AI/ML + Wireless: Signal Modulation Detection and Classification

1st Libo Zhang
Department of Electrical and Computer Engineering
Duke University
Durham, NC, USA
libo.zhang@duke.edu

1st Yujie Zhang
Department of Electrical and Computer Engineering
Duke University
Durham, NC, USA
yujie.zhang396@duke.edu

Abstract—**Deep Neural Networks (DNNs) have gained popularity in wireless communication systems due to their promising performance. However, their vulnerability is the same as other DNN-based applications. In this work, we re-implement various adversarial attacks for interfering with wireless RF signals during transmission and observe that the performance of well-trained models degrades significantly under attacks. Then, using the paper [SLB21] as a foundation, we construct effective detection and mitigation methods to defend against existing threats. Additionally, we compare various attack-defence approaches in white-box and black-box scenarios, and then optimize the robust model's performance through compression and fine-tuning.**

I. Introduction

Deep Neural Networks (DNNs) have lately been integrated into a variety of wireless communication systems due to their promising categorization performance. Examples of DNN-based wireless communication systems include autoencoder wireless communication [GCHB17], modulation recognition (radio signal classification) [MCWW18], [ORC18], and OFDM channel estimation and signal detection [YLJ18], which all leverage DNNs' high utility to improve the performance of their underlying wireless applications. However, adversarial examples are known to be prone to DNNs, which are minor perturbations given to the inputs of a DNN that cause it to misclassify the perturbed inputs.

In this paper, we examine adversarial examples against wireless systems that rely on DNNs. The attacker's objective in this scenario is to send a well-crafted perturbation signal over the channel, causing the underlying DNN-based wireless system (e.g., a radio signal classifier) to fail to function effectively due to misclassification of the perturbed signals.

II. Related Work

While there is a substantial body of work on adversarial examples versus image classification tasks, e.g., FGSM [GSS15], such work cannot be easily transferred to the setting of wireless systems where the adversary is unaware of the input signals to be disrupted. Recent research [ACA20], [FBH19], [RDAS+20], [SL19a], [SL19b] has examined input-independent adversarial examples for wireless systems by producing universal adversarial perturbations (UAP) [MDFFF17]. Recent efforts [FBH19], [RDAS+20] address this issue by generating perturbations using a DNN model; however, they only utilize a white-box situation in which the adversary is informed of the target wireless DNN model.

TABLE I: Proposed CNN architecture. The shapes of the convolutional layers correspond to $L \times W \times F$.

Layer	Dropout Rate (%)	Activation	Shape
Input	-	-	$2 \times \ell \times 1$
Conv 1	20	ReLU	$2 \times 5 \times 256$
Conv 2	20	ReLU	$1 \times 4 \times 128$
Conv 3	20	ReLU	$1 \times 3 \times 64$
Conv 4	20	ReLU	$1 \times 3 \times 64$
FC 1	-	ReLU	128
Output	-	Softmax	C

TABLE II: Proposed RNN architecture consisting of an LSTM layer with 256 units followed by a fully connected layer with 128 units.

Layer	Activation	Output Shape
Input	-	$2 \times \ell$
LSTM	-	256
FC	ReLU	128
Output	Softmax	10

Fig. 1. Table 1 and Table 2 are derived from paper [SLB21].

III. What we have done - Revalidation Work

A. Prepare and preprocess original RadioML dataset

The authors of paper [SLB21] do not publish their codes, therefore to complete the revalidation work, we decide to build from scratch with PyTorch deep learning (DL) framework, and the first step is to prepare and preprocess the original dataset to obtain appropriate data for machine learning (ML) model training/testing. We decided to utilize the RadioML 2016.10b dataset and to consider only positive signal to noise ratio (SNR) values ranging from 0dB to 18dB, with a step size of 2dB. In specific, we have 600,000 2128 wireless in-phase and quadrature (IQ) data samples, and we leverage 10 modulation schemes as true classification labels, 8 of which belong to digital (D) constellations, and 2 of which belong to analog (A) constellations. The 10 modulations are BPSK (D), QPSK (D), 8PSK (D), QAM16 (D), QAM64 (D), GFSK (D), CPFSK (D), PAM4 (D), WB-FM (A), and AM-DSB (A).

TABLE III: Model training parameters. The CCE cost function refers to categorical cross entropy.

Parameter	CNN	RNN
Cost Function	CCE	CCE
Optimizer Algorithm	Adam	Adam
Learning Rate	0.001	0.001
Batch Size	256	64
Epochs	100	100

Fig. 2. Table 3 is derived from paper [SLB21].

B. Build baseline CNN and RNN classifiers

For the convolutional neural network (CNN) model, based on Table 1 [SLB21], we define 4 convolutional (CONV) layers with 20% dropout rate and ReLU activation respectively. After flattening and going through the first fully-connected (FC) layer with 128 neurons, the Output FC layer has the Softmax activation leading to 1 of 10 modulation schemes classification prediction. In addition, we need to extend the original 2128 IQ data shape to 21281 and regard the additional one dimension as the input channels for our CNN model.

