Ownership Verification of DNN Models Using White-Box Adversarial Attacks with Specified Probability Manipulation

Teruki Sano
Graduate School of Information Sciences
Tohoku University
Sendai, Japan
sano.teruki.r2@dc.tohoku.ac.jp

Minoru Kuribayashi† Masao Sakai Shuji Isobe Eisuke Koizumi
Center for Data-driven Science and Artificial Intelligence
Tohoku University
Sendai, Japan
†kminoru@tohoku.ac.jp

***Abstract*—In this paper, we propose a novel framework for ownership verification of deep neural network (DNN) models for image classification tasks. It allows verification of model identity by both the rightful owner and third party without presenting the original model. We assume a *gray-box* scenario where an unauthorized user owns a model that is illegally copied from the original model, provides services in a cloud environment, and the user throws images and receives the classification results as a probability distribution of output classes. The framework applies a *white-box* adversarial attack to align the output probability of a specific class to a designated value. Due to the knowledge of original model, it enables the owner to generate such adversarial examples. We propose a simple but effective adversarial attack method based on the iterative Fast Gradient Sign Method (FGSM) by introducing control parameters. Experimental results confirm the effectiveness of the identification of DNN models using adversarial attack.**

***Index Terms*—Adversarial example, Deep neural network (DNN), Ownership verification**

I. Introduction

Artificial intelligence possesses high information processing capabilities and versatility. Among its various applications, deep neural networks (DNNs) are widely utilized, and machine learning tools are offered as cloud computing services, such as Machine Learning as a Service (MLaaS). As trained deep neural network (DNN) models take significant development costs and they have high social value, it is crucial to protect the ownership of DNN models.

Trained models are at risk of unauthorized misappropriation of their parameters and algorithms by individuals seeking to circumvent the effort and cost required for model development. To take countermeasures against such misappropriation, DNN watermarking and fingerprinting have been investigated for the protection of intellectual property of DNN models [1], [2]. DNN watermarking is a technique for embedding specific information in a DNN model, such as internal structure and weight parameters, during the training phase. On the other hand, DNN fingerprinting extracts some unique model properties like decision boundaries as the fingerprint. It is less realistic to assume white-box access to the model's internal structure and weight parameters for verification. Instead, it is assumed that the watermark/fingerprint will be obtained by throwing queries to a suspected model and receiving the responses. In such a case, the watermark/fingerprint reflects the behavior of the model on the given trigger images as the backdoor. The behavior is regarded as an identifier for the ownership verification. In both cases, the identifier is extracted from the suspected model through queries and then it is compared against the constructed identifier of the original model. There are two major drawbacks in these ownership verification approaches. One is that the number of trigger images is limited as those are pre-determined during the training of DNN models. The other risk is that unauthorized users might recognize that they are being tested in terms of their legitimacy. The behavior of the model on the trigger images is generally peculiar compared with other inputs. Due to the characteristics of queries and responses, unauthorized users may recognize the action of ownership verification approach.

In this study, we propose a novel framework for verifying the identity of DNN models by utilizing *white-box* adversarial attacks [3], allowing both rightful owners and third parties to demonstrate model ownership. With full access to the original model, the owner can generate adversarial samples that serve as proof of ownership, without disclosing the model. On any given requests from the third party on arbitrary images and specific probability values, the owner presents those accurate adversarial samples which are extremely difficult to generate without the model. Here, we assume a *gray-box* scenario where a potentially copied DNN model operates in a cloud environment under the control of an unauthorized user and outputs per-class probability distributions.

We propose a novel white-box adversarial attack that accurately manipulates the probability of a specific target class while maintaining the original class's probability. Our method allows the owner to generate adversarial samples that adjust class probabilities to designated values without revealing the model. To avoid being noticed by unauthorized users, the method ensures that the correct class probability remains

This study was supported by the JSPS KAKENHI(25K15225), JST SICORP (JPMJSC20C3), JST CREST (JPMJCR20D3), Japan.

dominant, preventing anomaly classifications.

