $\mathbf{W}_k \leftarrow \hat{\mathbf{W}}_k$

Output: Distill-pruned model $f_{\mathcal{P}}(.;\boldsymbol{\theta}_{\mathcal{P}})$

B. Distill-pruned model

In this method, we combine KD and network pruning to optimize the DL model for AMC further. Specifically, the goal is to incorporate the complementary benefits of the two methods, i.e., knowledge transfer from KD and sparsity from pruning, to optimize a DL model. Network pruning is powerful for model optimization because many parameters do not contribute significantly to the network's performance and, therefore, can be removed or pruned. From an edge application perspective, pruning can have several benefits, such as faster inference, less storage and compute requirements, and increased generalization. We first obtain the distilled model $f_{\mathcal{D}}(.;\boldsymbol{\theta}_{\mathcal{D}})$ using Algorithm 1, followed by applying the Net-Trim (NT) pruning method in [21]. The NT algorithm optimizes a model by maximizing the sparsity in the layer weights while minimizing the difference between the post-pruning output response and the initial output response of a layer. This can be formulated as a constrained optimization problem for a layer with index k as given by [21],

$$\hat{\mathbf{W}}_k = \arg\min_{\mathbf{V}_k} \|\mathbf{V}_k\|_1 \text{ s.t. } \quad \|\hat{\mathbf{Y}}_k - \mathbf{Y}_k\|_F \leq \eta\,,$$

where $\hat{\mathbf{Y}}_k = \max(\mathbf{V}_k^\top \mathbf{Y}_{k-1}, 0)$ is the output activation of the k^{th} layer for the intermediate weight matrix $\mathbf{V}_k$ during optimization, with $\mathbf{Y}_{k-1}$ being the output of the previous layer, $(\cdot)^\top$ denotes transpose, $\|\cdot\|_F$ is the Frobenius norm, and $\eta > 0$ is the threshold. In our implementation, the first fully-connected (FC) layer of the distilled VTCNN2 model $f_{\mathcal{D}}$ is pruned as it has the highest number of parameters, i.e., 2.7M out of the total 2.83M parameters, and pruning this layer will effectively optimize the overall model. The final sparse weight matrix $\hat{\mathbf{W}}_k$ is obtained by solving the problem in (1) using the alternating direction method of multipliers (ADMM) technique [21]. After updating the original weights $\mathbf{W}_k$ to the sparse weights $\hat{\mathbf{W}}_k$, we can obtain the distill-pruned model, denoted as $f_{\mathcal{P}}(.;\boldsymbol{\theta}_{\mathcal{P}})$, where $\boldsymbol{\theta}_{\mathcal{P}}$ are the parameters after pruning. The overall procedure is presented in Algorithm 2. In the algorithm, the TRIM method depicts the iterative solution of the optimization problem in (1) to obtain $\hat{\mathbf{W}}_k$ and the parameter η controls the extent of sparsity in the weight matrix. To generate the layer-wise output activations from the distilled model, only a small subset of samples $\mathbf{U}$, randomly chosen from the training dataset $\mathbf{X}$ for $f_{\mathcal{D}}$, is utilized in Algorithm 2. In this work, we have used $\eta = 0.8$ to achieve a 96.5% sparsity, which means that only 94.47K weights are non-zero out of the total 2.7M weights.