For the recurrent neural network (RNN) model, based on Table 2 [SLB21], we start with 1 long short-term memory (LSTM) layer. Given the 2128 IQ data shape, we set 2 as the sequence length of LSTM, set 128 as the number of expected features for each sequence, and set the hidden size as 256 since it controls the output shape of the LSTM layer. After flattening and going through the first FC layer with 128 neurons, the Output FC layer has the Softmax activation leading to 1 of 10 modulation schemes classification prediction.

For CNN and RNN training hyperparameters, based on Table 3 [SLB21], we utilize the categorical cross-entropy (CEE) loss function and Adam optimizer for both models. We set the training epochs to be 100 and set the batch size to be 256 for CNN, and 64 for RNN.

In summary, the proposed CNN model has totally 1,130,890 trainable parameters, while the proposed RNN model has totally 429,450 trainable parameters, which means that the proposed RNN model has explicitly less model complexity (or much shallower model architecture) compared with the proposed CNN model.

For experimental results, based on Figure 3 and 4, without data subsampling, our revalidated CNN can achieve 90.78% average testing accuracy among different SNRs, while our revalidated RNN can achieve 74.43% average testing accuracy. We come up with two intuitions trying to explain why the RNN model has relatively poor performance compared with the CNN model. Firstly, the RNN model has much fewer trainable parameters compared with the CNN model, which clearly indicates that the RNN has at least less potential to achieve as good performance as the CNN. Secondly, to successfully feed the IQ samples into the LSTM layer, we need to set 2 as the sequence length. Recall that an LSTM layer contains the input gate, the forget gate, and the output gate. Given a sequence of length 2, what to remember or what to forget does not make much sense, at least in terms of wireless IQ data samples.

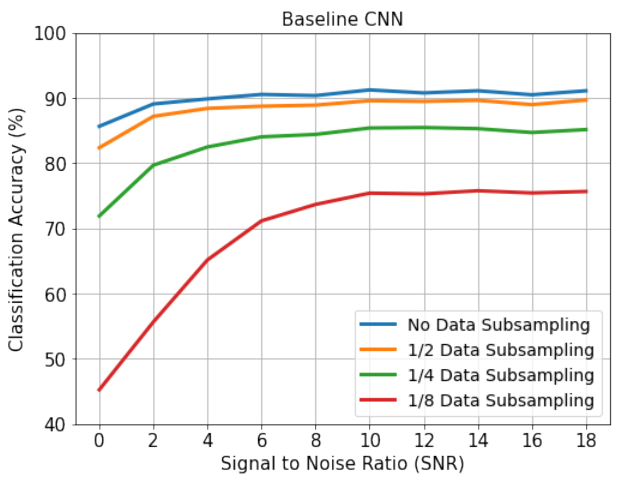

Fig. 3. Accuracy of baseline CNN model with various subsampling.

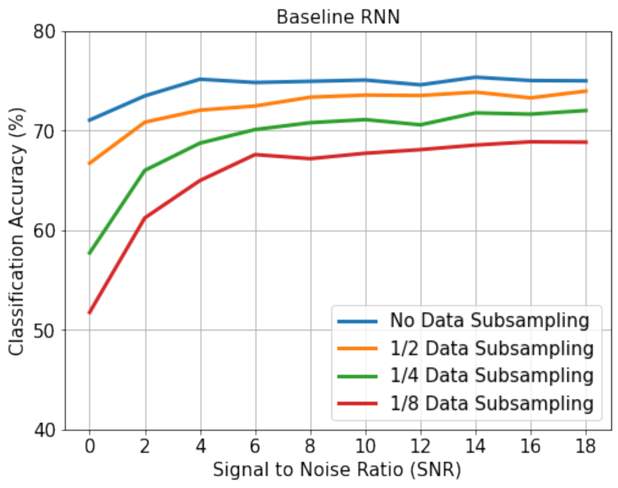

Fig. 4. Accuracy of baseline RNN model with various subsampling.

Furthermore, we also plot the confusion matrices for the baseline CNN and RNN classifiers to look more carefully into their modulation classification performance. Based on Figure 5 and Figure 6, we can see that both models encounter some difficulty in accurately classifying QAM16 (D) between QAM64 (D), as well as telling the difference between AM-DSB (A) and WB-FM (A).

C. Apply data subsampling (compression)

To apply data subsampling, although we cannot manipulate the dimension (2) which represents IQ patterns, we can subsample the input size (128) to compress data, we evenly slice the 128 features to formulate the compressed dataset. For example, if we want 1/2 subsampling rate, we evenly extract features of indices 0, 2, 4, 6, 8... and finally the compressed IQ samples has a data shape of 264. Similarly, the 1/4 subsampling rate results in a shape of 232, and the 1/8 subsampling rate leads to a shape of 216.