II. Related Work

A. Threat Model

We assume the following threat model, the rightful owner is the creator of the original model, while the unauthorized user is an entity that has illicitly obtained and deployed a copied model in a cloud service which will be available as an online API. The rightful owner seeks to verify the identity of the model in two cases: (i) when attempting to confirm whether the copied model in the cloud is identical to the original model, and (ii) when proving to a third party that the copied model is identical to the original model. Since the unauthorized user denies the rightful owner's claim and is unlikely to disclose model parameters, the owner must conduct verification without direct access to the copied model's internal structure and weight parameters. Moreover, given that the unauthorized user can observe all queries and outputs, it may attempt to manipulate or block specific queries to hinder verification.

B. DNN Watermarking

DNN watermarking embeds information into models to prevent unauthorized use and theft [1]. Methods include black-box watermarking, which utilizes input-output relationships, white-box watermarking, which embeds information in model parameters, and gray-box watermarking, which embeds watermarks into probability outputs [4]. Backdoor-based watermarking, such as that proposed by Adi et al. [5], modifies models to return specific outputs for trigger inputs. However, these methods face several challenges:(i) If a model is already published without embedded watermarking information, verification of ownership becomes challenging. (ii) Since watermarking techniques require models to learn information unrelated to their primary task, their outputs may exhibit recognizable patterns. These patterns can be statistically analyzed by malicious users, leading to anomaly detection. Retraining the model may further reduce the effectiveness of the watermark, diminishing its verification capabilities. (iii) The presence of hidden watermarks may be noticed and it will be removed/modified through retraining or fine-tuning [6], [7]. Furthermore, the requirement for DNN watermarking techniques to encode task-unrelated information may lead to a decline in model accuracy.

C. DNN Fingerprinting

DNN fingerprinting extracts distinctive model characteristics to verify identity. This approach utilizes adversarial samples designed near decision boundaries and verifies whether a target model exhibits specific behavior [8]. Unlike watermarking, it does not require embedding information directly into the model. However, if the fingerprinting mechanism is statistically analyzed, it may be detected as an anomaly. In particular, the model's output tendencies and decision boundary patterns can be analyzed to identify the presence of fingerprinting, leading to potential retraining or fine-tuning to circumvent verification.

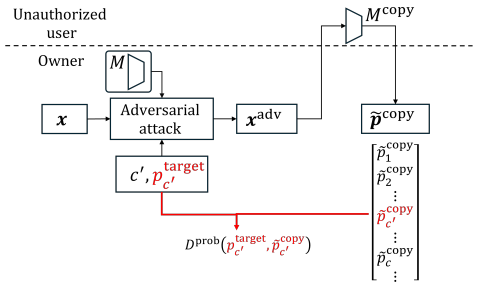


Fig. 1. The owner verifies the identity of the original model and the copied model.

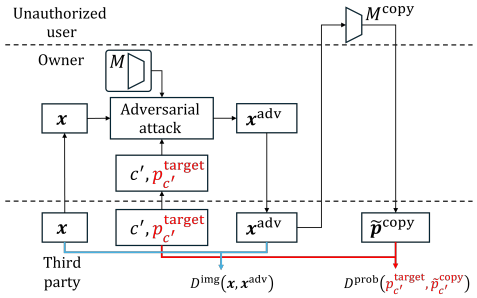


Fig. 2. A third party verifies the identity of the original model and the copied model by requesting the owner to create the adversarial sample for a given image with specified conditions.

III. Proposed Method

In this study, we propose an ownership verification method that applies adversarial attacks to enable both the owner and a third party to verify the identity of an original model and a copied model deployed in a cloud environment.

A. Assumed Environment

We consider two scenarios for verifying the identity of an original model and a copied model: (i) The owner verifies whether the copied model deployed in the cloud is identical to the original model, as illustrated in Fig. 1. (ii) The owner proves the identity of the copied model to a third party and claims ownership, as illustrated in Fig. 2.

In Fig. 1 and Fig. 2, M and M^{copy} represent a classification DNN model for k-class image classification designed and trained by a rightful owner and its unauthorized copied model, respectively. These models do not contain special embedding structures. Model M is preserved locally only at the owner, whereas M^{copy} is deployed in a cloud environment and operates as a gray-box scenario such that it outputs class probability distributions.

