2024 IEEE Middle East Conference on Communications and Networking (MECOM) | 979-8-3503-7671-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/MECOM61498.2024.10880963

# A Novel Method to Mitigate Adversarial Attacks Against AI-as-a-Service Functionality

Ömer Faruk Tuna
*Ericsson Research*
Istanbul, Turkey
omer.tuna@ericsson.com

Leyli Karaçay
*Ericsson Research*
Istanbul, Turkey
leyli.karacay@ericsson.com

Utku Gülen
*Ericsson Research*
Istanbul, Turkey
utku.gulen@ericsson.com

***Abstract*—Artificial Intelligence (AI) will play a crucial role in enabling 6G technology. In comparison to current networks that mainly offer ubiquitous communication capabilities, 6G aims to transform the network into a robust, distributed AI platform, making its AI capabilities accessible to consumers. This will be facilitated by the AI-as-a-Service (AIaaS) framework, which allows network service providers to effectively utilize AI as a service, opening up new possibilities. Therefore, it is vital to ensure that the AIaaS framework is well-protected against malicious attacks. However, research on Black-Box attacks indicates that merely having access to the model's input and output is sufficient to perform adversarial attacks by creating perturbations that can deceive AI models. In this study, we propose an effective defense mechanism to mitigate inference query-based Black-Box attacks which might target AI models deployed in an AIaaS framework. We have experimentally evaluated and verified the efficacy of our approach on standard datasets.**

***Index Terms*—Adversarial defense, uncertainty, AI-as-a-Service, deep learning, trustworthy AI, 6G security.**

## I. Introduction

The concept for 6G highlights the critical integration of artificial intelligence (AI) applications into mobile networks. To become a robust platform capable of supporting a wide range of AI applications, mobile network architecture must undergo fundamental changes. The future network architecture must be carefully planned in order to easily integrate and facilitate AI applications and services. In this scenario, mobile networks will play an important role in orchestrating, administering, scheduling, and exposing AI-related network services. This complex responsibility emphasizes the critical role that network infrastructure will play in allowing the secure and reliable use of AI-powered functions.

Despite its track record of success in a variety of applications, including wireless communication and network resource orchestration, Machine Learning (ML) poses some distinct security challenges. Recent research has showed that several adversarial attacks can be used effectively against AI-driven wireless networks [1], [2]. Adversarial ML-based attacks are more stealthy and hard to detect than classic attack scenarios because of their small footprints.

Adversarial attacks have the ability to essentially compromise the security of AI-driven networks, posing severe risks, particularly in areas such as telecommunications where security is a major issue. In this paper, we focus on inference query-based Black-Box attacks that may target AI/ML models deployed in an AI-as-a-Service (AIaaS) framework and present a defense mechanism to mitigate them. Our technique allows the model owner to quantify the model's uncertainty (aleatoric uncertainty) estimations during prediction time and use this information to update the model weights, resulting in more accurate predictions.

## II. System Model

AIaaS provides AI services and functions to the users, eliminating the need for users to build and maintain their own AI infrastructure [3]. AIaaS providers can supply pre-built AI models that are accessible via APIs (Application Programming Interfaces). Figure 1 shows an high level overview of AIaaS framework.

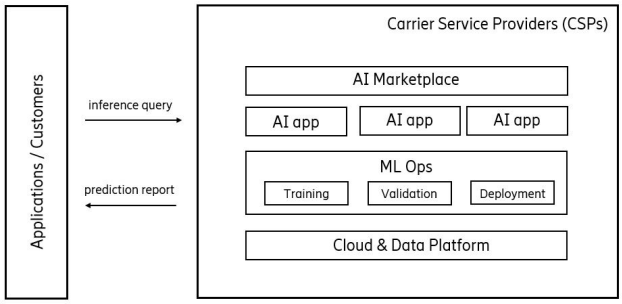


Fig. 1: A simple overview of AIaaS framework.

Using MLOps (Machine Learning Operations) features, the AIaaS framework offers higher-level AI services to the applications. The life-cycle management capabilities for machine learning pipelines are supplied by MLOps. It can be used to optimize the functionality of the network functions (NFs) and automate their (re-)training and (re-)deployment in the context of mobile networks. For example, a new AI/ML model for a NWDAF (Network Data Analytics Function) analytics service (such as network slice load analytics) can be trained, tested, and deployed using MLOps. Additionally, it can be applied to automate the lifetime of an additional AI service that will be made available to AIaaS customers. The automated workflow

of ML pipeline, which consists of steps for creating and using an ML model in order, is shown in Figure 2.