III. Adversarial attacks

A. Attack model

In general, we denote a trained DL-based wireless signal classifier as $f(.;\boldsymbol{\theta}) : \boldsymbol{x} \in \mathcal{X} \rightarrow \boldsymbol{y} \in \mathcal{Y}$, where $\boldsymbol{\theta}$ are the trained parameters of the model, $\boldsymbol{x}$ is the clean complex-valued input RF signal (no attack) in $\mathcal{X} \subset \mathbb{R}^{2\times n}$, referring to the in-phase (I) and quadrature (Q) components of dimension n. $\boldsymbol{y}$ is the clean output probability vector in $\mathcal{Y} \subset \mathbb{R}^K$, where K is the output dimension which corresponds to the number of modulation schemes. The goal of the adversary is to generate an adversarial perturbation for the input signal $\boldsymbol{x}$, denoted as $\boldsymbol{\delta}$, specific to the attacked classifier $f(.;\boldsymbol{\theta})$. The adversarial example is then generated as $\boldsymbol{x}_{adv} = \boldsymbol{x} + \boldsymbol{\delta}$. When $\boldsymbol{x}_{adv}$ is provided as the input signal during inference, the classifier predicts the corresponding output label as $\hat{l}(\boldsymbol{x}_{adv}) = \arg\max_j f^j(\boldsymbol{x}_{adv};\boldsymbol{\theta})$, where $f^j(\boldsymbol{x}_{adv};\boldsymbol{\theta})$ is the output probability of the classifier corresponding to the j-th class. If $l(\boldsymbol{x})$ is the original label of the clean RF signal $\boldsymbol{x}$, then the adversarial attack will be successful if $l(\boldsymbol{x}) \neq \hat{l}(\boldsymbol{x}_{adv})$. In this work, we focus on untargeted adversarial attacks, where the predicted label $\hat{l}(\boldsymbol{x}_{adv})$ can be any other class except the original label $l(\boldsymbol{x})$.

B. White-box attacks

a) FGM: In this attack method, the $\boldsymbol{x}_{adv}$ is generated by solving a constrained optimization problem as given by

$$\arg\max_{\boldsymbol{x}_{adv}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_{adv}, \boldsymbol{y}), \quad \text{s.t.} \quad \|\boldsymbol{x}_{adv} - \boldsymbol{x}\|_2 \leq \epsilon\,, \tag{1}$$

where $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_{adv}, \boldsymbol{y})$ is the loss for $f(.;\boldsymbol{\theta})$ when $\boldsymbol{x}_{adv}$ is the input signal, $\|\cdot\|_2$ is the L_2-norm and ϵ is the allowed perturbation. The solution to (1) is given by

$$\boldsymbol{x}_{adv} = \boldsymbol{x} + \epsilon \cdot (\|\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})\|)^{-1} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})\,, \tag{2}$$

b) FGSM: This method follows the same optimization problem in (1) to generate the $\boldsymbol{\delta}$, except the constraint that is being subjected to is the L_∞-norm instead of the L_2-norm, i.e., $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$. The resultant $\boldsymbol{x}_{adv}$ is given by

$$\boldsymbol{x}_{adv} = \boldsymbol{x} + \epsilon \cdot \mathrm{sign}(\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}))\,, \tag{3}$$

c) PGD: This is an advanced and iterative method, which involves refining the $\boldsymbol{x}_{adv}$ at each iteration by adjusting the perturbation in the direction that maximizes the loss $\mathcal{L}(\cdot)$ while staying within the ϵ-neighbourhood of the clean signal. The mathematical formulation is given by

$$\boldsymbol{x}_0 = \boldsymbol{x} \tag{4}$$

$$\boldsymbol{x}_{i+1} = \mathrm{clip}_{[\boldsymbol{x},\epsilon]}\{\boldsymbol{x}_i + \beta \cdot \mathrm{sign}(\nabla_{\boldsymbol{x}_i} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{y})\} \tag{5}$$

$$\boldsymbol{x}_{adv} = \boldsymbol{x}_T\,, \tag{6}$$

where β is a step size, T is the number of iterations and $\mathrm{clip}_{[\boldsymbol{x},\epsilon]}\{\boldsymbol{x}_i\}$ denotes constraining the intermediate sample $\boldsymbol{x}_i$ in the range $[\boldsymbol{x}_i - \epsilon, \boldsymbol{x}_i + \epsilon]$.

d) Deepfool: This is an iterative attack originally developed for binary classifiers. It is based on the idea that the minimum perturbation required for the misclassification of an input sample will be the orthogonal projection of the sample

