Robust and Imperceptible Black-Box DNN Watermarking Based on Fourier Perturbation Analysis and Frequency Sensitivity Clustering

Yong Liu, Hanzhou Wu, Member, IEEE, and Xinpeng Zhang

Abstract—Recently, more and more attention has been focused on the intellectual property protection of deep neural networks (DNNs), promoting DNN watermarking to become a hot research topic. Compared with embedding watermarks directly into DNN parameters, inserting trigger-set watermarks enables us to verify the ownership without knowing the internal details of the DNN, which is more suitable for application scenarios. The cost is we have to carefully craft the trigger samples. Mainstream methods construct the trigger samples by inserting a noticeable pattern to the clean samples in the spatial domain, which does not consider sample imperceptibility, sample robustness and model robustness, and therefore has limited the watermarking performance and the model generalization. It has motivated the authors in this paper to propose a novel DNN watermarking method based on Fourier perturbation analysis and frequency sensitivity clustering. First, we analyze the perturbation impact of different frequency components of the input sample on the task functionality of the DNN by applying random perturbation. Then, by K-means clustering, we determine the frequency components that result in superior watermarking performance for crafting the trigger samples. Our experiments show that the proposed work not only maintains the performance of the DNN on its original task, but also provides better watermarking performance compared with related works.

Index Terms—Black-box, copyright protection, deep neural networks, frequency analysis, k-means, robust, watermarking.

I. INTRODUCTION

Deep neural networks (DNNs) have achieved great success in many application areas such as computer vision, pattern recognition, and natural language processing. Many technology companies have deployed DNN models in their consumer products to improve the service quality and increase profits. It can be foreseen that DNN based intelligent services will become more and more popular in our daily life. However, creating a good DNN model requires large-scale well-labelled data, expertise of architecture design, and substantial computational resources, meaning that as a kind of expensive digital asset, we should protect the intellectual property of DNNs.

One may build strong access control mechanisms to prevent model leakage beyond authorized parties, which, however, has limited control once the model is shared with trusted users. It is also the case that no matter how well access control systems are designed, they are never foolproof and often fall prey to attacks on the human element [1]. Another option is to modify the host DNN such that the modified DNN not only maintains the performance on the original task, but also carries a secret message that can be used to verify the ownership of the DNN, which is referred to as DNN watermarking [2], [3]. However, unlike many media watermarking methods that treat media data as static signals, DNN watermarking requires us to embed information into a DNN with a specific task, implying that DNN watermarking is somehow dynamic due to the task functionality of the DNN. In other words, we cannot directly apply conventional media watermarking algorithms to DNNs since simply modifying a given DNN may significantly impair the performance of the DNN on its original task. It motivates people to design watermarking methods specifically for DNNs.

It is essential to emphasize that diverse types of DNN models necessitate distinct designs of model watermarking tailored to their specific characteristics. This study concentrates on DNN watermarking for classification models.

Considering whether the ownership verifier needs to access the model details, mainstream DNN watermarking algorithms can be roughly divided into two categories, i.e., white-box DNN watermarking and black-box DNN watermarking. White-box DNN watermarking requires the model verifier to be able to access the target model including the network structure and parameters. For example, Uchida et al. [2] mark the host DNN by designing an embedding regularizer, which embeds a secret watermark into the pooled weights by loss optimization. The owner has to collect the embedded weights of the target DNN model for watermark reconstruction. It is naturally required that the most suitable weights of the DNN can be used for watermark embedding so that the performance of the DNN on its original task will not be impaired, which motivates the authors in [4] to mark the host DNN by adding an independent neural network that is used only for DNN training and DNN verification, and will not be public. In order to achieve a better trade-off between watermark unobtrusiveness and watermark payload, by following the method introduced in [2], Li et al. [5] propose a novel method based on spread transform dither modulation. The above methods directly modulate the existing weights of the host DNN,
which rarely consider the ambiguity attack. Fan et al. [6] propose appending passport layers after convolution layers, which significantly enhances the ability to resist network modifications and ambiguity attack. In addition, Chen et al. [7] propose a novel collusion-secure fingerprinting framework, which is effective for ownership proof and user tracking. Recently, Zhao et al. [8] bypass common parameter-based attacks by introducing a structural watermarking scheme using channel pruning to embed the watermark into the host DNN architecture instead of modifying the parameters, which demonstrates good application prospects. The authors in [9] propose a black-and-white-box watermarking method for DNN classifiers that opens the door to collusion-resistant traitor tracing in black-box, exploiting the properties of Tardos codes, and making it possible to identify the source of the leak before access to the model is granted. Lv et al. [10] propose a novel DNN watermarking solution to detect intellectual property infringement of DNN models against the forgery watermarking attacks. More white-box systems can be found in [11], [12], [13], [14].

Black-box DNN watermarking, on the other hand, assumes that the ownership verifier is not allowed to access the internal details of the target model, but allowed to verify the ownership by interacting with the target model. It is often the case that the embedded watermark is retrieved by interacting with a target model and checking the predictions of the model corresponding to a certain number of carefully crafted samples. Therefore, to embed a watermark, the DNN model to be protected should be trained in such a way that the marked DNN model generates the correct predictions matching the watermark when inputting a sequence of carefully crafted samples, while maintaining the performance of the DNN model on its original task. Along this direction, in [15], Adi et al. present a formal analysis to DNN watermarking based on backdooring and propose a simple but effective watermarking strategy that marks a DNN by training the DNN with normal images and abstract images (associated with random labels). Instead of using abstract images, Zhang et al. [16] have further proposed different strategies to construct the trigger set such as adding a pre-defined meaningful marker or meaningless noise to some normal samples. In this way, by sending normal queries to the target model with the previously generated trigger set, the ownership can be verified. In order to not twist the decision boundary of the DNN on its original task, Zhong et al. [17] propose a novel strategy to watermark a DNN by assigning a new label to the trigger samples during training, which better facilitates feature learning for the trigger samples compared with the existing methods. Hua et al. [18] reveal the vulnerability of existing backdoor watermarking methods to ambiguity attacks and propose a novel unambiguous backdoor watermarking method for DNN models. They further reveal the limitations of existing watermarking fidelity measures and introduce deep fidelity [19] as a solution. Xue et al. [20] propose an active intellectual properties protection technique for DNN models via stealthy backdoor and users’ identities authentication, which has better robustness against query modification attacks. Recently, Zhang et al. [21] and Wu et al. [3] independently introduce a method to mark a model by marking the output of the model, which is suitable for models focusing on generative tasks. Li et al. [22] bind triggers to model owners to protect the copyright of pre-trained language models. He et al. [23] explore intellectual property infringement recognition in model extraction by adding lexical watermarks to the output of the text generator. Backdoor based model watermarking can be extended to dataset watermarking. The difference between the two watermarks is that their goals are different and they have different validation criteria. E.g., in [24], the authors exploit poison-only backdoor attacks for dataset watermarking and design a hypothesis-test-guided method for dataset verification. In [25], the authors explore the non-targeted backdoor watermarking paradigm to protect the copyright of datasets in both cases of toxic labeling and clean labeling. In addition, unlike mainstream methods that mainly focus on convolutional neural networks, researchers have also studied other types of DNNs such as graph neural networks [26]. We refer the reader to [27] for more black-box methods.

In real-world scenarios, it is more likely that the ownership verifier has no access to the internal details of the target DNN. In other words, compared with white-box DNN watermarking, black-box DNN watermarking is more desirable for real-world scenarios. As mentioned above, black-box model verification can be realized by querying the target DNN with a sequence of trigger samples. However, mainstream methods construct the trigger samples by inserting a noticeable pattern to normal samples in the spatial domain, which allows the adversary to construct new samples leading to a successful ambiguity attack according to Kerckhoffs’s principle. Even though Li et al. [28] have successfully used the frequency-domain modification for trigger generation, their intention was mainly to ensure that the marked model possesses good performance on common indicators, which does not take into account frequency sensitivity, malicious attacks to the trigger samples, and decision boundary of the DNN. Another similar work is [29], in which a random ternary sequence is directly added to the frequency coefficients without any special consideration. As a result of lack of frequency analysis, the watermarking performance and the model generalization of these works are limited.