Let $\boldsymbol{x}$ denote an input image. The output probability vectors of M and M^{copy} for $\boldsymbol{x}$ are represented as $\boldsymbol{p}$ and $\boldsymbol{p}^{copy}$, respectively, where each element corresponds to the probability of a class in a k-dimensional vector. Similarly, the output probability vectors for an adversarial sample $\boldsymbol{x}^{\text{adv}}$ are denoted as $\tilde{\boldsymbol{p}}$ and $\tilde{\boldsymbol{p}}^{copy}$ for M and M^{copy}, respectively. The correct class for $\boldsymbol{x}$ under M is $c = \underset{i}{\text{argmax}}\ p_i$. Additionally, let

$D^{\text{prob}}(\cdot)$ and $D^{\text{img}}(\cdot)$ be functions that calculate probability distance and perceptual image distance, respectively.

B. Adversarial Attack

An adversarial attack is a technique that generates adversarial samples by intentionally perturbing input data to induce misclassification in a machine learning model. These perturbations are small enough to be imperceptible to the human eye, making it difficult to be recognized.

The computation of perturbations is categorized into two types based on access privileges: (i) White-box attacks: Executed when the internal structure and parameters of the model are accessible. (ii) Black-box attacks: Executed when only input-output observations are available.

Additionally, adversarial attacks are classified based on their objectives: (i) Targeted attack: Aims to misclassify an adversarial sample into a specific class. (ii) Untargeted attack: Aims to misclassify an adversarial sample into any incorrect class.

A targeted attack is more accurate in a white-box setting with internal model access, whereas its success rate drops in a black-box setting due to limited information.

C. Overview

The objective of the owner, as illustrated in Fig. 1, is to verify whether $M = M^{\text{copy}}$. To do this, the owner specifies a target class c' and its target probability $p_{c'}^{\text{target}}$. Using M, the owner generates an adversarial sample $\boldsymbol{x}^{\text{adv}}$ via a white-box targeted adversarial attack under the restriction of the perceptual distance between $\boldsymbol{x}$ and $\boldsymbol{x}^{\text{adv}}$, $D^{\text{img}}(\boldsymbol{x}, \boldsymbol{x}^{\text{adv}})$. Remember that $\boldsymbol{x}^{\text{adv}}$ must not be excessively altered from the original image $\boldsymbol{x}$. The output probability vector $\tilde{\boldsymbol{p}} = M(\boldsymbol{x}^{\text{adv}})$ is expected to satisfy $\tilde{p}_{c'} \approx p_{c'}^{\text{target}}$, while $\underset{i}{\operatorname{argmax}}\, \tilde{p}_i$ remains c to prevent the unauthorized user from detecting the verification approach.

Next, the adversarial sample $\boldsymbol{x}^{\text{adv}}$ is fed into M^{copy}, and the output probability vector $\tilde{\boldsymbol{p}}^{\text{copy}} = M^{\text{copy}}(\boldsymbol{x}^{\text{adv}})$ is obtained. If the probability distance between the specified probability and the observed probability, $D^{\text{prob}}(p_{c'}^{\text{target}}, \tilde{p}_{c'}^{\text{copy}})$, is sufficiently small, it determines that M and M^{copy} are identical.

In the scenario depicted in Fig. 2, the owner provides proof of model identity to a third party. The third party selects an arbitrary image $\boldsymbol{x}$, a target class c', and a probability $p_{c'}^{\text{target}}$, and provides them to the owner as a request. The owner, possessing complete knowledge of M, generates an adversarial sample $\boldsymbol{x}^{\text{adv}}$ according to the request and returns it to the third party. The third party then queries M^{copy} with $\boldsymbol{x}^{\text{adv}}$ and evaluates the response whether the observed probability satisfies $\tilde{p}_{c'}^{\text{copy}} \approx p_{c'}^{\text{target}}$, namely $D^{\text{prob}}(p_{c'}^{\text{target}}, \tilde{p}_{c'}^{\text{copy}}) \approx 0$. Since accurate adversarial attacks are only possible with full model access, successful verification confirms the owner's possession of M^{copy}.

D. Generation of Adversarial Samples

In this study, we propose a new approach based on the targeted I-FGSM (Iterative-Fast Gradient Sign Method) [3] as a white-box adversarial attack that controls the probability of a targeted class while satisfying $c = \underset{i}{\operatorname{argmax}}\, \tilde{p}_i$. We propose targeted I-FDGSM (Iterative-Fast Dual Gradient Sign Method), a novel method that allows simultaneous control of two class probabilities.