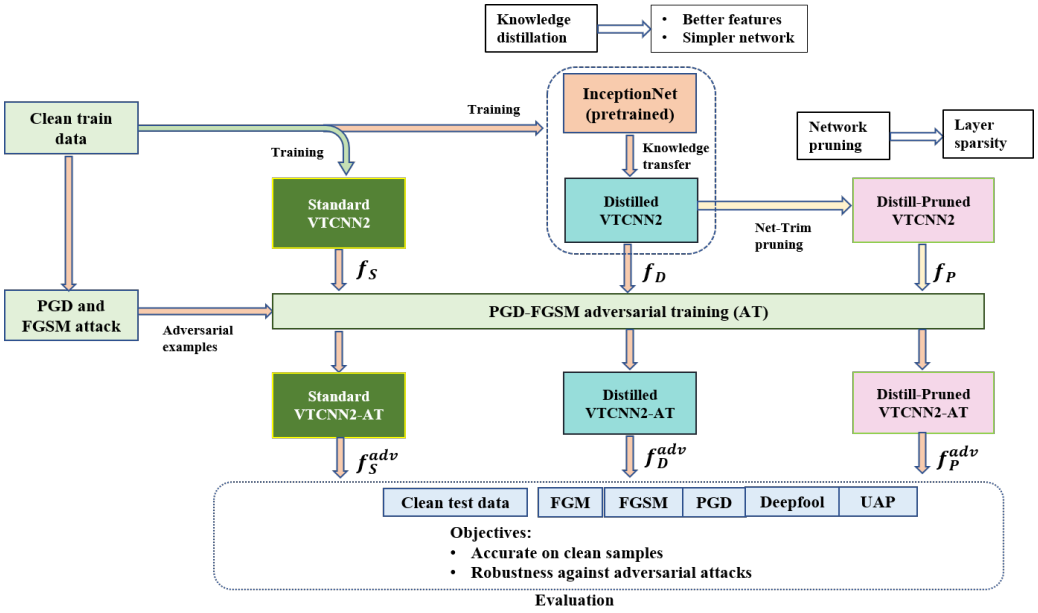


Fig. 1: Taxonomy for the evaluation of robustness against adversarial attacks of the proposed optimized models.

onto the decision boundary. The iterative optimization problem to generate the minimum perturbation, $\boldsymbol{\delta}_i$ is given by

$$\arg\min_{\boldsymbol{\delta}_i} ||\boldsymbol{\delta}_i||_2, \quad \text{s.t.} \quad f(\boldsymbol{x}_i) + \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}_i)^{\mathrm{T}} \boldsymbol{\delta}_i = 0, \qquad (7)$$

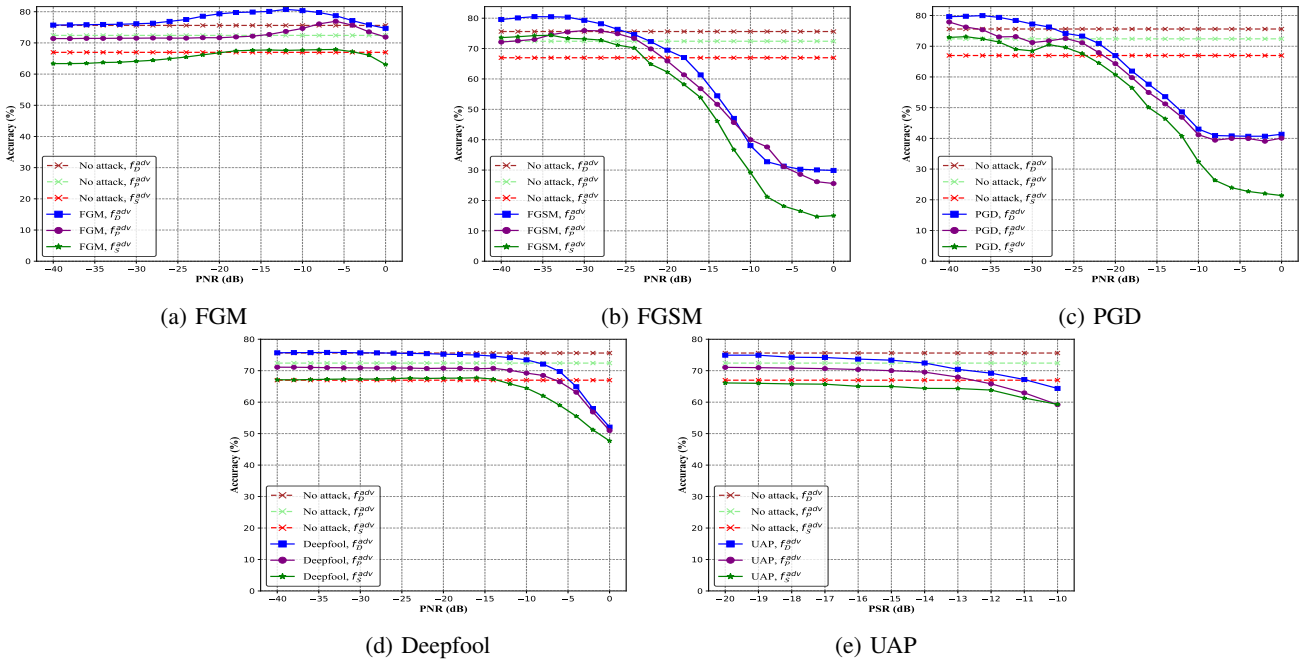
e) UAP: UAP is a method to generate adversarial perturbations that are input-agnostic and do not depend on the knowledge of the DL model; thus, these perturbations are universal in nature. In this work, we have used the PCA-based UAP method proposed in [10], as it is computationally efficient.