Recent interpretable theoretical studies on DNNs [30], [31] have enriched our understanding of their behavior. An analysis of the DNN training process in the Fourier domain has led to the proposition of the frequency domain principle [30]. This principle posits that neural networks adapt to the objective function across a spectrum of frequencies, with a pronounced focus on lower frequencies during training. The mathematical validation is presented in [31]. Further contributions to this discourse come from investigations into the robustness of visual models within the Fourier frequency domain, as explored in [32], [33]. Notably, reference [33] delves into the relationship between model robustness and frequency perturbations, employing visualizations within the Fourier domain to elucidate key insights. Building upon these foundational studies, our work utilizes Fourier perturbations to analyze the impact of different frequencies on the performance of the original task.

To further enhance the performance of model watermarking, we propose a robust and imperceptible black-box DNN watermarking technique based on Fourier perturbation analysis and
frequency sensitivity clustering. The proposed work constructs the trigger samples by modifying the normal samples in the frequency domain, which not only enables the resulting trigger samples to not introduce noticeable artifacts, but also provides good robustness for watermark embedding and retrieval. Moreover, we have analyzed different methods for label assignment of trigger samples and found that by assigning a new class to each trigger sample, the task performance of the DNN can be kept very well after model training. Experimental results also verify the applicability and superiority. In summary, our main contributions can be summarized as follows:

- We propose an interpretable model watermarking method. By analyzing model robustness using Fourier transform and Fourier heat maps, we present a robust and imperceptible black-box watermarking system, which combines model robustness analysis with image frequency analysis, rendering model watermarking interpretable.
- The proposed method embeds low-intensity trigger perturbations into the low-frequency components within the sensitive frequencies of image samples, which enhances the robustness and imperceptibility of the watermark.
- Extensive evaluations of our method are conducted across multiple datasets and DNN models. Experimental results indicate that the watermark can be successfully embedded and reliably extracted without impairing the original task of the DNN. Meanwhile, the proposed work has achieved satisfactory performance against common attacks such as fine-tuning, pruning and trigger-sample pre-processing attacks. Comparative analysis with related studies indicates that our work achieves superior performance in terms of robustness and imperceptibility of the watermark.

II. PROPOSED METHOD

A. Overview

This study investigates black-box watermarking for securing image classification models. When an assailant deploys a counterfeit model onto a cloud server or an embedded device, the lawful owner is impeded from accessing the internal parameter information of the illicit model. Verification of ownership through white-box watermarking becomes unattainable in such instances. Black-box model watermarking, conversely, necessitates interaction with the targeted black-box threat model to extract the corresponding output for ownership verification. This implies that black-box model watermarking can ascertain the model’s copyright without requiring access to the internal information during the verification phase. Such an approach proves more suitable for practical applications. The black-box threat model or the suspicious model represents either the attacker’s crafted proxy model or the model subsequent to a modification attack.

As shown in Fig. 1, the proposed framework includes three phases, i.e., trigger sample generation, watermark embedding and ownership verification. Trigger sample generation will be used for watermark embedding and ownership verification. The watermark embedding process is realized by training the DNN with the normal samples and the trigger samples. During training, the normal samples are associated with their own normal labels. However, a new class will be assigned to the trigger samples. In other words, all the trigger samples will share a new label that does not appear in the normal-label set. After training, the DNN model is deemed marked and will be put into use. For ownership verification, a set of trigger samples will be fed into the target model to obtain the prediction results. By analyzing the prediction results, the ownership of the target model can be identified.

B. Threat Model

For a more comprehensive elucidation of the threat model in model watermarking, we simulate the roles of the model owner and the attacker. The model owner possesses a deep model $M_1$ designed for a specific task $t$, while the attacker illicitly acquires this model $M_1$ to construct a stolen model $M_2$ targeting a task $t'$ closely resembling the original task $t$. It is worth noting that the way an attacker obtains $M_1$ may be different, and it may
involve insider leaks or malicious acquisition through malware. The specific details of how the attacker obtains model $M_1$ are not discussed in this context.

If the attacker presents a threat model $M_2$ equivalent to $M_1$, we can infer that the attacker has illicitly copied the model $M_1$ from the owner, where $t'$ corresponds to the copied service of $t$. Some existing white-box backdoor watermarking methods aim to safeguard the copyright of DNNs by assessing the equivalence of $M_2$ to $M_1$. However, these methods necessitate white-box access to $M_2$, which is often unfeasible, as plagiarizers typically do not deploy $M_2$ as a server and offer public white-box access. Additionally, we consider scenarios where a plagiarist may modify model $M_1$ while preserving the performance of $t'$ to make it approximate $t$.

Potential adversaries may exploit watermarking attack methods to compromise the validity of model watermark verification. Common watermarking attack strategies include model modification attacks and input modification attacks. Model modification attacks involve altering the parameters or structure of the model containing the watermark, with model fine-tuning and model pruning being prevalent methods. On the other hand, input modification attacks involve an attacker attempting to compromise potential trigger patterns within input samples, aiming to induce the model to produce results incongruent with its designated target labels, thereby causing the model’s copyright validation to fail. For instance, in the case of image inputs, typical operations for input modification encompass image reconstruction using a self-encoder, noise addition, input quantization, image smoothing, image flipping, JPEG compression, and similar techniques. Consequently, our proposed solution must demonstrate resilience against model fine-tuning, model pruning, and input modification attacks.

To tackle the aforementioned issues, we propose a black-box watermarking approach aimed at safeguarding model copyrights. This method assists model owners in verifying whether the service $t'$ originates from model $M_1$, all without necessitating white-box access to the threat model $M_2$.

### C. Watermark Embedding

Mathematically, let $M_0$ represent the host DNN, which has been trained on a normal dataset $D$ that consists of a number of sample-and-label pairs $\{(x_i, y_i) | 1 \leq i \leq |D|\}$. Here, $|*|$ represents the total number of elements in a set. By limiting $M_0$ to image classification, we can write $x_i \in \mathbb{R}^{h \times w \times d}$ and $y_i \in \{0, 1, \ldots, c-1\}$ for $1 \leq i \leq |D|$, where $h \times w \times d$ indicates the size of the image and $c$ gives the total number of classes on the normal dataset. Assuming that we have already generated two sets of trigger samples $T_1 = \{(x'_i, y'_i) | 1 \leq i \leq |T_1|\}$ and $T_2 = \{(x''_i, y''_i) | 1 \leq i \leq |T_2|\}$ according to the proposed trigger sample generation algorithm, the watermark embedding phase requires us to train such a model $M_1$ from scratch based on $D$ and $T_1$ so that $M_0$ and $M_1$ have the same generalization on “unseen” dataset $U = \{(x_i, y_i') | 1 \leq i \leq |U|\}$, which can be roughly measured from a statistical perspective as

$$\frac{1}{|U|} \sum_{i=1}^{|U|} \delta(M_0(x'_i), y'_i) - \frac{1}{|U|} \sum_{i=1}^{|U|} \delta(M_1(x'_i), y'_i) \leq \epsilon,$$  

where $0 \leq \epsilon \leq 1$, $\delta(x, y) = 1$ if $x = y$ otherwise $\delta(x, y) = 0$, and $M_i(e)$ is the classification result after feeding $e \in U$ into $M_i$, $i \in \{0, 1\}$. It should be remarked that (1) uses the final prediction result to measure the difference between the generalization ability of two models. It is not rigorous enough, but reasonable for black-box model watermarking since under black-box scenarios, when disputes arise, the model owner can only obtain the final output of the target model. In order to be consistent with the subsequent verification process, we use (1) to measure the difference between the generalization ability of two models. It is noted that it is possible that $T_1 \cap T_2 \neq \emptyset$, e.g., $T_1 = T_2$ is used for [26]. It is required that $M_1$ has a high prediction accuracy on $T_2$:

$$1 - \frac{1}{|T_2|} \sum_{i=1}^{|T_2|} \delta(M_1(x''_i), y''_i) \leq \epsilon.$$  

As mentioned in Section II-A, the proposed work assigns a new class tag to all the trigger samples, i.e., we can write $y'_1 = y'_2 = \ldots = y'_{|T_1|} = y''_1 = y''_2 = \ldots = y''_{|T_2|} = c$. In brief summary, the proposed method extends the label set from $\{0, 1, \ldots, c-1\}$ to $\{0, 1, \ldots, c\}$, which is inspired by the novel method introduced in [17]. It should be admitted that it is always free for us to design the labeling strategy, which is not the main contribution of this paper. In our experiments, we will analyze the performance of different labeling strategies. After model training, the resulting model $M_1$ is treated as a marked version of $M_0$ and will be put into use.