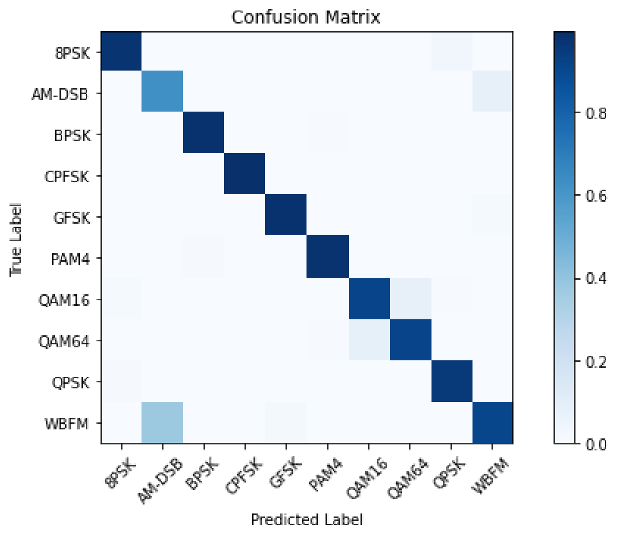

Fig. 5. Baseline CNN model confusion matrix.

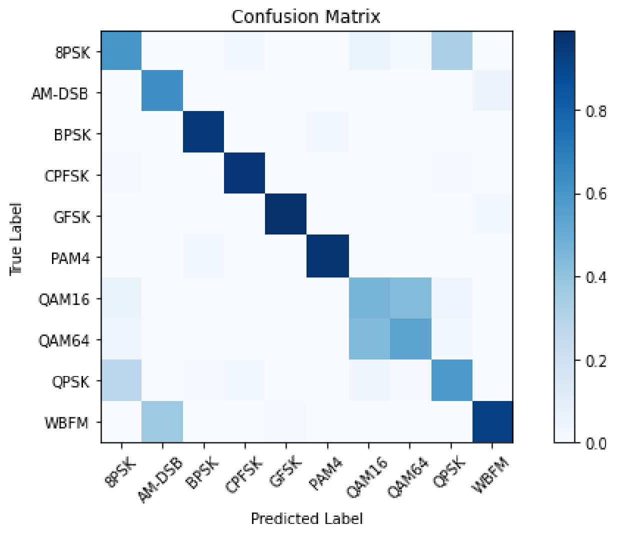

Fig. 6. Baseline RNN model confusion matrix.

For the CNN model, based on Figure 3, we can see that a subsampling rate of 1/2 or 1/4 would not significantly jeopardize the classification accuracy of CNN, but a subsampling rate of 1/8 would impose very negative effects on CNN's performance.

For the RNN model, based on Figure 4, we can see that data subsampling does jeopardize the classification accuracy, but such negative effects are not that significant compared with CNN, which could be explained by the relatively poor performance of the baseline RNN without any subsampling.

In summary, we can notice that the CNN with 1/8 subsampling can even achieve similar performance compared with the baseline RNN without subsampling. Therefore, we will focus on the CNN model for the following tasks and exclude the RNN model from consideration.

D. Implement adversarial attacks (L-infinity and L-2)

Adversarial interference introduced into transmitted signals has an ability to impact the DNN model $f(\cdot;\theta)$'s classification result, where θ denotes the parameters of the model. The authors consider perturbations constructed using the fast gradient sign method (FGSM) under both an L_∞-norm and L_2-norm constraint in the paper [SLB21]. They exploit the adversary's most vulnerable vulnerability, referred to as a white box threat model, in which the adversary has complete access to the trained model and its parameters. To be precise, the attacker is fully aware of $f(\cdot;\theta)$ architecture and parameters.

1) L_∞-bounded attack: The L_∞-bounded FGSM adds a small perturbation, $\epsilon \in \mathbf{R}^{l\times 2}$, to each feature of the input sample in the direction of the sign of the classifier's cost function (categorical cross entropy in this case), $J(x, y, \theta)$, which is a function of the input sample, x, its ground truth label, y, and the model parameters, θ. Formally, the L_∞ crafted perturbation is given by equation (1).

$$x' = x + \epsilon \cdot sgn(\nabla_x J(x, y, \theta)), \epsilon \in \mathbf{R}^{l\times 2}. \tag{1}$$

Employing (1) moves the sample, x, in the direction of the high-parameter neural network's decision boundary, and misclassification is induced when the sample crosses the decision boundary. The adversary here is not limited to adding an imperceptible perturbation, and therefore, the ϵ bound added to the sample can vary widely.

2) L_2-bounded attack: The L_2-bounded FGSM is natural to consider for wireless signals as it corresponds to the signal power of the transmission. Specifically, a perturbation, $\alpha \in \mathbf{R}^{l\times 2}$, is added to the signal, x, by equation (2)

$$x' = x + \alpha \cdot \frac{\nabla_x J(x, y, \theta)}{\|\nabla_x J(x, y, \theta)\|_2}, \alpha \in \mathbf{R}^{l\times 2}. \tag{2}$$

E. FGSM Adversarial Detection and Mitigation

1) Mitigation: The mitigation portion is concerned with correctly classifying adversarially perturbed inputs. To achieve this, we re-train $f(\cdot;\theta)$ using adversarial inputs generated on $\mathbf{X}_{train}$ using (1) for different ϵ bounds and arrive at $F(\cdot;\Theta)$. $F(\cdot;\Theta)$ can withstand adversarial perturbations to a greater extent than $f(\cdot;\theta)$ due to its augmented training set, which includes inputs artificially injected with adversarial interference.

