IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

DeepTrigger: A Watermarking Scheme of Deep Learning Models based on Chaotic Automatic Data Annotation

Ying-Qian Zhang[1,2], Yi-Ran Jia[2,1*], Xing-Yuan Wang[3], Qiong Niu[1,2], Nian-Dong Chen[4*]

[1]School of Information Science & Technology, Xiamen University Tan Kah Kee College, Zhangzhou 363105, China
[2]School of Informatics, Xiamen University, Xiamen 361005, China
[3]Information Science & Technology College, Dalian Maritime University, Dalian 116023, China
[4]New Huadu Business School, Minjiang University, Fuzhou 350108, China

Corresponding author: Yi-Ran Jia (e-mail: 23020181154213@stu.xmu.edu.cn), Nian-Dong Chen (e-mail: 171944938@qq.com)

This research is supported by the Natural Science Foundation of Fujian Province of China (No.2018J01100), the National Natural Science Foundation of China (No: 61672124), the Password Theory Project of the 13th Five-Year Plan National Cryptography Development Fund (No: MMJJ20170203), Liaoning Province Science and Technology Innovation Leading Talents Program Project (No: XLYC1802013), Key R&D Projects of Liaoning Province (No: 2019020105-JH2/103).

ABSTRACT With the rapid development of artificial intelligence, the intellectual property protection of deep learning models appeals widespread concerns of scientists and engineers. The black-box watermarking protection scheme has been favored by many scholars due to its many advantages. The trigger set containing data content and data annotation is the key of black-box watermarking technology. However, most of the trigger sets in literates were constructed by comprehensible features, such as Gaussian noise and badges on original data content. Then, the attacks based on machine learning can obtain the watermarking features and generate fake trigger set. Therefore, fraudulent ownership claim attacks may occur. In this paper, we turn our attention to data annotation and propose a black-box watermarking scheme based on chaotic automatic data annotation. Chaos has superior features, such as the sensitivity of initial value, aperiodic behavior and unpredictability of the chaotic sequence. We applies these chaotic features on data annotation so as to against the fraudulent ownership claim attacks. Firstly, this scheme applies chaotic automatic data annotation, which is time-saving and non-manual labeling. Secondly, chaotic sequences are unpredictable for long-terms, which can break the principle of empirical or statistical machine learning based attacks when chaotic labeling the trigger samples. Thirdly, the initial value and parameters in chaos offer a large range of key space, which can facilitate the commercialization of the intelligent models. The key formulation also guarantees the separation of the secret key and the trigger set. In addition, experiments and simulations show that the scheme is effective, secure and robust. It can resist fine-tuning attacks, compression attacks, fraudulent ownership claim attacks and overwriting attacks.

INDEX TERMS Black-box watermarking; trigger-set; chaos; automatic annotation; non-generalization; intellectual property protection.

I. INTRODUCTION

Due to the heavy work of labeling and training high-performance deep learning models and the increasing importance of these models in AI industries, intellectual property (IP) of deep learning models appeals great interests of scientists and engineers. The high-performance deep learning models, assisting society in saving time and manual labors, are in threats of being stolen and abuse. Nowadays, these deep learning models are lack of IP protections. Without related protections, they would be trapped in commercial disputes. Therefore, it is of great significance to authenticate the IP ownership of the deep learning models which are the neural networks obtained by deep learning training process.

In recent years, researchers have proposed some protection schemes of deep learning models inspired by digital watermarking [1, 7, 12]. The former watermarking protection schemes for deep learning models can be divided into two categories: white-box watermarking and black-box watermarking. The white-box watermarking assumes the model parameters and other internal details can be accessed for public. The black-box watermarking mechanism assumes

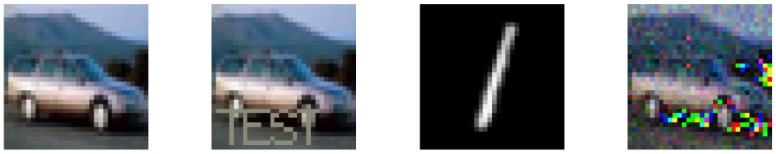
that the model can only be accessed through the API interface deployed by the model creator.