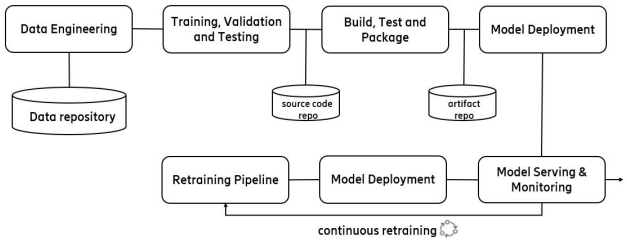


Fig. 2: MLOps Pipeline.

AI agents use the trained AI/ML models to carry out the inference process when the training, validation, and deployment phases are complete. Through an API, external consumers can access the models that have been deployed in the AI agents. Inference requests initiated by the consumers are handled by the AI agents and the outputs are in the form of predictions or reports for the customers in an as-a-service fashion. Deployed models are vulnerable to several types of attacks, such as model evasion and model inversion, in which adversaries attempt to influence the decision of the target model by carefully constructed perturbations to the input. These risks necessitate constant monitoring of the inference outputs provided by AI agents and activating retraining procedures when necessary.

## III. Adversarial Attacks on AI Models

Due to the vulnerabilities in deep neural network (DNN) models, it is challenging for them to defend in adversarial setting. The prediction of the model may lead unexpected outcomes due to its sensitivity to even minor variations in the input data. Figure 3 illustrates how an attacker can take advantage of such a vulnerability.

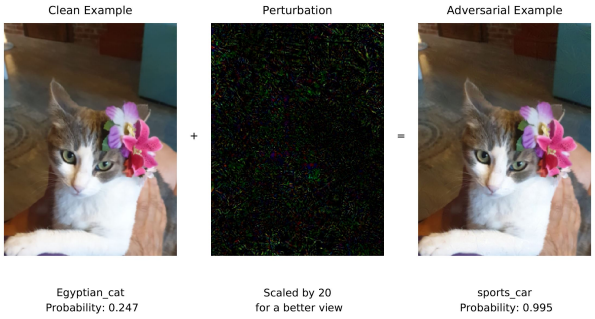


Fig. 3: An example of adversarial attack where a precisely crafted perturbation manipulates the model in such a way that a cat is wrongly classified as a car.

In general, adversarial strategies can be classified based on different criteria. Adversarial attacks can be divided into two groups according to the level of knowledge of the attacker. White-Box Setting is the one in which the attacker is fully aware of the model, including its architecture, weights, hyperparameters, etc. Nonetheless, if the attacker is unaware of the deployed model and protection plan, we refer to this type of setting as Black-Box Setting [4]. In this study, we mainly focus on attacks in a Black-Box setting and partly address some White-Box attacks that has a similar affect on the model decision as in the case of Black-Box attacks.

Most of the White-Box attack methods are based on perturbing the input sample in order to maximize the model's loss. In recent years, many different adversarial attack techniques have been suggested in the literature. The most widely known and used adversarial attacks are Fast-Gradient Sign Method, Iterative Gradient Sign Method, DeepFool and Carlini&Wagner. To begin with the Fast-Gradient Sign Method [5], it is one of the first and most well-known adversarial attacks so far. The direction in which the pixel value of the input image should be changed to minimize the model's loss function is determined by this attack algorithm using the derivative of the model's loss function with respect to the input sample. Once extracted, it simultaneously modifies every pixel in the opposite direction to maximize the loss. With the following formula, we may create adversarial samples for a model whose classification loss function is represented as $J(H_w, \mathbf{x}, y)$: $H_w$ represents the AI model with parameters $w$; $\mathbf{x}$ is the benign input; and $y_{true}$ is the real label of our input.

$$\mathbf{x}^{adv} = \mathbf{x} + \epsilon \cdot sign\left(\nabla_x J(H_w, \mathbf{x}, y_{true})\right) \tag{1}$$

Following the proposal of the FGSM attack, Kurakin et al. [6] suggested a small but important improvement to the FGSM. Using the given value $\epsilon$ to clip the output, instead of taking one large step $\epsilon$, we take multiple smaller steps $\alpha$ in the direction of the gradient sign. FGSM is merely applied iteratively to an input sample in this technique, which is sometimes referred to as the Basic Iterative Method (BIM). Equation 2 describes how to generate perturbed images under the $l_{inf}$ norm for a BIM attack.