2) Detection: For detection, we make use a convolutional autoencoder, which we denote as $U(\cdot;\Phi)$, where Φ represents the parameters of the autoencoder. We use an uncoding function, $h : \mathbf{R}^{l\times 2} \longrightarrow \mathbf{R}^{k\times 2}$ to map an input signal x_i to a latent space representation. Then, a decoding function, $g : \mathbf{R}^{k\times 2} \longrightarrow \mathbf{R}^{l\times 2}$, is used to reconstruct an approximation of the input. Intuitively, the reconstruction cost will be higher when an adversary has injected noise into the input.

The parameters of the encoder and decoder are simultanuously optimized, using the mean squared error functions, to produce the autoencoder given by equation (3).

$$U(\cdot;\Phi) = minimize\|\frac{1}{N}\sum_{x_i \in \mathbf{X}_{train}}(x_i - g(h(x_i))\|^2. \tag{3}$$

TABLE IV: Convolutional autoencoder architecture. The shapes of the convolutional layers correspond to $L \times W \times F$.

Layer	Activation	Shape
Input	-	$2 \times \ell \times 1$
Conv 1	Linear	$3 \times 3 \times 64$
Conv 2	Linear	$3 \times 3 \times 32$
Conv 3	Linear	$3 \times 3 \times 16$
Conv 4	Linear	$3 \times 3 \times 32$
Conv 5	Linear	$3 \times 3 \times 64$
Conv 6	Linear	$3 \times 3 \times 1$

Fig. 7. Table 4 is derived from paper [SLB21].

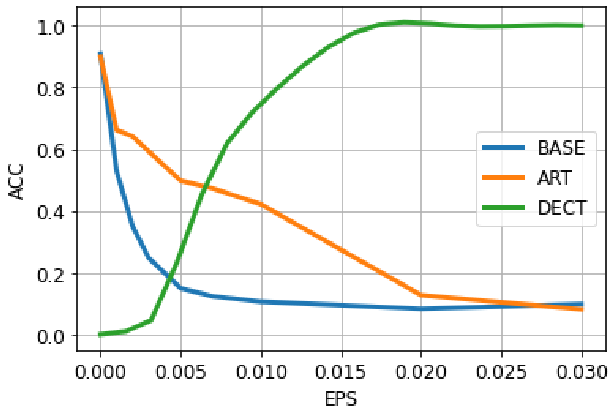

Fig. 8. Mitigation and detection effectiveness on the CNN (no sampling) against L_∞-bounded perturbation.

The autoencoder model architecture is shown in Table 4. Specifically, the reconstruction loss of $U(\cdot; \Phi)$ is used to measure the distance of a sample from the training data manifold, where samples beyond a threshold, T, are considered adversarial.

F. FGSM attack and defense performance analysis

Assuming a white box threat model, we implemented two types (L_∞ norm and L_2 norm) of fast gradient sign method (FGSM) attacks against our DL classifiers and evaluated the attack effectiveness under different sub-sampling rates.

1) L_∞-bounded attack: As shown in Figure 8, for $\epsilon < 0.01$, our re-train model (orange line) is able to obtain significant improvements in classification performance over base model (blue line) when under attack, while for $\epsilon > 0.01$, our implemented autoencoder (green line) has a high detection rate.

Figure 9 shows the defense effectiveness when applying 1/2 sub-sampling. The trend in no sampling is consistent over 1/2 sub-sampling rates.

2) L_2-bounded attack: As shown in Figure 10, our autoencoder again achieves nearly 100% detection at high perturbation bounds while the re-train model increases the classification rate at low perturbation bounds.

Figure 11 shows the defense effectiveness when applying 1/2 sub-sampling.

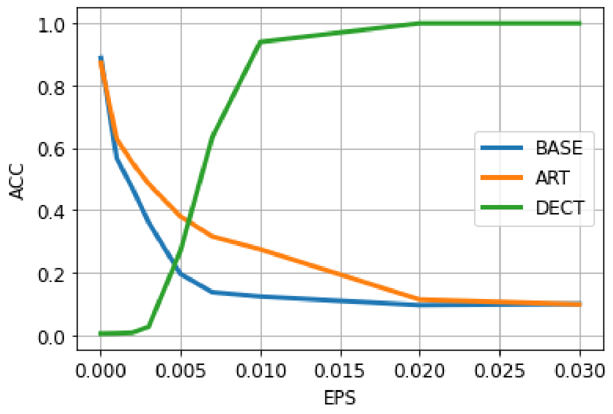

Fig. 9. Mitigation and detection effectiveness on the CNN (1/2 sampling) against L_∞-bounded perturbation.

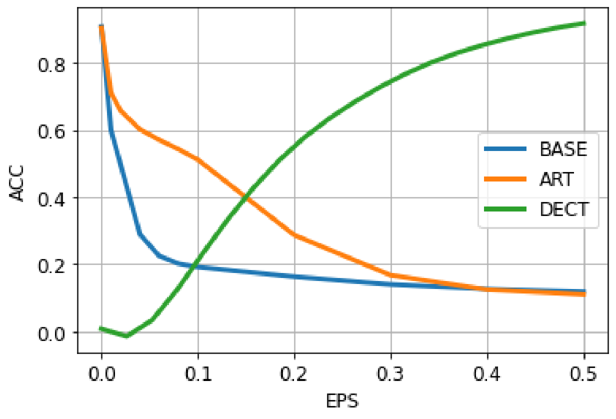

Fig. 10. Mitigation and detection effectiveness on the CNN (no sampling) against L_2-bounded perturbation.