### D. Ownership Verification

Assuming that $M_1$ has been leaked and probably tampered, we are to verify the ownership of the leaked version $M_2 \approx M_1$ by querying the classification results of $M_2$ for the trigger samples in $T_2$. Formally, the ownership can be verified if

$$1 - \frac{1}{|T_2|} \sum_{i=1}^{|T_2|} \delta(M_2(x''_i), y''_i) \leq \epsilon',$$  

where $0 \leq \epsilon' \leq 1$ is a pre-determined threshold. Otherwise, the ownership verification is deemed failed. From a more general perspective, the existing methods often assume that only the DNN model may be attacked, in this paper, we will further investigate the possible attack on the trigger set, i.e., the trigger sample $x''_i$ in (3) may be attacked prior to verifying the ownership. It is required that any DNN watermarking method should still resist such kind of attack, which, however, has not been considered in many existing methods.

Therefore, if we consider the attack to the trigger samples, such as input modification, we should rewrite (3) as:

$$1 - \frac{1}{|T_2|} \sum_{i=1}^{|T_2|} \delta(M_2(x''''_i), y''''_i) \leq \epsilon',$$  

where $x''''_i$ is the attacked version of $x''_i$. Our work is based on (4), which has better applicability. In other words, during the verification phase, when utilizing the trigger samples as inputs, the original trigger samples undergo a transformation resulting in $x''''_i$ after being subjected to an input modification attack. If
the original trigger samples undergo model modification attacks such as fine-tuning and pruning, without being subjected to input modification attacks, the trigger samples in the trigger set will be unchanged, i.e., $x_i''$ in (4) is equivalent to $x_i'$, making (4) identical to (3).

E. Trigger Sample Generation

It has been widely demonstrated by the conventional image watermarking methods that embedding secret information in the frequency domain has good robustness and can keep the visual quality of image well. Embedding secret information in the frequency domain is actually equivalent to adding a kind of perturbation in the frequency domain. Inspired by this key insight, a straightforward idea to generate the trigger sample is adding perturbation to the normal sample (or called clean sample) in the frequency domain, which raises two problems. The first problem is where to add the perturbation, i.e., what frequency components should be used to add the perturbation. The second problem is how to generate the perturbation. For the second problem, it is easy and free for us to design the perturbation, e.g., the off-the-shelf method is adding Gaussian noise. Therefore, our main task is to solve the first problem.

On the one hand, in image watermarking, embedding information in the low frequency region provides good robustness but may introduce noticeable distortion. On the other hand, embedding information in the high frequency region keeps the perceptual distortion low, but the robustness is not satisfactory. Therefore, a good balance between robustness and imperceptibility is to embed secret information in the mid-low frequency region. Since the trigger samples in this paper are images, it is also very suitable to add perturbation in the mid-low frequency region so that the perturbed samples, i.e., the trigger samples, not only have good visual quality but also can resist malicious attacks. However, we have to analyze the impact caused by frequency perturbation on the DNN since the trigger samples will be used for DNN training. Therefore, it is naturally to ask such a question: is adding perturbation in the mid-low frequency region also good for DNN training?

The aforementioned question actually requires us to analyze the impact of different frequency components of input samples on the performance of the host DNN on its original task by adding some specific perturbation in the frequency domain. We hope to find such frequency components that they not only ensure a good balance between perturbation robustness and perturbation imperceptibility, but also ensure a good balance between watermarking robustness and model generalization. The perturbation robustness means that even the trigger samples were attacked, the perturbation in the trigger samples can be still perceived by the DNN for ownership verification. The perturbation imperceptibility means that the perturbation added to the trigger samples should not introduce significant distortion to the clean samples. The watermarking robustness means that the ownership can be still reliably verified even the marked model was attacked. The model generalization means that the performance of the DNN on its original task after watermarking should not be impaired. It is pointed that we cannot completely separate perturbation robustness (imperceptibility) from watermarking robustness (imperceptibility) because they are entangled with each other. Here is just to facilitate analysis. In addition, good perturbation robustness also indicates that the marked DNN model has good robustness against attacks.

Below, we are to demonstrate that adding the perturbation in the mid-low frequency region is a good choice for constructing the trigger samples to balance the above requirements.

1) Fourier Perturbation Analysis: Recent theoretical results [30], [31] in deep learning indicate that DNNs always fit target functions from low to high frequencies during model training. This implies that DNNs are more robust to perturbation in the low frequency compared with high frequency in most cases. Advances in computer vision such as [32], [33] also show that model robustness in computer vision can be explained from a frequency perspective. It is demonstrated that by investigating model performance and perturbation in the frequency domain, connections between frequency perturbation and model performance can be built, which will be very helpful for us to study robust and imperceptible black-box DNN watermarking in this paper. Inspired by the aforementioned works and analysis, we exploit Fourier perturbation to analyze the impact of different frequencies on the performance of the original task of $M_0$.

The dataset $D$ mentioned in Section II-C can be partitioned into two disjoint subsets $D_1$ (training set) and $D_2$ (validation set) for training $M_0$. During training, after a certain number of epochs, we apply the Fourier perturbation to $D_2$ to evaluate the prediction performance of $M_0$ on the perturbed samples. In detail, given a clean example $c \in \mathbb{R}^{h \times w \times d}$ (which contains $d$ channels), we can apply 2D Discrete Fourier Transform (DFT) $\mathcal{F}: \mathbb{R}^{h \times w} \rightarrow \mathbb{C}^{h \times w}$ to each channel of $c$, i.e.,

$$z_k = \mathcal{F}(c_k), \forall 1 \leq k \leq d,$$

where $c_k \in \mathbb{R}^{h \times w}$ is the $k$-th channel of $c$. By applying the inverse 2D DFT, we are able to reconstruct $c_k$ from $z_k$, i.e.,

$$c_k = \mathcal{F}^{-1}(z_k), \forall 1 \leq k \leq d.$$  

We define $c_{i,j,k}$ as the element at position $(i,j)$ in $c_k$ whose value is $c_{i,j,k} \in \mathbb{R}$. When we visualize the frequency characteristics, we always shift the lowest frequency components to the center of the spectrum. Let $e_{i,j,k} \in \mathbb{R}^{h \times w}$, $k \in [1,d]$, be such a real-valued matrix that the $l_2$ norm of $e_{i,j,k}$ is equal to 1, i.e., $||e_{i,j,k}||_2 = 1$, and $\mathcal{F}(e_{i,j,k})$ has at most two non-zero elements at position $(i,j)$ and the symmetric coordinate with respect to the image center. All $\mathcal{F}(e_{i,j,k})$ are typically called 2D Fourier basis matrices [33], [34]. Given $c$, we can generate a perturbed sample $\tilde{c}_{i,j}$ with Fourier basis noise for position $(i,j)$, each component of which is expressed as:

$$\tilde{c}_{i,j,k} = \mathcal{F}^{-1}(\mathcal{F}(c_k) + \lambda_{i,j,k} \cdot \mathcal{F}(e_{i,j,k})), $$

where $1 \leq k \leq d$ and $\lambda_{i,j,k} \in \mathbb{R}$ is called perturbation coefficient whose absolute value means the perturbation intensity. We can evaluate the prediction performance of $M_0$ on its original task with a number of perturbed samples and visualize how the test error changes as a function of position $(i,j)$. The visualized result is called Fourier heat map of the model $M_0$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
We use ResNet-56 [35] evaluated on the CIFAR-10 dataset [36] consisting of 50,000 training images and 10,000 testing images in 10 classes, to perform the Fourier heat map analysis. In the experiments, we did not touch the testing images and split the training images into two disjoint subsets: training set (90%) and validation set (10%). The perturbation coefficients were randomly sampled from $[-1, 1]$. Fig. 2 shows the Fourier heat maps by first training $M_0$ on the training set with various epochs from scratch and then calculating the test errors on the perturbed validation set. In Fig. 2, when the color is closer to red, the corresponding test error is closer to 1. When the color is closer to blue, the corresponding test error is closer to 0. It can be observed that when the epoch number increases, the central low-frequency region preferentially gradually becomes blue from red. Afterwards, the corner high-frequency gradually becomes light blue from red as well.

Therefore, we can conclude that: adding perturbation does not change the frequency principle [30], i.e., the DNN fits the target function from low to high frequencies during training. Compared with higher frequency information, lower frequency information contributes more to the fitting of the model. By adding a trigger signal (which is a kind of perturbation) in the mid-low frequency region, the mapping relationship between the trigger samples and the corresponding assigned labels can be learned better. In other words, adding mid-low frequency perturbation is very good for trigger sample construction.