$$\begin{aligned} \mathbf{x}_t^* &= \mathbf{x} \\ \mathbf{x}_{t+1}^* &= clip_{x,\epsilon}\{\mathbf{x}_t + \alpha \cdot sign\left(\nabla_{\mathbf{x}} J(H_w, \mathbf{x}_t^*, y_{true})\right)\} \end{aligned} \tag{2}$$

where $\mathbf{x}$ is the clean sample input to the model, $\mathbf{x}^*$ is the output adversarial sample at $i^{\text{th}}$ iteration, $J$ is the loss function of the model, $w$ denotes model parameters, $y_{true}$ is the true label for the input, $\epsilon$ is a configurable parameter that limits maximum perturbation amount in given $l_{inf}$ norm, and $\alpha$ is the step size.

Moosavi-Dezfooli et al. proposed the Deepfool attack algorithm [7], which is distinct from the gradient-based attack mentioned above. The idea behind this attack is that neural network models behave like linear classifiers, with hyperplanes separating the classes. At each iteration, the algorithm starts at the initial input point $\mathbf{x_t}$ and finds the closest hyperplane as well as the smallest perturbation amount, which is the orthogonal projection to the hyperplane. The algorithm then computes

$\mathbf{x}_{t+1}$ by adding the smallest perturbation to the $\mathbf{x}_t$ and checks for misclassification. Typically, the generated adversarial sample is in close proximity to the model's decision boundary. There is also Carlini&Wagner [8] attack algorithm which might be considered as one of the strongest attack algorithms so far. The attack is reformulated by the authors as an optimization problem that can be resolved by gradient descent. The level of prediction score for generated adversarial sample can be altered using the algorithm's $confidence$ parameter. Application of this attack with default setting (confidence set to 0) would often result in adversarial samples near decision boundary for a normally trained model. And high confident adversaries generally located further away from decision boundary.

In the Black-Box attack scenario, as opposed to the White-Box setup, the adversary only has access to the outputs of the target model (either only the decisions or all the probability scores). For the cases where the adversary has access to all the probability scores of a given input, Ilyas et al. [9] and Chen et al. [10] proposed score-based methods using zeroth-order gradient estimation for craft adversarial perturbations. An attacker has just access to decisions in a more practical and realistic scenario that applies to the majority of AI-as-a-Service implementations. Brendel et al. [10] introduced Boundary Attack, which generates adversarial examples via rejection sampling and achieves comparable performance with state-of-the-art White-Box attacks. Nevertheless, their approach requires a relatively large number of model queries, rendering it impractical for real-world applications. Later, Chen et al. [11] introduced HopSkipJumpAttack which is a decision-based attack method relied on an estimation of model's gradient direction and binary-search procedure for approaching the decision boundary. The powerful part of this attack is that it requires significantly fewer model queries than previously proposed decision-based Black-Box attack algorithms, yet achieves competitive performance. Finally, Andriushchenko et. al. [12] proposed Square Attack which is again a query efficient Black-Box attack that is not based on model's gradient and can break defenses that utilize gradient masking.

Figure 4 shows where the adversarial samples crafted with different attack algorithms are projected on the imaginary 2D manifold of high dimensional decision space.

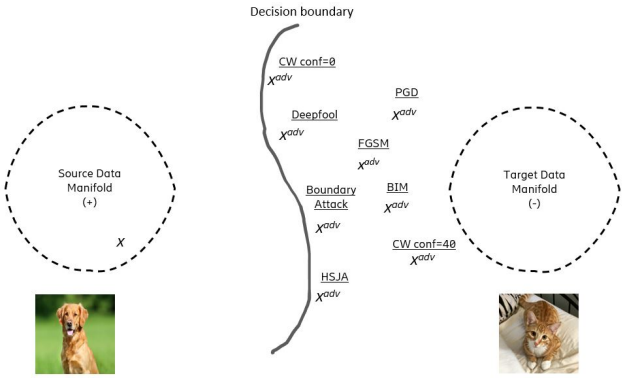


Fig. 4: Projection of adversarial samples on the imaginary 2D submanifolds.