IV. PGD-FGSM Adversarial Training

Adversarial examples are generated by maximizing the loss function of a DL model as formulated in (1). Therefore, AT exposes the DL model to these samples by augmenting the clean training data and then aim to minimize the classification loss through standard training, which will enhance the robustness of the model against adversarial attacks. A key objective of this work is to reduce the cost of AT so that it can be performed locally on an edge device, which will enhance the privacy and security of the application on demand. However, most of the existing work considers incorporating adversarial samples generated from iterative or multi-step attacks as they have a high attack success rate, but it also increases the computational cost of AT significantly. In contrast, we consider incorporating examples from both single-step and multi-step attacks. This will help in reducing the computational complexity of AT because the cost of generating N examples from a multi-step attack is significantly higher than the combined cost of generating N_1 examples from the multi-step attack and N_2 examples from another single-step attack, where $N_1 + N_2 = N$ and $N_1 = N_2$. Also, incorporating adversarial examples generated in different gradient directions can effectively improve the robustness of the models. Therefore, this work utilizes both PGD and FGSM attacks to generate adversarial examples during the AT process. To achieve more computational efficiency, we have also fixed the weights of the first FC layer of the standard model $f_{\mathcal{S}}$, the distilled model $f_{\mathcal{D}}$, and the distill-pruned model $f_{\mathcal{P}}$, while performing AT. This reduces the number of trainable parameters to around 126K compared to the original 2.83M parameters. The corresponding adversarially trained models are denoted as $f_{\mathcal{S}}^{adv}$, $f_{\mathcal{D}}^{adv}$, and $f_{\mathcal{P}}^{adv}$. Fig. 1 shows the taxonomy for developing the proposed robust, optimized models and the evaluation against adversarial attacks.

V. Results and Discussion

In this work, for the demonstration purpose of the proposed techniques, we have considered the well-known RML2016.10A RF modulation classification dataset [2]. The dataset consists of $220,000$ RF signals from 11 modulation schemes. For each modulation, the signals are generated using signal-to-noise ratios (SNRs) in the range of -20 dB to 18 dB with a step size of 2 dB. Each complex-valued signal is of dimension 2×128, where both the I and Q components contain 128 samples each. We have chosen this dataset because it is publicly available, which enables the reproducibility of our results. We have considered 50% of the dataset as the training set, denoted as $\mathcal{D}_{Train}$, which is used to develop the models $f_{\mathcal{S}}$, $f_{\mathcal{D}}$, and $f_{\mathcal{P}}$. The remaining 50%, denoted as $\mathcal{D}_{Test}$, is used to evaluate the proposed defense method. The AT process is performed on the three models using the examples generated from $\mathcal{D}_{Train}$. The performance of the models $f_{\mathcal{S}}^{adv}$, $f_{\mathcal{D}}^{adv}$, and $f_{\mathcal{P}}^{adv}$ are evaluated in terms of i) robustness performance for adversarial test samples generated from five WB attacks, namely FGM, PGD, FGSM, Deepfool, and UAP, ii) classification performance of the models on the clean test samples. Further, we define two quantities in the context of adversarial attacks: the perturbation-to-noise ratio (PNR) and the perturbation-to-signal ratio (PSR). PNR is the