1) I-FGSM: I-FGSM is a white-box adversarial attack proposed by Kurakin et al., which is an extension of FGSM [9]. It utilizes the gradient of the loss function obtained through backpropagation to compute perturbations iteratively. Let $\boldsymbol{x}_i^{\text{adv}}$ represent the adversarial sample images of $\boldsymbol{x}$ at i-th iteration. The update rule for targeted I-FGSM is expressed as follows:

$$\boldsymbol{x}_0^{\text{adv}} = \boldsymbol{x},$$
$$\boldsymbol{x}_{N+1}^{\text{adv}} = \text{Clip}_{\boldsymbol{x},\varepsilon}\left\{\boldsymbol{x}_N^{\text{adv}} - \alpha^{c'}\ \text{sign}\left(\nabla_x C(\boldsymbol{x}_N^{\text{adv}}, c')\right)\right\}. \quad (1)$$

Here, $\varepsilon, \alpha^{c'}, N, C$ represent the maximum perturbation range, the magnitude of perturbation at each step, the number of iterations, and the loss function, respectively. The term $\nabla_x C(\boldsymbol{x}^{\text{adv}} N, c')$ denotes the gradient of the loss function C concerning the target class c'. By applying perturbations in the inverse direction of this gradient, the loss C for the target class c' is reduced, thereby enabling a targeted attack. However, this method only considers the influence on c', leading to a situation where the probability of c' becomes excessively high while the probability of c decreases significantly.

2) I-FDGSM: To maintain the highest probability of c while adjusting the probability value of c', we propose I-FDGSM, formulated as follows:

$$\boldsymbol{x}_0^{\text{adv}} = \boldsymbol{x},$$
$$\boldsymbol{x}_{N+1}^{\text{adv}} = \text{Clip}_{\boldsymbol{x},\varepsilon}\Big\{\boldsymbol{x}_N^{\text{adv}} -$$
$$\alpha^{\text{com}}\ \text{sign}\left(\beta^c \nabla_x C(\boldsymbol{x}_N^{\text{adv}}, c) + \beta^{c'} \nabla_x C(\boldsymbol{x}_N^{\text{adv}}, c')\right)\Big\} \quad (2)$$

Here, $\alpha^{\text{com}}, \beta^c, \beta^{c'}$ represent the magnitude of perturbation at each step and the coefficients applied to the gradients of c and c', respectively. By appropriately tuning these parameters, it becomes possible to control the probabilities of both classes. It leverages the interaction between the gradients of the two classes to reduce perturbations and improve effectiveness.

Algorithm 1 illustrates the procedure to create adversarial samples $\boldsymbol{x}^{\text{adv}}$.

IV. Computer Simulation

In this study, both M and M^{copy} are set as CNN models performing an image classification task on the ImageNet dataset [10], consisting of $k = 1000$ classes. In the following sections, we describe experiments evaluating the adversarial sample generation capability of I-FDGSM and the verification of model ownership.

A. Adversarial Sample Generation

We conducted a comparative experiment to evaluate the capability of adversarial sample generation using I-FGSM and I-FDGSM.

Algorithm 1 I-FDGSM

Require: Original Image: $\boldsymbol{x}$, Model: M,
Target Classes: c, c', Target Probability Value: $p_{c'}^{\text{target}}$,
Factors of I-FDGSM: $\alpha^{\text{com}}, \beta^{c} = 1, \beta^{c'} = 1$,
Averaging Interval: l, Tolerance for Error: T^{diff}
Ensure: Adversarial Image: $\boldsymbol{x}^{\text{adv}}$

$\boldsymbol{x}_0^{\text{adv}} = \boldsymbol{x}$
for $N \in \{1, 2, \cdots, N^{\max}\}$ **do**
  $\boldsymbol{x}_N^{\text{adv}} = Adv(\boldsymbol{x}_{N-1}^{\text{adv}}, M, \alpha^{\text{com}}, \beta^{c}, \beta^{c'})$
  $\tilde{\boldsymbol{p}}_{\boldsymbol{N}} = M(\boldsymbol{x}_N^{\text{adv}})$
  if $N \mod l = 0$ **then**
    $\tilde{\boldsymbol{p}}^{\text{mean}} = \frac{1}{l} \sum_{i=N-l+1}^{N} \tilde{\boldsymbol{p}}_{\boldsymbol{i}}$
    if $\tilde{p}_{c'}^{\text{mean}} < (1 - T^{\text{diff}}) p_{c'}^{\text{target}}$ **then**
      $\beta^{c'} \leftarrow \beta^{c'} + 1$
    else
      $\beta^{c} \leftarrow \beta^{c} + 1$
    end if
    if $(1 - T^{\text{diff}}) p_{c'}^{\text{target}} < \tilde{p}_{c'}^{\text{mean}} < (1 + T^{\text{diff}}) p_{c'}^{\text{target}}$ **then**
      $\alpha^{\text{com}} \leftarrow 0.5\ \alpha^{\text{com}}$
    end if
    if $\alpha^{\text{com}} < 10^{-10}$ **then**
      $\boldsymbol{x}^{\text{adv}} = \boldsymbol{x}_N^{\text{adv}}$
      break
    end if
  end if
end for