## IV. Defense Against Adversarial Attacks

As a defense solution, adversarial training [5] is suggested, in which the robustness of the deep learner is increased by training it using adversarial samples. This technique is mathematically represented as a Mini-max game. The end result of the entire procedure is a model that should be resilient to adversarial attacks used during the training of the model. Nevertheless, adversarially trained models remain vulnerable to Black-Box attacks [11]. Quantifying uncertainty in model predictions and avoid making decision when there is high uncertainty level has been suggested by some studies [13] in the literature. This has been shown to be effective in some situations and keep the model from making mistakes by discarding the prediction when the model is not confident. However, this choice might also result the model to abstain even there is a chance of making a correct prediction. Furthermore, it is known that the model may make wrong prediction with a high degree of confidence for particular adversarial samples. The adversary will eventually be able to trick the model in these cases because there will be low quantified uncertainty, preventing the model from abstaining.

Regarding the studies which specifically aiming to address inference query-based Black-Box attack scenarios, Chen et. al. [14] proposed a method which requires the remote AI system to store the previous input data per user account basis and compare the historical data with the current input to the model. The idea is that the adversary adds a minimal perturbation amount each time and one can make intelligent data analysis on the model inputs to capture the similarity. Nevertheless, this method increases the memory complexity and add a burden to the AIaaS platform. And this method can only work if the adversary persists sending the adversarial input from only one account. If the adversary sends data from multiple accounts, the defense method will eventually fail. Li et. al. [15] enhanced the idea of Chen et. al. and proposed a method which requires less memory for the remote AI system. And their method can work even in the case of adversary implementing the attack using multiple different accounts. However, their method still needs additional memory within the remote AI system. And these methods only help to detect adversarial samples. They don't give any clue about the actual class of the input sample.

### A. Proposed Defense Method

Our proposed solution is a defense mechanism to mitigate inference queries based Black-Box attacks (evasion attacks) during inference phase of the AI/ML model. It aims to defend against the attack cases when the probability of the wrongly classified class is close to the probability of the actual class which potentially means highly uncertain predictions. That is the impact of the proposed method is high against the adversarial samples which are located close to decision boundary

of the model. And this is the case for Black-box adversarial attack scenarios as the adversary does not have access to the model details and therefore does not have a chance to optimize the adversarial perturbation to force the model to make over confidently wrong prediction. Some other known White-box adversarial attacks (Deepfool, Carlini&Wagner with default setting) fall into this category as well.

Our solution lets the model owner to quantify the uncertainty estimates of the model during prediction time. And for the highly uncertain cases, we suggest updating the model weights in a direction to minimize quantified uncertainty value (aleatoric uncertainty). This can be considered as a one-time update operation specific to the suspicious input. And after the model has been updated, the prediction will be done with this updated model. After the prediction is over for the suspicious input, the model owner returns back to the old model. Figure 5 shows an high level illustration of our method.

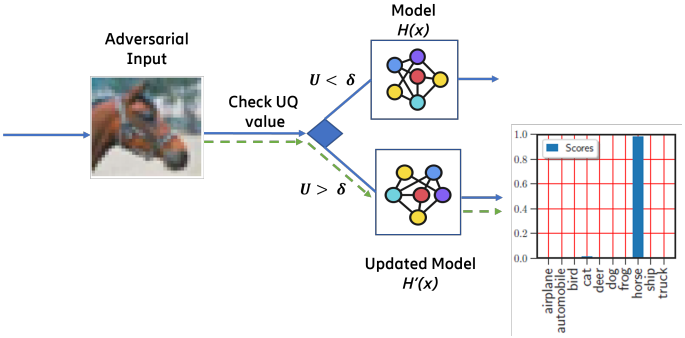


Fig. 5: High level illustration of our method.

### B. Advantages of the Proposed Solution

Our solution provides robustness to the exposed ML models for the cases where the adversarial samples are located close to the decision boundary of the model. Existing Black-box adversarial attacks and some of the known White-box adversarial attack algorithms (i.e. Deepfool or Carlini&Wagner attacks) result in this type of adversarial samples. The proposed method lets the model to predict these kinds of malicious inputs correctly. Besides, it can also be used to detect the adversarial samples. If the initial prediction $H(x)$ and the prediction of the updated model $H'(x)$ are different, then it is a sign of potential evasion attack incident. Since the method can detect the adversarial samples and since the method also knows the true class $(H'(x) = y')$ of the adversarial sample, the model owner can have a chance to store these adversarial samples $(x, y')$ pairs and use them for adversarial training purposes. It is known that crafting adversarial perturbation is a computationally costly process. Via our proposed solution, the model owner can have a chance to use the adversarial samples free of charge, without needing to craft them (adversarial training for free). The algorithmic details are provided in Algorithm 1. Finally, it is known that adversarial samples are transferable. That is; any adversarial sample that can fool a classifier will mostly fool any other classifier that is trained on the same task. Based on this fact, if an adversary targets publicly available AI service and can craft a successful adversarial sample, then it might be risk for other similar AI services as well. Therefore, successfully defending open access AI services can also secure the other AI services as well. Figure 6 shows is a detailed implementation of our solution during a Black-box Adversarial Attack. It might be argued that our proposed solution might introduce additional latency in the AI model responses and requires additional resources. However, the level of robustness it provides might compensate this potential trade-off.