(a) FGM (b) FGSM (c) PGD

(d) Deepfool (e) UAP

Fig. 2: Classification accuracy of the adversarially trained standard and optimized models for adversarial attacks at SNR=10 dB.

relation between the perturbation power and the noise power defined as $\text{PNR} = \epsilon^2 \frac{(\text{SNR}+1)}{||\boldsymbol{x}||_2^2}$, where $||\boldsymbol{x}||_2^2$ is the signal power and ϵ is the maximum allowed perturbation. PSR is the relation between the perturbation power and the signal power defined as $\text{PSR} = \text{PNR}/\text{SNR}$.

a) Robustness against WB attacks: In Fig. 2, we have compared the robustness performance of $f_{\mathcal{D}}^{adv}$, $f_{\mathcal{P}}^{adv}$, and $f_{\mathcal{S}}^{adv}$ when tested against the five representative adversarial attacks, at a fixed SNR=10 dB. Specifically, we have evaluated the proposed models against both single-step attacks, i.e., FGM FGSM, and UAP, as well as multi-step (iterative) attacks, i.e. PGD and Deepfool. It can be observed that $f_{\mathcal{D}}^{adv}$ and $f_{\mathcal{P}}^{adv}$ overall performs better than the $f_{\mathcal{S}}^{adv}$ across all the attacks. Fig. 2a shows that for FGM attack, $f_{\mathcal{D}}^{adv}$ performs significantly better than $f_{\mathcal{S}}^{adv}$, with an average accuracy gain of 12% across all PNRs. The accuracy of $f_{\mathcal{P}}^{adv}$ is comparable to $f_{\mathcal{D}}^{adv}$ at higher PNRs for FGM attack. It can also be observed for FGSM and PGD attacks in Fig. 2b and Fig. 2c, respectively, that both $f_{\mathcal{D}}^{adv}$ and $f_{\mathcal{P}}^{adv}$ performs significantly better than $f_{\mathcal{S}}^{adv}$ at the high PNR values. For example, both $f_{\mathcal{D}}^{adv}$ and $f_{\mathcal{P}}^{adv}$ achieve an accuracy gain of around 20% compared to $f_{\mathcal{S}}^{adv}$ when evaluated for PGD attack at PNR = 0 dB. Similarly, for FGSM attack, $f_{\mathcal{D}}^{adv}$ and $f_{\mathcal{P}}^{adv}$ achieve 15% and 12% higher accuracies, respectively, at PNR = 0 dB.

These results signify that a DL model that is optimized using KD can achieve improved adversarial robustness than the standard model when both are subjected to a computationally efficient AT process. We have also observed that pruning the distilled model can still provide better adversarial robustness than the standard model. This is beneficial for edge applications as we can achieve both high sparsity and robustness simultaneously with the distilled-pruned model. Our evaluation

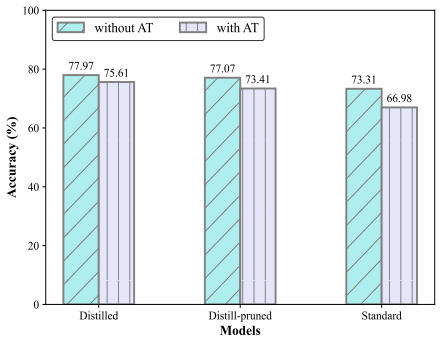


Fig. 3: Classification accuracy of the models on clean samples with and without AT at SNR=10 dB.