For validation, perturbations are applied to an image $\boldsymbol{x}$ randomly selected from ImageNet based on the information of M. The model M is set as a pre-trained ResNet50-v1 [11], provided by PyTorch [12]. Using Eq.(1) and Eq.(2), adversarial samples $\boldsymbol{x}^{\text{adv}}$ are generated over a maximum of $N^{\max}$ iterations, and the output $\tilde{\boldsymbol{p}} = M(\boldsymbol{x}^{\text{adv}})$ is obtained for each iteration N. The variations of $\tilde{p}_c$ and $\tilde{p}_{c'}$ with respect to N are analyzed.

The parameters for each method are set as follows for validation: $N^{\max} = 1000$, $\alpha^{c'} = 5 \times 10^{-4}$, $\alpha^{\text{com}} = 1 \times 10^{-3}$, $\varepsilon = 0.05$, $l = 5$, and $T^{\text{diff}} = 5 \times 10^{-3}$. These parameters were chosen empirically to ensure stable convergence and effective probability control in preliminary tests.

1) Results and Discussion: The results of I-FGSM and I-FDGSM are shown in Fig. 3. The blue circles and red crosses in the figure represent the variations of $\tilde{p}_c$ and $\tilde{p}_{c'}$ with respect to the number of iterations N.

From Fig. 3 (a), in I-FGSM, $\tilde{p}_c$ and $\tilde{p}_{c'}$ rapidly decrease and increase, respectively, as the number of iterations N increases, with $\tilde{p}_{c'}$ converging to a value close to 1. On the other hand, from Fig. 3 (b)(c)(d), in I-FDGSM, $\tilde{p}_{c'}$ converges to a value sufficiently close to the specified $p_{c'}^{\text{target}}$. Furthermore, $\tilde{p}_c$ remains at the highest probability value and converges with the increase of N. It is interesting to note that the sum of $\tilde{p}_c$ and $\tilde{p}_{c'}$ approaches 1. This result suggests that, in Eq. (2), considering the gradient influences of both c and c' enables the adjustment of probabilities to the local optimal values for the two classes.

TABLE I
SIMULATION CONDITIONS

c'	Randomly selected from 1000 classes
$p_{c'}^{\text{target}}$	Uniform{0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4}
$\alpha^{\text{com}}, \varepsilon, l, T^{\text{diff}}$	$1 \times 10^{-3}, 5 \times 10^{-2}, 5, 5 \times 10^{-3}$

B. Ownership Verification of Models

To verify the identity of copied models using I-FDGSM, we conducted ownership verification experiments.

1) Experimental Conditions: The input image $\boldsymbol{x}$ is randomly selected from the ImageNet dataset, and $\boldsymbol{x}^{\text{adv}}$ is generated using I-FDGSM to satisfy $\tilde{p}_{c'} \approx p_{c'}^{\text{target}}$ based on Eq (2) and Algorithm 1. In practice, if the owner performs the verification, c' and $p_{c'}$ are specified by the owner. Meanwhile, when a third party conducts the verification, $\boldsymbol{x}$ and c' and $p_{c'}$ are specified by the third party.