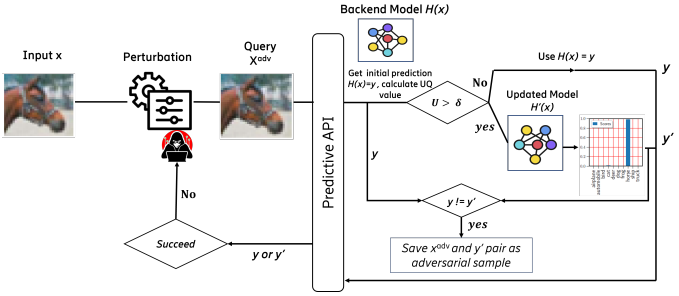


Fig. 6: Detailed implementation of our solution during a Black-Box Adversarial Attack.

**Algorithm 1:** The proposed defense algorithm.

**Input:** $x, \delta$
**Output:** $y$

Compute initial output $y$ through AI-model: $H_w(x)$;
  Quantify uncertainty level $U$ of initial output;
**if** $\delta < U$ **then**
  Update AI-model weights in a direction to reduce uncertainty of output and get $\hat{H}_{w'}$
  Compute updated output $y' : \hat{H}_{w'}(x)$
  Save input and updated output pair $x, y'$ for later stages of adversarial training of AI-model
  Discard updated AI-model $\hat{H}_{w'}(x)$ and revert to initial AI model $H_w(x)$
  return $y'$
**else**
  return $y$

For the uncertainty quantification, we have used the efficient approach which was initially proposed by Gal et al. [16] and then improved by Kwon et. al. [17]. Gal et al. showed that a DNN model with inference time dropout is equivalent to a Bayesian approximation of the Gaussian process. Inference time dropout acts as an ensemble approach. In each single ensemble model, the system needs to drop out different neurons in network layers according to the dropout ratio in the prediction time. The overall prediction uncertainty is approxi-

mated by finding the variance of the probabilistic feed-forward MC dropout sampling during prediction time. Then, Kwon et al. suggested an alternate approach for quantifying both epistemic and aleatoric uncertainty in classification models. In the author's method, the variance of the prediction is comprised of two parts that represent aleatoric and epistemic uncertainty. Let $H_{\hat{\omega}}$ represents the neural network model with parameters denoted by $\hat{\omega}$, the number of different output classes is represented by $k$, then the prediction $y$ of a model for any test sample $x$ given the weights of the model is denoted by $p(y|x, H_{\hat{\omega}})$ where $y \in \mathbb{R}^k$. The formulation for their method is given below:

$$Var_{p(y|x,H_\omega)}(y) = \mathbb{E}_{p(y|x,H_\omega)}(y^{\otimes 2}) - \mathbb{E}_{p(y|x,H_\omega)}(y)^{\otimes 2} \quad (3)$$

$$= \underbrace{\frac{1}{T}\sum_{t=1}^{T}[diag\{p(y|x, H_{\hat{\omega}_t})\} - p(y|x, H_{\hat{\omega}_t})^{\otimes 2}]}_{aleatoric} \quad (4)$$

$$\underbrace{+\frac{1}{T}\sum_{t=1}^{T}[p(y|x, H_{\hat{\omega}_t}) \; - \hat{p}(y|x, H_{\hat{\omega}_t})]^{\otimes 2}}_{epistemic} \quad (5)$$

where, $\hat{p}(y|x, H_{\hat{\omega}_t}) = \sum_{t=1}^{T} p(y|x, H_{\hat{\omega}_t})$ and $y^{\otimes 2} = yy^T$

Equation 4 and 5 output a matrix of shape $k \times k$ where the diagonal elements represent the variance of each output class and we used the mean of the diagonal terms for quantifying uncertainty metrics for a given input $\mathbf{x}$. In our study, we opted to use aleatoric uncertainty and used Equation 4 for uncertainty quantification.