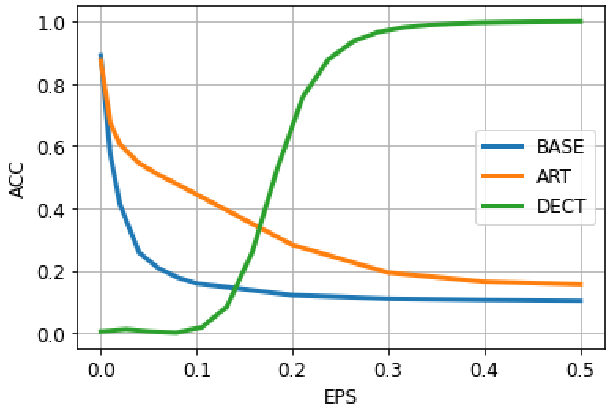

Fig. 11. Mitigation and detection effectiveness on the CNN (1/2 sampling) against L_2-bounded perturbation.

IV. What we have done - Challenge beyond Revalidation

A. Visualize adversarially perturbed data

In order to better illustrate the signal distribution after attacks, we plot the constellation figures for one example of GFSK modulation data, which are shown in the Figure 12. As the attack strength ϵ grows up, both the FGSM L_∞ attack and the L_2 attack cause the poisoned data to deviate more from the clean data. However, when $\epsilon < 0.03$ for L_∞ attack and $\epsilon < 0.04$ for L_2 attack, it is still hard to distinguish whether a data is poisoned or clean in traditional way, but the test accuracy has already significantly reduced. Therefore, without any mitigation and detection methods, FGSM attacks can perform well in cheating traditional detection mechanisms.

B. Implement projected gradient descent (PGD) attack

The FGSM attack might have gradient masking problem, which means the local minima of loss function near the data point could cause the FGSM direction to contradict the direction of decision boundary, and therefore the attacker would fail to distort the target model. The gradient masking problem of FGSM motivates us to implement another type of adversarial attack, the projected gradient descent (PGD) attack.

$$x'_0 \sim Uniform(x - \epsilon, x + \epsilon). \tag{4}$$

$$x'_{N+1} = clip_{x,\epsilon}\{x'_N + \beta \cdot sign(\nabla_x J(x'_N, y, \theta)\}. \tag{5}$$

Based on equation (4) and equation (5), PGD attack starts from a random point around the data point (a circle with the radius as the perturbation strength), and applies iterative update towards the gradient sign direction, with clipping after each iteration to enforce lower and upper perturbation bounds. In general, we can also claim that the FGSM attack is a special case (the weakest form) of PGD attack, in which there is no random start, there is only 1 iteration, and the perturbation strength ϵ is equal to the iteration step size β.

For experimental results, based on Figure 13, when applying 3 different attacks to the baseline CNN without subsampling, we can see that the PGD attack can achieve the most effective attacking/distortion, which demonstrates our analysis and implementation.

C. Implement PGD-based adversarial retraining

Based on discussion in the previous PGD attack implementation task (Task B), we know that PGD attack is a universal adversary among first-order (gradient based) attacking approaches. Theoretically, if we can achieve good robustness against PGD attack, then we can also achieve good robustness against all first-order attacks (including but not limited to FGSM L-infinity and FGM L-2 attacks) within a certain perturbation constraint. Keeping this motivation in mind, we implement the proposed mitigation algorithm [SLB21] again, by means of PGD-based adversarial retraining.

For experimental results, based on Figure 14 and Figure 15, we can indeed achieve some mitigation with PGD-based adversarial retraining, but it seems that we are still far away from achieving good robustness against FGSM and PGD attacks.

D. Compare whitebox attacks and blackbox attacks

The original paper [SLB21] only applies whitebox attacks, which assume that attackers have full access to the target model including model architecture and weight parameters. To challenge beyond revalidation, we also want to evaluate the attacking effectiveness of blackbox attacks, in which attackers only have query access to the target model. In specific, we propose a new CNN model as the blackbox, and its architecture is shown in Figure 16. Compared with the old CNN model in Table 1 [SLB21], we can see that the new CNN has gradually increasing output channels for the 4 CONV layers, which means the data pipeline is getting thicker instead of thinner as the old CNN. In addition, the new CNN has one more FC layer with 64 neurons before the Output layer. We train the new CNN with exactly the same clean dataset and hyperparameters as the old CNN, and obtain 90.01% average classification accuracy, similar to 90.78% average classification accuracy of the old CNN. Having prepared for all these prerequisites, now we can implement blackbox attacks by means of transfer attacks. In specific, firstly, we generate the adversarially perturbed data from the old CNN (whitebox), this is achievable because the attacker can calculate the gradients with respect to data given the whitebox model architecture and weights. Secondly, the attacker feeds (query access) the adversarially perturbed data to the new CNN (blackbox) to complete blackbox attacks.