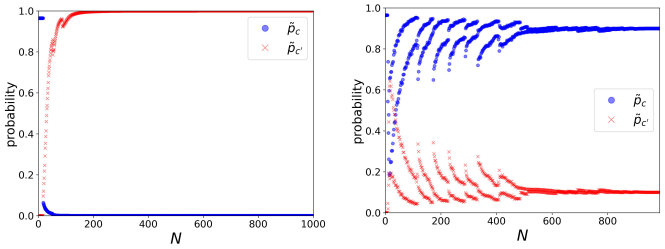
The probability distance function $D^{\text{prob}}(\cdot)$ is used to calculate the relative error between the obtained output probability $\tilde{p}_{c'}^{\text{copy}}$ and the target probability $p_{c'}^{\text{target}}$, as given by Eq. (3):

$$D^{\text{prob}}(p_{c'}^{\text{target}}, \tilde{p}_{c'}^{\text{copy}}) = \frac{|p_{c'}^{\text{target}} - \tilde{p}_{c'}^{\text{copy}}|}{p_{c'}^{\text{target}}}. \tag{3}$$

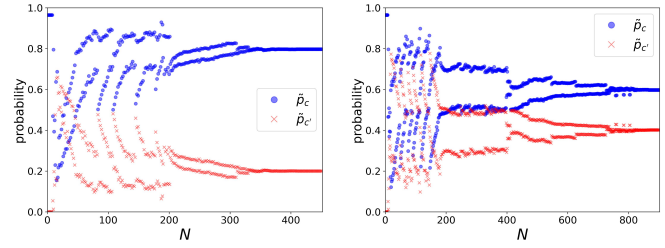
Furthermore, the perceptual distance function $D^{\text{img}}(\cdot)$ is used to measure the similarity between $\boldsymbol{x}$ and $\boldsymbol{x}^{\text{adv}}$ using SSIM$(\cdot)$ (Structural Similarity Index Measure) [13], which considers human visual perception.

The simulation conditions are summarized in Table I.

2) Results and Discussion: Using PyTorch's ResNet50-v1 as M and ResNet50-v1 and ResNet50-v2 as M^{copy}, we evaluated 100 randomly selected images from ImageNet. The results are shown in Fig. 4, where blue bars represent the cases where $M = M^{\text{copy}}$, and red bars represent the cases where $M \neq M^{\text{copy}}$.



(a) I-FGSM (b) I-FDGSM, $p_{c'}^{\text{target}} = 0.1$



(c) I-FDGSM, $p_{c'}^{\text{target}} = 0.2$ (d) I-FDGSM, $p_{c'}^{\text{target}} = 0.4$

Fig. 3. Variations of $\tilde{p}_c$ and $\tilde{p}_{c'}$ with respect to N. (a): I-FGSM, (b)(c)(d): I-FDGSM, with $p_{c'}^{\text{target}} = 0.1, 0.2, 0.4$.

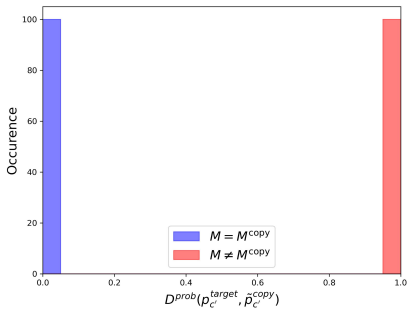


Fig. 4. Distribution of $D^{\text{prob}}(p_{c'}^{\text{target}}, \tilde{p}_{c'}^{\text{copy}})$. Blue bars: $M = M^{\text{copy}} =$ ResNet50-v1. Red bars: $M =$ ResNet50-v1, $M^{\text{copy}} =$ ResNet50-v2.

As shown in Fig. 4, $D^{\text{prob}} \approx 0$ for $M = M^{\text{copy}}$ and $D^{\text{prob}} \approx 1$ otherwise. This result indicates that adversarial samples generated using I-FDGSM are well adjusted for M, allowing more accurate probability control only for the model. Consequently, if $\tilde{p}_{c'}$ in M^{copy} can be controlled, it indicates that the owner has sufficient knowledge of M^{copy}, proving the ownership of M^{copy} to a third party without presenting it.

Next, model ownership verification was conducted using various ResNet and VGG models [14] from PyTorch. For each model, 100 randomly selected images from ImageNet were used as $\boldsymbol{x}$. Fig. 5 represents the average of $D^{\text{prob}}(p_{c'}^{\text{target}}, \tilde{p}_{c'}^{\text{copy}})$ value, denoted by $\overline{D}^{\text{prob}}$, for each pair. From the figure, when $M = M^{\text{copy}}$, $\overline{D}^{\text{prob}} \approx 0$, and when $M \neq M^{\text{copy}}$, $\overline{D}^{\text{prob}} \approx 1$. This indicates that probability value adjustments are possible only for specific models, thereby proving the ownership of M^{copy} across all tested model pairs.