2) Frequency Sensitivity Clustering: However, as shown in Fig. 2, different frequencies in the mid-low frequency region result in different test errors. On the one hand, this is surely affected by the image content and training strategy, e.g., the Fourier heat map may reasonably change due to different training parameters even though the frequency principle will still hold. On the other hand, it is required that the watermarking operation should not impair the performance of the DNN on its original task, namely, it is not suggested to perturb those mid-low frequencies that have very low test errors (e.g., the dark-blue region in Fig. 2) because these low-test-error frequencies are very important for the DNN to fit the original task and “a low test error” implies that the DNN has adequately fitted the original task. If we modify those frequencies with very low test errors (e.g., the dark-blue region in Fig. 2) during watermark embedding, it forces the DNN to learn the watermarking task, rather than the original task, which will significantly degrade the performance of the DNN on its original task.

Therefore, based on the aforementioned analysis, we conclude that in the mid-low frequency region, it is more desirable to use those frequencies with relatively higher test errors for trigger sample generation because they result in a good trade-off between the watermarking task and the original task. One may empirically choose some mid-low frequencies in specific positions out for trigger sample generation. However, from the point of view of algorithmic design, it is very necessary to find a general way to determine the frequencies suitable for trigger sample generation, which motivates us to propose a frequency sensitivity clustering method in this paper.

Mathematically, let $t \in [0, 1]^{h \times w}$ represent the Fourier heat map, for which $t_{i,j}$ is the test error for the frequency position $(i, j)$. It is pointed that the lowest frequency components have been shifted to the center of the spectrum for better analysis, and $|t_{i,j}| = |t_{i',j'}|$ holds if the two positions $(i, j)$ and $(i', j')$ are symmetrical with respect to the center of the spectrum.

First of all, we determine a sensitivity map $s = \{0, 1\}^{h \times w}$

$$s_{i,j} = \begin{cases} 1 & t_{i,j} \geq \rho \\ 0 & \text{otherwise}, \end{cases} \quad (8)$$

where $\rho \in [0, 1]$ is a threshold that controls the total number of sensitivity frequencies. A frequency is deemed sensitive if the corresponding element is “1” in the sensitivity map. These sensitivity frequencies, actually, are candidate frequencies for trigger sample generation. For each $s_{i,j} (= 1)$, we determine a feature vector expressed as

$$d_{i,j} = (d_{i,j,0}, d_{i,j,1}), \quad (9)$$

**Algorithm 1:** Pseudocode for Trigger Sample Generation.

**Input:** Normal dataset $D$, a set of normal samples used for trigger sample generation $S_N = \{s_1, s_2, \ldots, s_{|S_N|}\}$, $M_0$.

**Output:** A set of trigger samples $S_T = \{s'_1, s'_2, \ldots, s'_{|S_N|}\}$.

1: Divide $D$ into two subsets $D_1$ (training) and $D_2$ (validation) such that $D = D_1 \cup D_2$ and $D_1 \cap D_2 = \emptyset$
2: Train $M_0$ from scratch with $D_1$ and $D_2$
3: Perform Fourier perturbation analysis to generate a Fourier heat map with $M_0$ (trained) and $D_2$ (to be perturbed)
4: Perform frequency sensitivity clustering to finally generate a clustering map that is a binary matrix where the positions marked as “1” are used for trigger sample generation
5: for $i = 1, 2, \ldots, |S_N|$ do
6: Apply Fourier perturbation to $s_i$ to generate the trigger sample $s'_i$ based on the clustering map and (7)
7: end for
8: return $S_T = \{s'_1, s'_2, \ldots, s'_{|S_N|}\}$

**Fig. 2.** Fourier heat maps by training $M_0$ with various epochs from scratch.
where $d_{i,j,0} = 1 - t_{i,j}$ and

$$d_{i,j,1} = \left[\left(\frac{i - \bar{i}}{2}\right)^2 + \left(\frac{j - \bar{j}}{2}\right)^2\right]^{\frac{1}{2}}. \tag{10}$$

Here, $d_{i,j,1}$ actually represents the euclidean distance between $(i,j)$ and the center of the spectrum. Thereafter, we apply K-means clustering to divide all the sensitive frequencies into two disjoint subsets (namely, the number of clusters is 2) such that one subset will be used for adding perturbation but the other one will be unchanged. It is noted that the frequency at $(i,j)$ is equivalent to the frequency at the symmetrical position with respect to the center of the spectrum. The clustering bases on the aforementioned feature vector, each component of which should be normalized along the corresponding dimension prior to K-means clustering [37], e.g., after normalization, $d_{i,j,0}$ will become $(d_{i,j,0} - \min_{a,b} d_{a,b,0})/(\max_{a,b} d_{a,b,0} - \min_{a,b} d_{a,b,0})$.

After K-means clustering, two clusters can be obtained. Fig. 3 provides some visual examples, from which we can find that the sensitive frequencies generated by different models on same or different datasets are different from each other, that is, disturbances at different frequencies have different influences on the model. The cluster with the lower average euclidean distance to the center of the spectrum will be used for trigger sample generation. The other one will be unchanged. The most important advantage of using K-means clustering is that it is model dependent, i.e., the generated perturbation considers the influence caused by the model. It is significantly different from many existing methods that directly insert a noticeable signal into a clean sample in the spatial or frequency domain, and makes a step towards interpretable DNN watermarking.

According to the aforementioned analysis, we are now ready to describe the steps of trigger sample generation, which are provided in Algorithm 1. Fig. 1(a) also shows an example for sensitivity map and clustering map. In the clustering map, the white positions constitute the cluster used for trigger sample generation. In the next section, we will conduct experiments and analysis to verify the superiority and applicability.

**Remark:** In Line 6 of Algorithm 1, $s'_1$, $s'_2$, ..., $s'_{S_N}$ should use the same perturbation, namely, in (7), the perturbation term $\lambda_{i,j,k} \cdot e_{(i,j),k}$ for $s'_1$, $s'_2$, ..., $s'_{S_N}$ are equal to each other, which facilitates learning the trigger pattern. Notice that, for a single trigger sample, different sensitive frequencies will use different perturbations controlled by a secret key.

### III. Experimental Results and Analysis

#### A. Setup

We used three popular datasets CIFAR-10, CIFAR-100 [36] and GTSRB [38] to conduct extensive experiments. CIFAR-10 and CIFAR-100 contain 60,000 images. CIFAR-10 contains 10 classes and CIFAR-100 contains 100 classes. Each of them was randomly divided into three disjoint subsets, i.e., training set (75%), validation set (5%) and testing set (20%). GTSRB contains 43 classes of traffic signs, split into 39,209 training images and 12,630 testing images. The training images were divided into two disjoint subsets, i.e., training set (80%) and validation set (20%). The testing images were belonging to the testing set. The popular architectures VGG19 [39], ResNet-18 [35], ResNet-56 [35], DenseNet-121 [40], and WideResNet-34 [41] were used as the original DNN. The proposed work is not subjected to the above models and datasets.

Without the loss of generalization, we here take VGG19 for example to explain how to generate the trigger samples, how to generate the marked model and how to verify the ownership.

- **Trigger Sample Generation:** The training set denoted by $D_1$ and the validation set denoted by $D_2$ are collected to build the dataset $D$ in Algorithm 1. $M_0$ is set to VGG19. $M_0$ can be trained with $D$ from scratch to generate the trained but non-marked model (see Line 2 in Algorithm 1). Given any normal sample, we are able to generate the corresponding trigger sample by Algorithm 1.