For experimental results, based on Figure 17, we can see that the attacking effectiveness of blackbox attacks is a bit weaker than whitebox attacks, but the blackbox PGD attack can still heavily distort the target model even with relatively small attacking strength, which could be good news from the attacker's perspective, but definitely very bad news from the perspective of defender or model owner.

E. Implement model compression

The original paper [SLB21] only applies subsampling to compress data. How will the classification accuracy be affected by model compression? To challenge beyond revalidation, we decide to compress our CNN model by pruning weights to increase model sparsity. In general, larger model weights tend to have more significant effects on the activation, and therefore they are more important.

For specific methods, firstly, we take the absolute values (magnitude) of all weight parameters and sort them from the smallest to the largest. Secondly, we define a desired model sparsity (pruning percentage). Thirdly, if we want 50% sparsity for example, then we directly prune the smaller half weights to 0s. In other words, we are trying to identify and remove "unimportant" elements in our CNN model to meet the compression and sparsity requirements.

For experimental results, based on Figure 18, our CNN model can still maintain approximately 90% testing accuracy even with 50% model sparsity. However, the classification

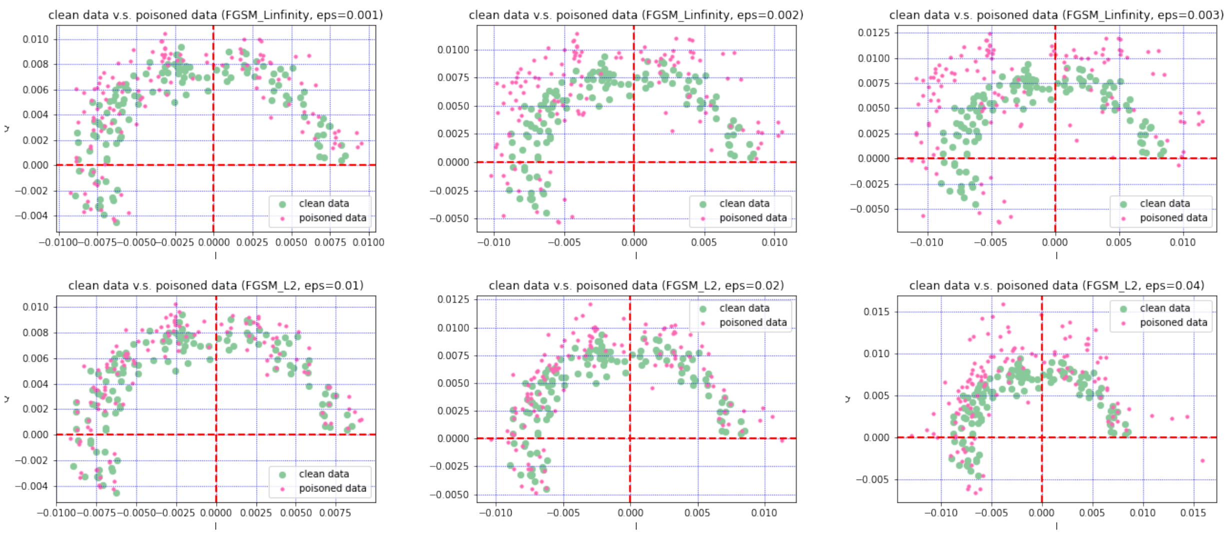

Fig. 12. First row: Constellation figures after FGSM L_∞ attack (GFSK); the test accuracy after attack are respectively 0.53, 0.35, and 0.25. Second row: Constellation figures after FGSM L_2 attack (GFSK); the test accuracy after attack are respectively 0.60, 0.50, and 0.29.

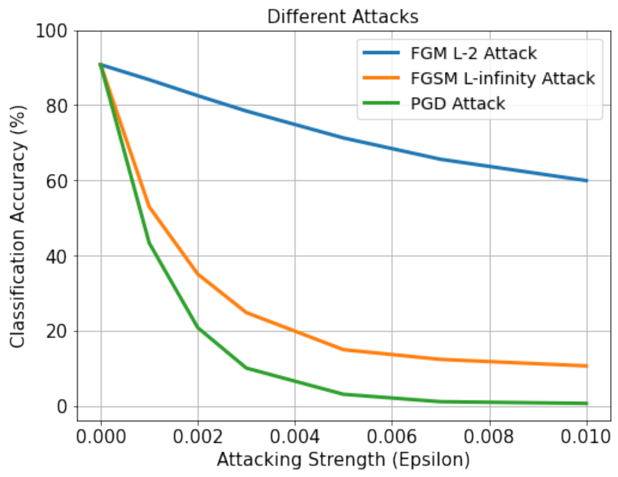

Fig. 13. Accuracy when applying 3 different attacks (FGM L_2, FGSM L_∞, and PGD attacks) to the baseline CNN without subsampling.