## V. Simulation Results

We started our experiments by first training two different CNN models using MNIST (Digit) and CIFAR-10 datasets and attained accuracy rates of 99.44% and 82.43%, respectively. The architectures of our CNN models and the hyper parameters used in training are listed in Table I and Table II. We continued our experiments by applying several different attack types on each test sample to craft their adversarial counterparts. The selected set of attacks for our study include both Black-Box attack and White-Box attack algorithms. We then tried to feed those adversarial samples back to the target models. And we performed these steps with and without our defense solution enabled. For the one time model update operation which is available in Step 3 of Algorithm 1, we used Adam optimizer with a learning rate of $0.001$. For the choice of $\delta$, we suggest to inspect the uncertainty values of the model for all the correct and wrong predictions on test dataset, calculate the mean values and then take the average of these two. So for MNIST, we set $\delta = 0,0125$ and for CIFAR10 we set $\delta = 0,0114$. The results of our experiments are provided in Table III and IV.

TABLE I: CNN model architectures

| Dataset | Layer Type | Layer Info |
|---|---|---|
| MNIST | Conv. (padding:1) + ReLU | $3 \times 3 \times 16$ |
| | Max Pooling | $2 \times 2$ |
| | Conv. (padding:1) + ReLU | $3 \times 3 \times 16$ |
| | Max Pooling | $2 \times 2$ |
| | Conv. (padding:1) + ReLU | $3 \times 3 \times 32$ |
| | Dropout | p : 0.25 |
| | Conv. (padding:1) + ReLU | $3 \times 3 \times 32$ |
| | Dropout | p : 0.25 |
| | Fully Connected + ReLU | $1568 \times 100$ |
| | Dropout | p : 0.25 |
| | Fully Connected + ReLU | $100 \times 10$ |
| CIFAR10 | Conv. (Padding = 1) + ReLU | $3 \times 3 \times 32$ |
| | Conv. (Padding = 1) + ReLU | $3 \times 3 \times 64$ |
| | Max Pooling (Stride 2) | $2 \times 2$ |
| | Conv. (Padding = 1) + ReLU | $3 \times 3 \times 128$ |
| | Conv. (Padding = 1) + ReLU | $3 \times 3 \times 128$ |
| | Max Pooling (Stride 2) | $2 \times 2$ |
| | Conv. (Padding = 1) + ReLU | $3 \times 3 \times 256$ |
| | Dropout | p : 0.1 |
| | Conv. (Padding = 1) + ReLU | $3 \times 3 \times 256$ |
| | Dropout | p : 0.1 |
| | Max Pooling (Stride 2) | $2 \times 2$ |
| | Fully Connected + ReLU | $4096 \times 512$ |
| | Dropout | p : 0.25 |
| | Fully Connected + ReLU | $512 \times 512$ |
| | Dropout | p : 0.25 |
| | Fully Connected + ReLU | $512 \times 10$ |

TABLE II: Hyperparamters

| | MNIST | CIFAR10 |
|---|---|---|
| Optimizer | Adam | Adam |
| Learning Rate | 0.001% | 0.001% |
| Batch Size | 30 | 75 |
| number of epochs | 30 | 75 |

TABLE III: Experimental results on MNIST dataset

| | Attack Success Rate Without Our Defense | Attack Success Rate With Our Defense |
|---|---|---|
| HopSkipJump $\ell_{inf}$ $eps = 0.15$ | 79,02% | 6,10% |
| Boundary $\ell_2$ $eps = 2.016$ | 87,02% | 6,73% |
| Square $\ell_{inf}$ $eps = 0.15$ | 91,81% | 20,08% |
| Carlini & Wagner $\ell_2$ $eps = 2.016$ $conf = 0$ | 94,58% | 10,41% |
| Deepfool $\ell_{inf}$ $eps = 0.15$ | 67,59% | 13,14% |

The results show that our proposed solution introduces considerable degree of robustness to the AI models which might be deployed to the cloud and consumed in a as-a-service manner. And the performance of our method is higher in inference queries based Black-Box attacks compared to White-Box attacks where the attacker has more control on the produced adversarial samples. Finally, we have checked the effect of our proposed method on the natural (clean) performance of the model. To do this, we used all the clean test data samples from MNIST dataset and measured the accuracy