also proves that performing AT with the combination of a single-step attack (FGSM) and an iterative attack (PGD) can be computationally efficient as this also achieves robustness against unseen single and multi-step attacks, such as FGM, Deepfool, and UAP, without incorporating any adversarial samples during the AT process.

b) Performance on clean samples: We have compared the classification performance of the optimized and the standard models for the clean samples when evaluated with and without AT at SNR=10 dB, as shown in Fig. 3. The AT process can lead to a reduction in accuracy for the clean samples, i.e., the clean accuracy, when compared to the model without AT. The reduction in clean accuracy affects the reliability of the model for classifying received signals without any attack. Therefore, it is important to minimize the decrease in the clean accuracy. It can be observed in Fig. 3 that the drop in clean accuracy after AT is the lowest for $f_{\mathcal{D}}^{adv}$ and the highest for $f_{\mathcal{S}}^{adv}$. Specifically, the clean accuracies of $f_{\mathcal{D}}$, $f_{\mathcal{P}}$, and $f_{\mathcal{S}}$ (models before AT) are 77.97%, 77.07%, and

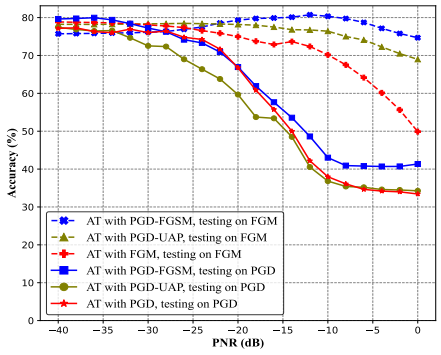


Fig. 4: Classification accuracy of $f_{\mathcal{D}}^{adv}$ for the PGD and FGM attacks when trained with different AT methods at SNR=10 dB.

73.31%, respectively. After performing AT, the accuracies of the models $f_{\mathcal{D}}^{adv}$, $f_{\mathcal{P}}^{adv}$, and $f_{\mathcal{S}}^{adv}$ are 75.61%, 73.41%, and 66.98%, respectively. It can be observed that both distilled and distill-pruned models have higher accuracies on clean samples than the standard model, before as well as after AT. This shows that KD also helps the lightweight model to learn more robust features and is not affected significantly by AT.

To emphasize the choice of FGSM for AT, we have also analyzed the robustness of $f_{\mathcal{D}}^{adv}$ when UAP along with PGD is used for AT. UAP is also a single-step attack and, thus, can provide comparable computational benefits for AT. Fig. 4 shows a comparison of the accuracies of $f_{\mathcal{D}}^{adv}$ for the PGD and FGM attacks with different AT methods at SNR=10 dB. For the PGD attack, it can be observed that AT with PGD and FGSM performs the best and is even better than AT with only PGD samples. Similarly, for the FGM attack, AT with PGD and FGSM achieves the highest robustness and is significantly higher than the model trained with only FGM samples. Thus, using a combination of multi and single-step attacks for AT can significantly increase the robustness of the model for FGM attack. Increasing the robustness of a DL model against the FGM attack is especially relevant for wireless communication applications as it takes into account the perturbation power (L_2-norm).

VI. Conclusion

In this work, we proposed two DL-based optimized models for AMC, namely distilled and distill-pruned models, based on knowledge distillation and network pruning. The primary objective of the proposed approach is to enhance the robustness of the optimized models against adversarial attacks for secure deployment in edge devices. To achieve this, we performed adversarial training with PGD and FGSM samples on the optimized models in a computationally efficient manner. Further, we investigated the robustness of these models using five adversarial attacks: FGM, FGSM, PGD, Deepfool, and UAP. Experimental results have shown that the optimized models can achieve better robustness than the standard model, with the distilled model achieving the maximum robustness across all attacks. We have demonstrated that distillation also helps with minimizing the loss in accuracy for the clean samples for the adversarially trained optimized models. Future work will incorporate developing computationally efficient, retraining-free countermeasure techniques to enable the on-device robustness improvement of DL models.