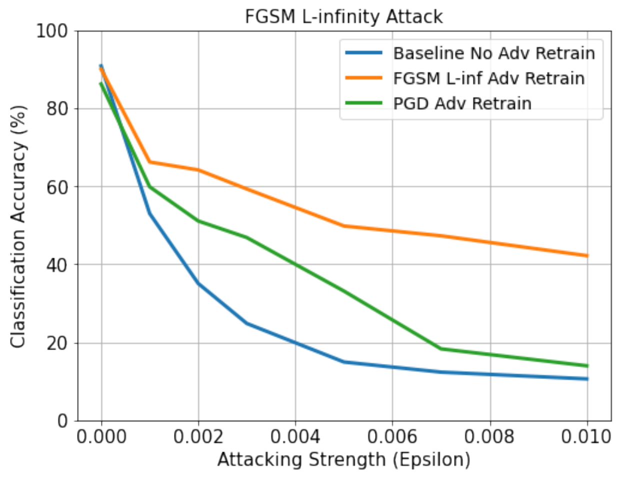

Fig. 14. Accuracy after applying FGSM L_∞ retraining under 3 different scenarios (Baseline, FGSM L_∞, and PGD attacks).

accuracy significantly decreases once the model sparsity goes beyond 60%, and finally degrades to nearly random guessing (10%) with 90% sparsity. The results of model compression actually motivate us to explore another question: can we fine-tune the highly compressed model to improve its classification performance at least to some extent?

F. Finetune model with high sparsity

The proposed question at the end of model compression task (Task E) motivates us to explore model finetuning, which can also be expressed as model retraining with sparse constraints. In specific, before retraining, we create a binary mask to locate large weight parameters for all layers of our CNN model (0 for pruning, 1 for not pruning) and compress the baseline CNN to induce high sparsity. Subsequently, we retrain the compressed model, and the high sparsity of model weights will be destroyed during backpropagation. Therefore, after each retraining epoch, we need to apply the binary mask to all layers' weights to restore high sparsity. In other words, given 90% sparsity as an example, we only finetune/retrain the top 10% "important" large weights in our CNN model, while leaving the rest 90% "unimportant" small weights as 0s all the time.

For experimental results, we implement model finetuning starting with at least 40% sparsity constraint in that smaller sparsity constraints do not explicitly distort the classification accuracy. Based on Figure 18, after finetuning for at most 30 epochs, we can achieve approximately 85% testing classification accuracy even with 90% model sparsity. This demonstrates that significant performance improvement can be accomplished if we finetune the highly compressed CNN model appropriately.

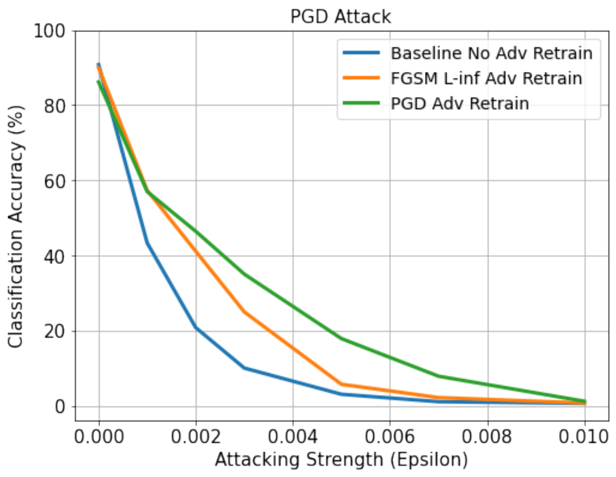

Fig. 15. Accuracy after applying PGD retraining under 3 different scenarios (Baseline, FGSM L_∞, and PGD attacks).

Layer	Dropout Rate (%)	Activation	Shape
Input	-	-	$2 \times l \times 1$
Conv 1	20	ReLU	$2 \times 5 \times 64$
Conv 2	20	ReLU	$1 \times 4 \times 64$
Conv 3	20	ReLU	$1 \times 3 \times 128$
Conv 4	20	ReLU	$1 \times 3 \times 256$
FC 1	-	ReLU	128
FC 2	-	ReLU	64
Output	-	Softmax	C

Fig. 16. CNN architecture for black-box attack.

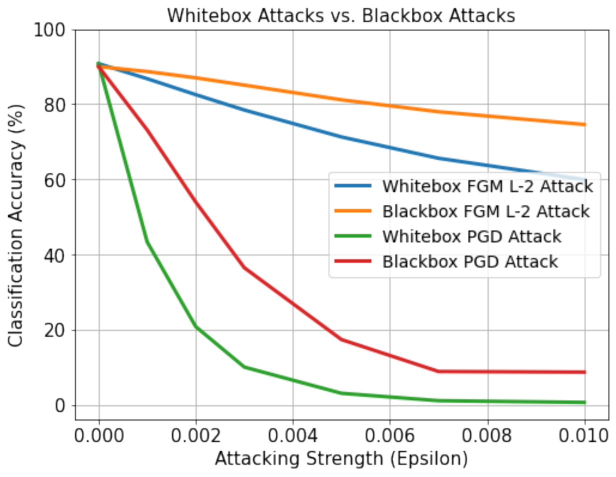

Fig. 17. Comparison of white-box attack and black-box attack.