It is worth noting that the average of $\text{SSIM}(\boldsymbol{x}, \boldsymbol{x}^{\text{adv}})$ calculated for each of the 100 generated images per M was at least 0.9875. This confirms that perturbations introduced by I-FDGSM are visually imperceptible, making detection by unauthorized users difficult.

V. Conclusion and Future Work

In this study, we proposed a novel framework for verifying the identity of trained DNN models without presenting the original model. This framework enables the rightful owner to prove the model's identity to a third party while preserving confidentiality. To avoid detection by unauthorized users, we introduced I-FDGSM, an adversarial attack method that precisely controls the probability values between the original and target classes. Experiments confirmed its high-accuracy verification capability.

Due to the transferability of adversarial perturbations, the proposed method is expected to remain effective against slightly modified models such as those retrained or pruned. Evaluating this robustness is left for our future work. Furthermore, one of the threats in the proposed method is the potential detectability by anomaly detection mechanisms. A prior work has shown that such patterns can be flagged even in encrypted domains like VoIP traffic [15]. Improving stealth and robustness against such detection is another direction for future work.

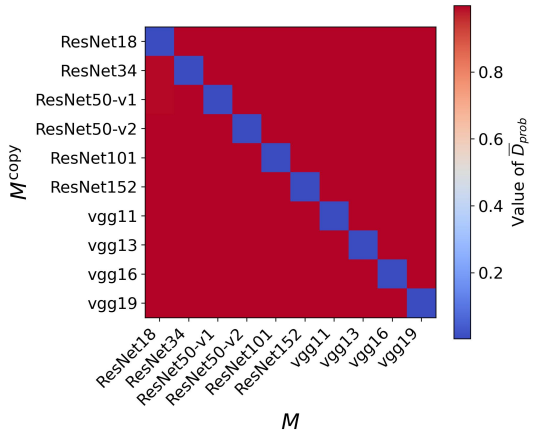


Fig. 5. Heatmap of the average $D^{\text{prob}}(p_{c'}^{\text{target}}, \tilde{p}_{c'}^{\text{copy}})$ values for multiple M and M^{copy}.

References

[1] J. Guo, S. Chen, W. Ding, C. Liang, and Z. Yu, "A survey of deep neural network watermarking techniques," *arXiv preprint arXiv:2103.09274*, 2021. [Online]. Available: https://arxiv.org/abs/2103.09274

[2] Y. Sun, T. Liu, P. Hu, Q. Liao, S. Fu, N. Yu, D. Guo, Y. Liu, and L. Liu, "Deep intellectual property protection: A survey," *arXiv preprint arXiv:2304.14613*, 2023. [Online]. Available: https://arxiv.org/abs/2304.14613

[3] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[4] X. Wang, X. Li, W. Zhang, C. Guo, J. Yan, and Z. Zhang, "Customized watermarking for deep neural networks via label distribution perturbation," *arXiv preprint*, vol. 2208.05477, 2022. [Online]. Available: https://arxiv.org/abs/2208.05477

[5] Y. Adi, C. Baum, M. Cissé, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *Proceedings of the 27th USENIX Security Symposium*, 2018, pp. 1615–1631.

[6] B. Chen, L. Xie, Y. Rong, and L. Huang, "Aeva: Black-box backdoor detection using adversarial extreme value analysis," *arXiv preprint arXiv:2110.14880*, 2021.

[7] Y. Yu, S. Hu, Y. Xiao, J. Chen, Y. Chen, S. Ma, and X. Zhang, "Scale-up: Scalable black-box input-level backdoor detection via analyzing predictions across scales," *arXiv preprint arXiv:2302.03251*, 2023.

[8] X. Cao, J. Jia, and N. Z. Gong, "IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (AsiaCCS)*. ACM, 2021, pp. 14–25. [Online]. Available: https://arxiv.org/abs/1910.12903

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2015.

[10] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[15] P. Addesso, M. Cirillo, M. D. Mauro, and V. Matta, "ADVoIP: Adversarial detection of encrypted and concealed voip," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3342–3357, 2020.