TABLE IV: Experimental results on CIFAR10 dataset

| | Attack Success Rate Without Our Defense | Attack Success Rate With Our Defense |
|---|---|---|
| HopSkipJump $\ell_{inf}$ $eps = 4/255$ | 89.24% | 12.69% |
| Boundary $\ell_2$ $eps = 0.42$ | 91.58% | 13.34% |
| Square $\ell_{inf}$ $eps = 4/255$ | 94.01% | 22.23% |
| Carlini & Wagner $\ell_2$ $eps = 0.42$ $conf = 0$ | 98.46% | 18.60% |
| Deepfool $\ell_{inf}$ $eps = 4/255$ | 94.69% | 24.86% |

of the model when our method is enabled. We get 99.37% accuracy which is almost identical to the original performance of the model.

For the samples where the model is introduced to an adversarial sample and still makes a correct prediction via our proposed one-time model update operation, the model owner has a chance to save the adversarial sample for later stages of adversarial training. This way, the robustness of the model might be further improved without any need of spending extra resources for crafting these adversarial samples beforehand. Our proposed method and the adversarial training can be seen complementary and used together. In Figure 7, you can see the automated workflow of ML pipeline which consists of sequential stages for developing and utilizing an ML model.

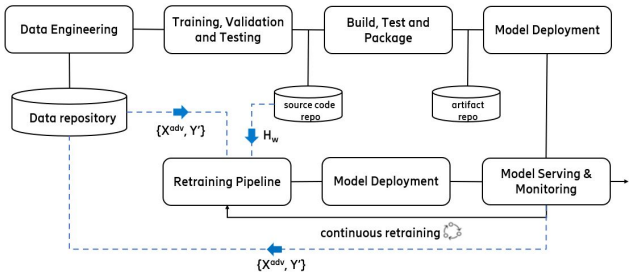


Fig. 7: Proposed updates to the MLOps Pipeline to enable our solution.

## VI. Conclusion

In this paper, we proposed a novel method to defend the AI models deployed and offered in AIaaS framework from potential Black-Box attacks. Our solution can also be used to identify the adversarial samples introduced by the malicious parties to the AI apps in inference phase, which then helps to benefit from the detected adversarial samples in later stages of adversarial (re-)training. We empirically shown the effectiveness of the proposed approach and suggested possible interactions within the key components of the MLOps pipeline to enable our solution. As a future work, we plan to elaborate on whether our proposed method is applicable to other types of Black-Box attacks such as model inversion or membership inference attacks.

## Acknowledgment

This work was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) through the 1515 Frontier Research and Development Laboratories Support Program under Project 5169902, and has been partly funded by the European Commission through the Horizon Europe/JU SNS project ROBUST-6G (Grant Agreement no. 101139068).

## References

[1] B. Kim, Y. Shi, Y. E. Sagduyu, T. Erpek, and S. Ulukus, "Adversarial attacks against deep learning based power control in wireless communications," in *2021 IEEE Globecom Workshops*, 2021, pp. 1–6.

[2] O. F. Tuna, F. E. Kadan, and L. Karacay, "Practical adversarial attacks against ai-driven power allocation in a distributed mimo network," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 759–764.

[3] "HEXA-X Project, D5.3: Final 6G architectural enablers and technological solutions," https://hexa-x.eu/.

[4] Z. Zheng and P. Hong, "Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[5] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[6] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2017. [Online]. Available: https://arxiv.org/abs/1607.02533

[7] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," 2016.

[8] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2017.

[9] A. Ilyas, L. Engstrom, and A. Madry, "Prior convictions: Black-box adversarial attacks with bandits and priors," 2019.

[10] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2018.

[11] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1277–1294.

[12] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 484–501.

[13] B. Kompa, J. Snoek, and A. L. Beam, "Second opinion needed: communicating uncertainty in medical machine learning," *npj Digital Medicine*, vol. 4, no. 1, p. 4, Jan 2021.

[14] *SPAI '20: Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*. New York, NY, USA: Association for Computing Machinery, 2020.

[15] H. Li, S. Shan, E. Wenger, J. Zhang, H. Zheng, and B. Y. Zhao, "Blacklight: Scalable defense for neural networks against Query-Based Black-Box attacks," in *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, 2022, pp. 2117–2134.

[16] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," 2016.

[17] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation," *Computational Statistics & Data Analysis*, vol. 142, p. 106816, 2020.