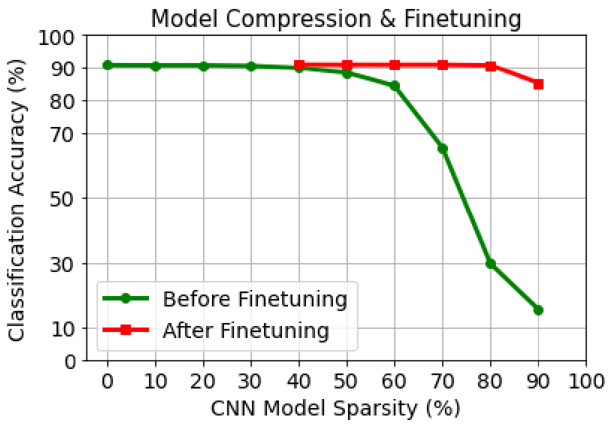

Fig. 18. Accuracy after applying model compression and finetuning.

V. Conclusions and Future Work Discussion

In this project, we completed revalidation and reproduced experimental results based on paper [SLB21]. We started with data preparation and preprocessing. We built and trained baseline CNN and RNN classifiers, and then we evaluated data subsampling and classification accuracy trade-off. We implemented FGSM L-infinity attack and FGM L-2 attack against our CNN model, while also taking data subsampling into consideration. We implemented the proposed mitigation and detection algorithms and evaluated the performances of the adversarial mitigator and detector. Therefore, we can conclude that our revalidation work is successful.

In addition to revalidation work, we also challenged beyond the "Advanced" requirements by implementing new algorithms and conducting exploratory experiments. We visualized the difference between the clean data constellation and the adversarially attacked data constellation. We implemented PGD attack against our baseline CNN and revalidated the proposed mitigation algorithm with PGD-based adversarial retraining. We implemented blackbox attacks by means of transfer attacks and compared the attacking effectiveness between whitebox attacks and blackbox attacks. We evaluated model compression and classification accuracy trade-off and implemented model finetuning to improve classification performance under highly sparse constraints. For each task, apart from presenting our experimental results, we also provided demonstrable explanations and reasonable insights to illustrate what we were thinking when conducting these experiments.

As for future work discussion, firstly, we could not achieve very good robustness against adversarial attacks even if we have successfully revalidated the proposed mitigation algorithm in paper [SLB21] (we would assume perhaps this is why the algorithm is named "mitigation" instead of "defending"). Therefore, in the future, maybe we can try to modify the mitigation algorithm and explore more adversarial training and retraining methods to improve modulation detection/classification robustness against adversarial attacks. Secondly, we know that adversarial attacks would always try to heavily distort a ML/DL model, while keeping the injected perturbation as imperceptible by humans as possible. Jumping out of machine learning, deep learning, and automatic modulation classification, we can also explore how adversarial attacks with relatively small perturbation strengths affect signal transmission quality, if we want to receive the signals and generate decoding results on the receiver side. Our exploration in the future shall evaluate and demonstrate how the imperceptible attacks would affect/distort/ruin the actual decoding bits on the receiver side.

Last but not least, we have uploaded our work for this project on GitHub and Google Drive, in which there are codes we have written and a large number of model checkpoints we have trained to conduct different experiments. Feel free to have a look and play around!

GitHub: https://github.com/baina23/Robust-AMC

Google Drive Folder: https://drive.google.com/drive/folders/14ovsEO4oVs_uLPwlm9lHhE7jHpChzZt3?usp=sharing

References

[ACA20] Abdullatif Albaseer, Bekir Sait Ciftler, and Mohamed M. Abdallah. Performance evaluation of physical attacks against e2e autoencoder over rayleigh fading channel. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 177–182, 2020.

[FBH19] Bryse Flowers, R. Michael Buehrer, and William C. Headley. Communications aware adversarial residual networks for over the air evasion attacks. In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, pages 133–140, 2019.

[GCHB17] Tobias Gruber, Sebastian Cammerer, Jakob Hoydis, and Stephan ten Brink. On deep learning-based channel decoding. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2017.

[GSS15] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[MCWW18] Fan Meng, Peng Chen, Lenan Wu, and Xianbin Wang. Automatic modulation classification: A deep learning enabled approach. *IEEE Transactions on Vehicular Technology*, 67(11):10760–10772, 2018.

[MDFFF17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94, 2017.

[ORC18] Timothy James O'Shea, Tamoghna Roy, and T. Charles Clancy. Over-the-air deep learning based radio signal classification. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):168–179, 2018.

[RDAS$^+$20] Francesco Restuccia, Salvatore D'Oro, Amani Al-Shawabka, Bruno Costa Rendon, Kaushik Chowdhury, Stratis Ioannidis, and Tommaso Melodia. Generalized wireless adversarial deep learning. In *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*, WiseML '20, page 49–54, New York, NY, USA, 2020. Association for Computing Machinery.

[SL19a] Meysam Sadeghi and Erik G. Larsson. Adversarial attacks on deep-learning based radio signal classification. *IEEE Wireless Communications Letters*, 8(1):213–216, 2019.

[SL19b] Meysam Sadeghi and Erik G. Larsson. Physical adversarial attacks against end-to-end autoencoder communication systems. *IEEE Communications Letters*, 23(5):847–850, 2019.

[SLB21] Rajeev Sahay, David J. Love, and Christopher G. Brinton. Robust automatic modulation classification in the presence of adversarial attacks. In *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2021.

[YLJ18] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Juang. Power of deep learning for channel estimation and signal detection in ofdm systems. *IEEE Wireless Communications Letters*, 7(1):114–117, 2018.