- **Watermark Embedding:** We randomly generate two sets $A_1 \subset D_1$ and $A_2 \subset D_2$. We apply Algorithm 1 to $A_1$ and $A_2$ to generate the corresponding trigger samples, whose labels are set to $c$. In this way, we obtain two trigger sets $B_1$ (corresponding to $A_1$) and $B_2$ (corresponding to $A_2$). Two sets $D_1 \cup B_1$ (training) and $D_2 \cup B_2$ (validation) are used to train $M_0$ from scratch to generate the trained and marked model $M_1$. It can be inferred that $T_1 = B_1 \cup B_2$ according to the definition of $T_1$ in Section II-C.
TABLE I
RELATIONSHIPS BETWEEN DIFFERENT SUBSETS

| Original dataset | Subset Trigger set |
|------------------|--------------------|
| D                | D_1               |
|                  | A_1 \setminus A_2 |
| D_1 \setminus D_2 | D_2               |
|                  | A_2 \setminus A_1 |
|                  | A_2 \setminus A_1 |
|                  | B_2 \setminus B_1 |
|                  | B_2 \setminus B_1 |

Other constraints:

| Dataset for watermark embedding | D \cup T_1 = D_1 \cup D_2 \cup B_1 \cup B_2 |
| Dataset for ownership verification | T_2 = V_2 |
| Dataset for original task evaluation | U \cup V |

TABLE II
CLASSIFICATION ACCURACY ON THE ORIGINAL TASK FOR DIFFERENT MODELS BEFORE WATERMARKING

| Model          | CIFAR-10 | CIFAR-100 | GTSRB |
|----------------|----------|-----------|-------|
| VGG19          | 91.43%   | 57.38%    | 97.31%|
| ResNet-18      | 93.95%   | 75.76%    | 97.46%|
| ResNet-56      | 85.85%   | 72.86%    | 98.28%|
| DenseNet-121   | 94.54%   | 78.63%    | 97.63%|
| WideResNet-34  | 94.79%   | 79.51%    | 98.56%|

-- Ownership Verification: We verify the ownership of the target model by (4). The key step is how to generate the trigger set T_2. To deal with this issue, we divide the testing set E into two disjoint subsets U and V. V is used to generate a set of trigger samples V_T. We set T_2 = V_T.

Table I shows the relationships between different sets.

We set |A_1| = |B_1| = |A_2| = |B_2| = |V| = |V_T| = q_t. We empirically set q_t = 500 by default in our experiments unless otherwise specified. We applied the stochastic gradient descent (SGD) optimizer to train each network. The learning rate was 0.01, with a momentum of 0.9. The batch size was 512. The perturbation coefficients were randomly sampled from [-1, 1]. The threshold \( \rho \) was set to 0.65 by default. It is always free for us to adjust these parameters to achieve better performance.

B. Fidelity

Task fidelity means that the performance of the DNN model on its original task after watermarking should be kept well. It requires us to determine the accuracy of image classification for the original model (which is deemed non-marked) and the marked model. Table II shows the classification accuracy of the non-marked model. Table III shows the classification accuracy of the marked model. It can be inferred from Table II that different models have different performance on different image classification tasks, which is reasonable due to the different learning capabilities of deep models. It can be also inferred from Table III that the accuracy declines as q_t increases from the viewpoint of overall trend, which is a normal phenomenon because the model needs to make a sacrifice on the original task for model watermarking. However, by comparing Tables II and III, we can find that the performance degradation on the original task is very low after watermarking with different q_t, which indicates that the proposed method does not impair the utilization of the model and thereby has good potential in application scenarios.

It is admitted that we did not optimize the hyper-parameters and the training strategy for all models, meaning that the baseline performance in Table II may be not optimal, which does not affect the evaluation of our work.

On the other hand, watermark fidelity means that the hidden watermark should be reconstructed accurately. Our work aligns with zero-bit watermarking [27], which is designed to detect the presence or the absence of the watermark in the marked model. We calculate the classification accuracy on the trigger set for different models with different q_t. The results are shown in Fig. 4, from which we can find that different models have different accuracy values, which means that different models have different capabilities of carrying a watermark. It is observed that all the accuracy values are no less than 80%, meaning that the proposed method enables the watermark to be reliably extracted. It can be inferred that the accuracy will increase when q_t increases in most cases. The reason is that a larger q_t means that more trigger samples are used for model training, allowing the watermarking task to be accomplished very well by the model. For example, in Fig. 4, the accuracy is approaching 100% when q_t = 1000, which has demonstrated the superiority of the proposed method.

In addition, sample fidelity means that the perturbed sample should be visually close to the original sample. Fig. 5 shows some examples of the perturbed sample. It can be seen that the proposed perturbation technique does not introduce noticeable artifacts. To quantize the visual quality, we determine the mean peak signal-to-noise ratio (PSNR, dB) and structural similarity (SSIM) [42] between the original samples in the testing set and the corresponding perturbed samples. The results are shown in Table IV, from which we can find that the mean PSNRs are

TABLE III
CLASSIFICATION ACCURACY ON THE ORIGINAL TASK FOR DIFFERENT MODELS AFTER WATERMARKING

| Dataset     | Model     | q_t |
|-------------|-----------|-----|
|             | CIFAR-10  | CIFAR-100 | GTSRB |
| VGG19       | 91.43%    | 91.36%    | 91.32% |
| ResNet-18   | 93.95%    | 93.91%    | 93.88% |
| ResNet-56   | 83.86%    | 83.79%    | 83.41% |
| DenseNet-121| 94.54%    | 94.52%    | 94.48% |
| WideResNet-34| 94.82%   | 94.86%    | 94.38% |

-- Ownership Verification: We verify the ownership of the target model by (4). The key step is how to generate the trigger set T_2. To deal with this issue, we divide the testing set E into two disjoint subsets U and V. V is used to generate a set of trigger samples V_T. We set T_2 = V_T.

Table I shows the relationships between different sets.

We set |A_1| = |B_1| = |A_2| = |B_2| = |V| = |V_T| = q_t. We empirically set q_t = 500 by default in our experiments unless otherwise specified. We applied the stochastic gradient descent (SGD) optimizer to train each network. The learning rate was 0.01, with a momentum of 0.9. The batch size was 512. The perturbation coefficients were randomly sampled from [-1, 1]. The threshold \( \rho \) was set to 0.65 by default. It is always free for us to adjust these parameters to achieve better performance.

B. Fidelity

Task fidelity means that the performance of the DNN model on its original task after watermarking should be kept well. It requires us to determine the accuracy of image classification for the original model (which is deemed non-marked) and the marked model. Table II shows the classification accuracy of the non-marked model. Table III shows the classification accuracy of the marked model. It can be inferred from Table II that different models have different performance on different image classification tasks, which is reasonable due to the different learning capabilities of deep models. It can be also inferred from Table III that the accuracy declines as q_t increases from the viewpoint of overall trend, which is a normal phenomenon because the model needs to make a sacrifice on the original task for model watermarking. However, by comparing Tables II and III, we can find that the performance degradation on the original task is very low after watermarking with different q_t, which indicates that the proposed method does not impair the utilization of the model and thereby has good potential in application scenarios.

It is admitted that we did not optimize the hyper-parameters and the training strategy for all models, meaning that the baseline performance in Table II may be not optimal, which does not affect the evaluation of our work.

On the other hand, watermark fidelity means that the hidden watermark should be reconstructed accurately. Our work aligns with zero-bit watermarking [27], which is designed to detect the presence or the absence of the watermark in the marked model. We calculate the classification accuracy on the trigger set for different models with different q_t. The results are shown in Fig. 4, from which we can find that different models have different accuracy values, which means that different models have different capabilities of carrying a watermark. It is observed that all the accuracy values are no less than 80%, meaning that the proposed method enables the watermark to be reliably extracted. It can be inferred that the accuracy will increase when q_t increases in most cases. The reason is that a larger q_t means that more trigger samples are used for model training, allowing the watermarking task to be accomplished very well by the model. For example, in Fig. 4, the accuracy is approaching 100% when q_t = 1000, which has demonstrated the superiority of the proposed method.

In addition, sample fidelity means that the perturbed sample should be visually close to the original sample. Fig. 5 shows some examples of the perturbed sample. It can be seen that the proposed perturbation technique does not introduce noticeable artifacts. To quantize the visual quality, we determine the mean peak signal-to-noise ratio (PSNR, dB) and structural similarity (SSIM) [42] between the original samples in the testing set and the corresponding perturbed samples. The results are shown in Table IV, from which we can find that the mean PSNRs are

| Dataset     | Model     | q_t |
|-------------|-----------|-----|
|             | CIFAR-10  | CIFAR-100 | GTSRB |
| VGG19       | 97.34%    | 97.30%    | 97.28% |
| ResNet-18   | 97.45%    | 97.46%    | 97.86% |
| ResNet-56   | 98.30%    | 98.27%    | 98.23% |
| DenseNet-121| 97.61%    | 97.82%    | 97.57% |
| WideResNet-34| 98.57%   | 98.99%    | 98.51% |

The representative Resnet-18 was used as the host network.
higher than 36 dB and the mean SSIMs are higher than 0.99. It indicates that the perturbed samples have very good visual quality, which can guarantee perturbation imperceptibility.