References

[1] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24 411–24 432, 2018.

[2] T. O'Shea and N. West, "Radio machine learning dataset generation with GNU Radio," in *Proc. GNU Radio Conf.*, Boulder, CO, USA, Sep. 2016.

[3] N. E. West and T. O'shea, "Deep architectures for modulation recognition," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Baltimore, MD, USA, Mar. 6-9, 2017, pp. 1–6.

[4] B. R. Manoj, G. Tian, S. Gunnarsson, F. Tufvesson, and E. G. Larsson, "Sensing and classification using massive MIMO: A tensor decomposition-based approach," *IEEE Wireless Commun. Letts.*, vol. 10, no. 12, pp. 2649–2653, Dec. 2021.

[5] L. Sanguinetti, A. Zappone, and M. Debbah, "Deep learning power allocation in massive MIMO," in *Proc. Asilomar Conf. on Signals, Systs., and Computers*, Pacific Grove, CA, USA, Oct. 2018, pp. 1257–1261.

[6] Y. Cui, F. Liu, X. Jing, and J. Mu, "Integrating sensing and communications for ubiquitous IoT: Applications, trends, and challenges," *IEEE Network*, vol. 35, no. 5, pp. 158–167, 2021.

[7] R. Mishra, H. P. Gupta, and T. Dutta, "A survey on deep neural network compression: Challenges, overview, and solutions," Oct. 2020. [Online]. Available: https://arxiv.org/abs/2010.03954

[8] L. Pajola, L. Pasa, and M. Conti, "Threat is in the air: Machine learning for wireless network applications," in *Proc. of ACM Workshop on Wireless Security and Machine Learning*, New York, NY, USA, 2019, pp. 16–21.

[9] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, 2019.

[10] M. Sadeghi and E. G. Larsson, "Adversarial attacks on deep-learning based radio signal classification," *IEEE Wireless Commun. Letts.*, vol. 8, no. 1, pp. 213–216, Aug. 2018.

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

[12] Z. Zhang *et al.*, "Automatic modulation classification using CNN-LSTM based dual-stream structure," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13 521–13 531, Nov. 2020.

[13] S. Hamidi-Rad and S. Jain, "Mcformer: A transformer based deep neural network for automatic modulation classification," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, Dec. 7-11, 2021, pp. 1–6.

[14] A. Bahramali, M. Nasr, A. Houmansadr, D. Goeckel, and D. Towsley, "Robust adversarial attacks against DNN-based wireless communication systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, 2021, pp. 126–140.

[15] B. R. Manoj, P. M. Santos, M. Sadeghi, and E. G. Larsson, "Toward robust networks against adversarial attacks for radio signal modulation classification," in *Proc. IEEE 23rd Int. Workshop Signal Process. Adv. Wireless Commun.*, Oulu, Finland, 2022, pp. 1–5.

[16] J. Maroto, G. Bovet, and P. Frossard, "SafeAMC: Adversarial training for robust modulation recognition models," 2021. [Online]. Available: https://arxiv.org/abs/2105.13746

[17] Z. Wang, W. Liu, and H.-M. Wang, "GAN against adversarial attacks in radio signal classification," *IEEE Commun. Letts.*, vol. 26, no. 12, pp. 2851–2854, 2022.

[18] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, Salt Lake City, UT, USA, Jun. 2018.

[19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," Mar. 2015. [Online]. Available: https://arxiv.org/abs/1503.02531

[20] L. Beyer *et al.*, "Knowledge distillation: A good teacher is patient and consistent," in *Proc. IEEE Conf. on Comp. Vis. and Patt. Recog. (CVPR)*, New Orleans, LA, USA, Jun. 2022, pp. 10 925–10 934.

[21] A. Aghasi *et al.*, "Net-trim: Convex pruning of deep neural networks with performance guarantee," in *Proc. Adv. Neural. Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, Dec. 4-7, 2017, pp. 3180–3189.