C. Robustness

Generally, robustness evaluates the ability to reconstruct the embedded watermark from the marked DNN model when the watermarking system was attacked by the adversary. Unlike many existing methods only considering attacks to the marked model, we further take into account the possible attacks to the trigger samples. In the following, we first analyze the performance of the model on the original task and the watermarking task when the marked model was attacked. Then, we analyze the performance when the trigger samples were attacked.

Two most popular attacks are applied to the marked model. One is model fine-tuning, and the other is model pruning. To mimic the fine-tuning attack, we randomly choose 50% normal samples of the validation set to fine-tune the marked model. To mimic the model pruning attack, we apply the $\ell_1$-norm pruning strategy to the marked model, where a pruning rate is used to denote the percentage of pruned parameters. We evaluate the performance of the attacked model on the original task and the watermarking task when the marked model was attacked. Then, we analyze the performance when the trigger samples were attacked.

Fig. 4. Classification accuracy on the trigger set for different models after watermarking with different $q$: (a) CIFAR-10, (b) CIFAR-100 and (c) GTTSRB.

Fig. 5. Examples of the perturbed sample: (a), (c), (e) original examples randomly selected from CIFAR-10, CIFAR-100 and GTTSRB, respectively, (b), (d), (f) the corresponding perturbed samples. The representative ResNet-18 was used.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
inferred that for both UDA and NUDA, when the quality factor (QF) gradually decreases, both $\text{Acc}_o$ and $\text{Acc}_w$ tend to decrease, which is due to the reason that a smaller QF results in more loss of feature information. Specifically, the observed phenomenon can be attributed to the inverse relationship between the quantization factor and the preservation of feature information. The inherent lossy nature of JPEG compression leads to the degradation or impairment of specific feature information within both clean samples and trigger-critical samples. As the QF decreases, there is a heightened loss of feature information. Consequently, this degradation contributes to a reduction in accuracy metrics. Consequently, the degradation results in a decrease in the performance of both UDA and NUDA. We can find that the performance difference between UDA and NUDA is small, especially for larger QFs. It indicates that the proposed method has good ability to make the trigger samples capable of resisting JPEG compression. The latest image compression literature [43] also reveals that the essence of image compression is to preserve the low-mid frequencies and filter out the high-frequency information through the visualization of Fourier heat maps, which further validates the effectiveness of our method to add perturbations in the low-mid frequency region. It is noted that both the normal sample and the trigger samples are attacked during the testing phase. Table VII provides the results against horizontal flipping under different conditions, from which we can infer that flipping does not impair watermarking because the trigger patterns are symmetrical in the frequency domain.

We further consider low-pass filtering in the DFT domain. Low-pass filtering with bandwidth $B$ is defined as the operation

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
\textbf{Dataset} & \textbf{Quality factor} & \textbf{Indicator} & \textbf{100} & \textbf{80} & \textbf{60} & \textbf{40} & \textbf{20} \\
\hline
\hline
\textbf{CIFAR-10} & NUDA & $\text{Acc}_o$ & 93.88\% & 99\% & 89.96\% & 88\% & 84.65\% & 29\% & 78.91\% & 3\% & 68.57\% & 0\% \\hline
& UDA & $\text{Acc}_o$ & 89.95\% & 97\% & 87.95\% & 86\% & 83.76\% & 35\% & 75.64\% & 5\% & 61.28\% & 0\% \\hline
\textbf{CIFAR-100} & NUDA & $\text{Acc}_o$ & 67.34\% & 95\% & 65.78\% & 82\% & 52.83\% & 22\% & 46.95\% & 2\% & 42.76\% & 0\% \\hline
& UDA & $\text{Acc}_o$ & 67.16\% & 95\% & 65.23\% & 95\% & 50.42\% & 68\% & 43.48\% & 5\% & 41.52\% & 0\% \\hline
\textbf{GTRSB} & NUDA & $\text{Acc}_o$ & 97.28\% & 98\% & 96.46\% & 85\% & 88.89\% & 28\% & 72.37\% & 3\% & 67.06\% & 0\% \\hline
& UDA & $\text{Acc}_o$ & 97.05\% & 98\% & 96.37\% & 98\% & 88.78\% & 78\% & 70.96\% & 8\% & 66.83\% & 0\% \\hline
\end{tabular}
\caption{Performance Against JPEG Compression Under Different Conditions for VGG19}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
\textbf{Dataset} & \textbf{Quality factor} & \textbf{Indicator} & \textbf{100} & \textbf{80} & \textbf{60} & \textbf{40} & \textbf{20} \\
\hline
\hline
\textbf{CIFAR-10} & NUDA & $\text{Acc}_o$ & 93.88\% & 99\% & 89.96\% & 88\% & 84.65\% & 29\% & 78.91\% & 3\% & 68.57\% & 0\% \\hline
& UDA & $\text{Acc}_o$ & 89.95\% & 97\% & 87.95\% & 86\% & 83.76\% & 35\% & 75.64\% & 5\% & 61.28\% & 0\% \\hline
\textbf{CIFAR-100} & NUDA & $\text{Acc}_o$ & 67.34\% & 95\% & 65.78\% & 82\% & 52.83\% & 22\% & 46.95\% & 2\% & 42.76\% & 0\% \\hline
& UDA & $\text{Acc}_o$ & 67.16\% & 95\% & 65.23\% & 95\% & 50.42\% & 68\% & 43.48\% & 5\% & 41.52\% & 0\% \\hline
\textbf{GTRSB} & NUDA & $\text{Acc}_o$ & 97.28\% & 98\% & 96.46\% & 85\% & 88.89\% & 28\% & 72.37\% & 3\% & 67.06\% & 0\% \\hline
& UDA & $\text{Acc}_o$ & 97.05\% & 98\% & 96.37\% & 98\% & 88.78\% & 78\% & 70.96\% & 8\% & 66.83\% & 0\% \\hline
\end{tabular}
\caption{Performance Against JPEG Compression Under Different Conditions for ResNet-18}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline
\textbf{Dataset} & \textbf{Model} & \textbf{VGG19} & \textbf{ResNet-18} \\
\hline
\hline
\textbf{CIFAR-10} & NUDA & 93.88\% & 97\% & 93.68\% & 99\% \\hline
& UDA & 91.15\% & 97\% & 93.75\% & 99\% \\hline
\textbf{CIFAR-100} & NUDA & 67.34\% & 95\% & 75.72\% & 99\% \\hline
& UDA & 66.29\% & 95\% & 75.38\% & 99\% \\hline
\textbf{GTRSB} & NUDA & 97.28\% & 98\% & 96.78\% & 99\% \\hline
& UDA & 97.17\% & 98\% & 96.43\% & 99\% \\hline
\end{tabular}
\caption{Performance Against Horizontal Flipping}
\end{table}
of setting all frequency components outside the center square of width B in the Fourier spectrum at the center lowest frequency to zero and then applying the inverse DFT to the spectrum. Tables VIII and IX report the performance against low-pass filtering under different conditions for VGG19 and ResNet-18. It can be inferred that the classification accuracy on the trigger set achieves more than 80% when the bandwidth is larger than 12. However, when the bandwidth is smaller than 12, it causes a significant drop of the classification accuracy for both the original mission and the watermarking task. This is because our method perturbs mid-low frequency components. When the trigger samples pass through a large filter, although some high-frequency information is lost, the model can still learn the trigger. The image content can be learned by the model as well. Therefore, both the original task and the watermarking task are performed well. However, by applying a smaller filter, much information about the trigger and the image content will be lost. As a result, the performance on both tasks will be surely declined.

### D. Comparisons With Previous Methods

One of the key contributions of this paper is that we propose a new method to generate the trigger samples. It is necessary to compare the visual quality of the trigger samples generated by different methods. To this purpose, we compare the proposed method with three most representative methods, i.e., unrelated samples (URS), logo based trigger (LBT), noise based trigger (NBT). In detail, URS uses images that are not related to the original task of the host model as the triggers [15], [16]. LBT simply adds a logo such as character(s) and specific pattern to a clean image to construct the trigger sample [16]. NBT uses a pre-defined noise to play the role of the trigger [16], [17].

The above strategies can be all applied to model verification through analyzing the prediction results of the target model given a certain number of trigger samples. However, in terms of imperceptibility, the above strategies are not desirable for application scenarios. Fig. 8 provides some trigger samples obtained by different methods. Although URS may not significantly impair the performance of the host model on its original task, the irrelevant image content may arouse suspicion of the attacker. LBT and NBT introduce noticeable artifacts in the trigger image, which will impair imperceptibility. Moreover, the noticeable artifacts may expose how the trigger signal was constructed, thereby threatening security. As shown in Fig. 8, the visual difference between the clean samples and the trigger samples generated by the proposed method is small, indicating that the proposed method keeps the imperceptibility of trigger signal very well and therefore is very suitable for practice.

### Tables VIII and IX

**Table VIII**

| Dataset  | Bandwidth | Indicator | 16 | 14 | 12 | 8 | 4 |
|----------|-----------|-----------|----|----|----|---|---|
|          |           | Acc_o     | Acc_w | Acc_o | Acc_w | Acc_o | Acc_w | Acc_o | Acc_w | Acc_o | Acc_w |
| CIFAR10  | UDA       | 91.32%    | 97%   | 82.85% | 87%   | 73.83% | 82%   | 36.19% | 23%   | 23.83% | 0%   |
| CIFAR100 | NUDA      | 67.34%    | 95%   | 62.28% | 81%   | 49.45% | 72%   | 27.49% | 16%   | 13.38% | 0%   |
|          | UDA       | 67.21%    | 95%   | 61.73% | 87%   | 47.64% | 78%   | 23.53% | 19%   | 11.39% | 0%   |
| GTRSB    | NUDA      | 97.28%    | 98%   | 87.79% | 81%   | 75.69% | 81%   | 46.38% | 25%   | 19.06% | 0%   |
|          | UDA       | 97.13%    | 98%   | 86.49% | 95%   | 74.91% | 85%   | 45.72% | 28%   | 18.76% | 0%   |

**Table IX**

| Dataset  | Bandwidth | Indicator | 16 | 14 | 12 | 8 | 4 |
|----------|-----------|-----------|----|----|----|---|---|
|          |           | Acc_o     | Acc_w | Acc_o | Acc_w | Acc_o | Acc_w | Acc_o | Acc_w | Acc_o | Acc_w |
| CIFAR10  | NUDA      | 95.88%    | 99%   | 84.87% | 85%   | 76.86% | 80%   | 43.31% | 17%   | 26.39% | 0%   |
| CIFAR100 | UDA       | 93.63%    | 99%   | 84.65% | 90%   | 76.57% | 84%   | 41.32% | 26%   | 25.44% | 0%   |
| GTRSB    | NUDA      | 75.72%    | 99%   | 70.80% | 83%   | 56.84% | 73%   | 38.79% | 15%   | 19.57% | 0%   |
|          | UDA       | 75.45%    | 99%   | 69.95% | 94%   | 52.91% | 81%   | 34.37% | 28%   | 17.16% | 0%   |

![Fig. 8. Trigger samples by different methods for different datasets (from top to bottom: CIFAR-10, CIFAR-100, GTRSB). Here, NBT uses Gaussian noise with zero mean and a variance of 0.01.](image-url)
difficulty of the learning for the model since in this case the model needs to learn various mappings that are not related to the original task which will impair the generalization ability of the model. Therefore, for fair comparison, for each method to be compared, all the trigger samples share the same ground-truth label specified by random in advance (and it should not be the label of the original clean sample). Some examples of trigger samples have been shown in Fig. 8. It is inferred that the proposed method not only achieves better visual quality for the trigger samples, but also shows better performance on both the original task and the watermarking task. It can be said that by adding well-designed perturbation in the frequency domain, the distortion between the clean sample and the trigger sample can be kept low. Meanwhile, as the perturbation will be spread throughout the entire spatial domain, it will better facilitate the learning of the trigger signal.

E. Ablation Study

In the proposed method, the label assigned to all the trigger samples can be either a randomly selected label or a new label. The randomly selected label should not be the one belonging to the original clean sample. We conduct experiments to evaluate the impact of using different label assignment strategies on both the original task and the watermarking task. Except for the assigned label, all the other experimental settings are same as each other. Table XII provides the experimental results from which we can find that using a new label is superior to using a randomly selected label since the classification accuracies on the two tasks for the former are significantly higher than that for the latter. The reason is that, using a randomly selected label may inevitably distort the decision boundary of the model on the original task, while using a new label maps the trigger samples to a new category so that the learning of the original task and the learning of the watermarking task can be separated to a certain extent, thereby allowing both the original task and the watermarking task to be better performed.

We further analyze the impact caused by perturbing different frequency areas. Taking ResNet-18 for explanation, Fig. 9 shows the Fourier heat maps evaluated across three datasets and the corresponding mask outcomes for distinct frequency regions. As shown in Fig. 9, our work yields diverse sensitivity distributions across various datasets and different frequencies have different sensitivities. The masks for low and mid-frequency sensitive regions are positioned proximal to the frequency center, while the distribution of high-frequency sensitive masks extends out from the frequency center. Moreover, the distribution of low-frequency sensitive masks is more concentrated, whereas the distribution of high-frequency sensitive masks is more sparse. It is necessary to analyze their impact on the performance of the model. Tables XIII and XIV have given the experimental results, from which we can find that the mean PSNRs and the mean SSIMs of the trigger samples due to high frequency perturbation are higher than that due to mid-low frequency perturbation. The reason is that mid-low frequency perturbation results in more

![Fig. 9. Fourier heat maps and different masks for ResNet-18 evaluated on different datasets.](image-url)
TABLE XII
PERFORMANCE COMPARISON FOR WATERMARKED MODELS USING DIFFERENT LABEL ASSIGNMENT STRATEGIES

| Dataset   | Model     | ResNet-18 | ResNet-56 | DenseNet-121 | WideResNet-34 |
|-----------|-----------|-----------|-----------|--------------|---------------|
|           | Strategy  | Acc_v     | Acc_w     | Acc_v     | Acc_w     | Acc_v     | Acc_w     | Acc_v     | Acc_w     | Acc_v     | Acc_w     |
| CIFAR-10  | RSL       | 91.32%    | 97%       | 93.88%    | 99%       | 83.41%    | 93%       | 94.48%    | 98%       | 93.14%    | 99%       |
| CIFAR-100 | RSL       | 64.12%    | 93%       | 74.65%    | 97%       | 70.25%    | 94%       | 76.96%    | 96%       | 77.59%    | 96%       |
| GTRSB     | RSL       | 97.56%    | 97%       | 96.72%    | 99%       | 97.49%    | 95%       | 97.32%    | 96%       | 97.78%    | 98%       |
|           | NL        | 97.28%    | 98%       | 96.72%    | 99%       | 98.23%    | 97%       | 97.57%    | 97%       | 98.51%    | 99%       |

"RSL" is short for "randomly selected label", and "NL" is short for "new label".

TABLE XIII
PERFORMANCE COMPARISON BETWEEN DIFFERENT PERTURBATION STRATEGIES FOR VGG19

| Dataset   | Strategy | Acc_v | Acc_w | PSNR | SSIM |
|-----------|----------|-------|-------|------|------|
| CIFAR-10  | Mid-low  | 91.32%| 97%   | 38.65| 0.9854|
|           | High     | 90.76%| 96%   | 39.81| 0.9864|
| CIFAR-100 | Mid-low  | 67.34%| 95%   | 36.62| 0.9851|
|           | High     | 65.81%| 93%   | 37.94| 0.9873|
| GTRSB     | Mid-low  | 97.28%| 98%   | 36.64| 0.9838|
|           | High     | 97.22%| 96%   | 38.58| 0.9843|

PSNR and SSIM use mean values.

TABLE XIV
PERFORMANCE COMPARISON BETWEEN DIFFERENT PERTURBATION STRATEGIES FOR RESNET-18

| Dataset   | Strategy | Acc_v | Acc_w | PSNR | SSIM |
|-----------|----------|-------|-------|------|------|
| CIFAR-10  | Mid-low  | 93.88%| 99%   | 36.14| 0.9862|
|           | High     | 92.72%| 97%   | 39.27| 0.9885|
| CIFAR-100 | Mid-low  | 75.72%| 99%   | 37.46| 0.9927|
|           | High     | 74.29%| 96%   | 38.65| 0.9954|
| GTRSB     | Mid-low  | 96.78%| 99%   | 38.53| 0.9935|
|           | High     | 96.63%| 98%   | 40.84| 0.9971|

PSNR and SSIM are mean values.

IV. CONCLUSION AND DISCUSSION

Protecting the intellectual property of deep models under the black-box condition against infringement is a very important and challenging problem. How to ensure imperceptibility and robustness of black-box DNN model watermarking is urgently to be solved. Existing methods use trigger samples to achieve black-box model watermarking. Although they can be used for verifying the ownership of the target model, they introduce noticeable visual distortion into the trigger samples. It impairs the imperceptibility of the embedded watermark. Moreover, these methods do not take into account attacks applied to the trigger samples. As a result, the robustness of the watermark is limited. To deal with the above problems, this paper presents a novel method for black-box DNN model watermarking by applying Fourier perturbation analysis and frequency sensitivity clustering. By crafting the trigger samples in the frequency domain, both the original task and the watermarking task can be better performed, which has been verified by our extensive experiments. Additionally, the trigger generation strategy introduced in this paper takes into account the influence caused by the model, which makes the proposed method interpretable.

On the other hand, it should be also admitted that we cannot ensure that the proposed watermarking system is robust against all the real-world attacks because it is surely impossible for us to foresee all the attacks performed by the adversary. Actually, even for the existing methods, they only resist specific attacks.

distortion in the spatial domain while high frequency perturbation generally distorts local details of an image. However, from the viewpoint of functionality, it can be observed that using mid-low frequency perturbation achieves better performance on the original task and the watermarking task. It is due to the reason that mid-low frequency perturbation can better facilitate model learning, which has been analyzed in the previous section.

We also analyze the influence of different clustering methods on the proposed method. Taking ResNet-18 evaluated on CIFAR-10 as an example, Fig. 10 provides the experimental results due to different clustering methods, including K-means [37], Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [44] and spectral clustering [45]. The above parameter $\rho$ controls the total number of sensitive frequencies. We set $\rho = 0.25, 0.50, 0.75$ for the first, second and third rows in Fig. 10, respectively. As shown in Fig. 10, if $\rho$ is small, the results obtained by different clustering methods are slightly different from each other. As $\rho$ increases, the results obtained by different methods become the same. This indicates that the performance of different clustering methods are close to each other when to apply them to the proposed method. In terms of practical use, it is suggested to use the clustering method with a low computational complexity. That is why we use K-means.
Therefore, in the future, we will improve the robustness of the proposed method by taking into account more attacks. In addition, the proposed method combines model interpretability with trigger samples to protect the model copyrights, which has good applicability. It is noted that different types of models use different sample data and have different results when using interpretable methods. This paper designs interpretable model watermarks based on the sensitive features obtained by the frequency-domain interpretable method. In the future, we will explore different interpretable methods to design different types of watermarks with better interpretability. We hope this attempt can inspire more advanced works.

REFERENCES

[1] X. Zhao, Q. Liu, H. Zheng, and B. Y. Zhao, “Towards graph watermarks,” in Proc. ACM Int. Conf. Online Soc. Netw., 2015, pp. 101–112.
[2] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Sato, “Embedding watermarks into deep neural networks,” in Proc. ACM Int. Conf. Multimedia Retrieval, 2017, pp. 269–277.
[3] H. Wu, G. Liu, Y. Yao, and X. Zhang, “Watermarking neural networks with watermarked images,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 7, pp. 2591–2606, Jul. 2021.
[4] J. Wang, H. Wu, X. Zhang, and Y. Yao, “Watermarking in deep neural networks via error back-propagation,” in Proc. IET Electron. Imag., Media Watermarking, Secur. Forensics, 2020, pp. 22–1–22–9.
[5] Y. Li, B. Tondi, and M. Barni, “Spread-transform dither modulation watermarking of deep neural network,” J. Inf. Secur. Appl., vol. 63, 2021, Art. no. 103804.
[6] L. Fan, K. W. Ng, and C. S. Chan, “Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks,” in Proc. Neural Inf. Process. Syst., 2019, pp. 4716–4725.
[7] H. Chen, B. D. Rouhani, C. Fu, J. Zhao, and F. Koushanfar, “DeepMarks: A secure fingerprinting framework for digital rights management of deep learning models,” in Proc. ACM Int. Conf. Multimedia Retrieval, 2019, pp. 105–113.
[8] X. Zhao, Y. Yao, H. Wu, and X. Zhang, “Structural watermarking to deep neural networks via network channel pruning,” in Proc. IEEE Int. Workshop Inf. Forensics Secur., 2021, pp. 1–6.
[9] E. Rodriguez-Lois and F. Pérez-González, “Towards traitor tracing in black-and-white-box DNN watermarking with tardos-based codes,” in Proc. IEEE Int. Workshop Inf. Forensics Secur., 2023, pp. 1–6.
[10] P. Lv et al., “A robustness-assured white-box watermark in neural networks,” IEEE Trans. Dependable Comput., vol. 20, no. 6, pp. 5214–5229, Nov./Dec. 2023.
[11] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009. [Online Available] https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf
[12] S. Zagoruyko and N. Komodakis, “Wide residual networks,” 2016, arXiv:1605.07146.
[13] L. Deng and X. Z. Chang, “Embed watermarking neural network with compensation mechanism,” in Proc. Int. Conf. Knowl. Sci., Eng. Manage., 2020, pp. 363–375.
[14] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, “Turning your weakness into a strength: Watermarking deep neural networks by backdooring,” in Proc. 27th USENIX Conf. Secur. Symp., 2018, pp. 1615–1631.
[15] J. Zhang et al., “Protecting intellectual property of deep neural networks with watermarking,” in Proc. ACM ASIA Conf. Comput. Commun., 2018, pp. 159–172.
[16] Q. Zhong, L. Zhang, J. Zhang, L. Gao, and X. Yang, “Protecting IP of deep neural networks with watermarking: A new label helps,” in Proc. Pacific-Asia Conf. Adv. Knowl. Discov. Data Mining, 2020, pp. 462–474.
[17] G. Luo, A. B. J. Teoh, Y. Xiang, and H. Jiang, “Unambiguous and high-fidelity backdoor watermarking for deep neural networks,” IEEE Trans. Neural Netw. Learn. Syst., to be published, doi: 10.1109/TNNLS.2023.3250210.
[18] G. Hua and A. B. J. Teoh, “Deep fidelity in DNN watermarking: A study of backdoor watermarking for classification models,” PatternRecognit., vol. 144, 2023, Art. no. 109844.
Yong Liu received the MS degree from the Southwest Jiaotong University, Chengdu, China, in 2018. He is currently working toward the PhD degree with the Shanghai University, Shanghai, China. His research interests include model watermarking and artificial intelligence security.

Hanzhou Wu (Member, IEEE) received the BS and PhD degrees from the Southwest Jiaotong University, Chengdu, China, in 2011 and 2017, respectively. From October 2014 to October 2016, he was a visiting scholar with the New Jersey Institute of Technology, New Jersey, USA. He was a research staff with the Institute of Automation, Chinese Academy of Sciences, Beijing, China, from 2017 to 2019. He is currently an associate professor with Shanghai University, Shanghai, China. His research interests include steganography, steganalysis and digital watermarking. He has authored/co-authored more than 100 research papers and 4 book chapters. He served as the local organization chair of 14th IEEE International Workshop on Information Forensics and Security, the steering committee member of 14th/15th/16th International Conference on Advances in Multimedia, and the technical committee member of Multimedia Security and Forensics of Asia-Pacific Signal and Information Processing Association (APSIPA). He was also awarded 2022 CCF-Tencent Rhino-Bird Young Faculty Open Research Fund. He once won three Silver Medals and two Bronze Medals as a contestant in ACM-ICPC Asia Regional Programming Contests and Invitational Programming Contests. He was also selected to participate in Yahoo! Hack Beijing 2013 Final as a contestant based on technical merit.

Xinpeng Zhang received the BS degree from Jilin University, China, in 1995, and the MS and PhD degrees from Shanghai University, in 2001 and 2004, respectively. Since 2004, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a full-time professor. He was with The State University of New York at Binghamton as a visiting scholar from 2010 to 2011, and also with Konstanz University as an experienced researcher, sponsored by the Alexander von Humboldt Foundation from 2011 to 2012. His interests include multimedia security, image processing, and digital forensics. He has published more than 200 papers. He was an associate editor for IEEE Transactions on Information Forensics and Security from 2014 to 2017.