A. RELATED WORKS

On the white-box watermarking, some schemes have been proposed. Uchida et al. [19] and Nagai et al. [20] made the first attempt to embed a watermark into a neural network model inspired by digital watermarking. The watermark was embedded in the model parameters by a parameter regularizer without significantly changing the distribution of model parameters. It would not impair the performance of neural network models. Wang et al. [13] found out that the above schemes modified the statistical distribution of the model, and the watermark length could be detected by measuring the standard deviation. As a result, they leveraged an overwriting algorithm similar to the scheme [19] to remove the watermark. Rouhani et al. [21] exploited two loss functions to modify the weights of the model and to produce a specific activation under a particular input. Thus, it works in both white-box and black-box environments. Chen et al. [16] have proposed to watermark intelligent models by adding a new regularization term to the loss function. But, Adi et al. [16] found out that their schemes do not explicitly address fraudulent ownership claim attacks. These methods are white-box watermarking schemes. Although they all have good performance to the protection of intelligent models, the white-box watermarking schemes are vulnerable to statistical attacks [9]. Besides, the extraction of the watermark has the limitation that it can only be successfully extracted when a complete model is obtained locally. Therefore, many scholars began to turn to black-box watermarking protection schemes.

Merrer et al. [2] proposed a new zero-bit watermarking algorithm leveraging adversarial samples. It can solve the difficulty of watermark extraction by utilizing a small set of queries through the remote API. Zhang et al. [6] proposed three different watermark generation mechanisms ($WM_{content}$, $WM_{unrelated}$, WM_{noise}), shown in Fig.1, and a model protection framework. This framework applies recreated images as a watermark to verify whether the outputs are consistent with the labels. The main idea comes from the Trojan Trigger [18], which uses a special set of inputs designed by the owner to trigger the abnormal behavior of Trojan Neural Network. Thus, the users can prove the ownership of the model. Adi et al. [16] proposed embedding watermarks by the error classified images, which can be theoretically called "backdoor". Guo et al. [5] proposed to add the message mark associated with the signature to part of the original images as a watermark. For the watermarking schemes that changed the original images as the key trigger samples, Namba et al. [9] proposed a query modification attack. It can make the verification of the trigger set invalid.

Although the white-box watermarking can carry more information, its application scenarios are limited. It cannot resist statistical attacks [9]. The black-box watermarking is more practical because it works remotely by trigger sets.



(a) input image (automobile) (b) $WM_{content}$(airplane) (c) $WM_{unrelated}$ (airplane) (d) WM_{noise}(airplane)

FIGURE 1. Watermark generation mechanisms [6]

B. MOTIVATIONS

The principle of the black-box watermarking scheme is to embed a set of specific trigger set into a model as a watermark. The trigger set is a non-public dataset containing watermark features. It contains two parts: data content and data annotation. The data content and annotations of the trigger set are often different from their corresponding original content and labels, which makes a fingerprint on a model. Then, proving the fingerprint can verify ownership of the model. The embedding of the watermark is to put the trigger set and normal training data together into the neural network for training so that we can verify the identity of the model during the watermark extraction stage.

The former trigger sets are usually created by constructing additional meaningful features such as noises and badges on the original data, as is shown in Fig.1. The focus of creating a trigger set is data content. These schemes cannot guarantee the separation of the secret key and the trigger set. Therefore, there may be a risk of key leakage when verifying the identity of the model by trigger set. The key leakage exposures the watermarking features of the model, which leads the insecurity of the schemes. In addition, the trigger sets of the former schemes are generalized, which leads the trigger set out of enclosure. A good watermark protection scheme should avoid the generalization of watermarks. Generalization of watermarks means that others can easily create fake trigger sets based on the original watermark features. The adversary may choose similar key samples to trigger the model under the fraudulent ownership claim attacks [21]. Using the fake samples to trigger the watermarked model, the attacker can achieve the purpose of fraudulently claiming the ownership of the model.

Therefore, in this paper, we turn our attention to data annotation. The annotation of the trigger set can effectively compensate for the generalization of watermarks. Most of the previous trigger set labeling work was done manually. However, the automatic annotation of the trigger set in black-box watermarking is necessary to save manpower. In order to

meet the above needs, we propose a chaos based automatic labeling scheme of trigger sets.

Due to the sensitivity of initial value, aperiodic behavior and unpredictability of the chaotic sequence, chaos in long-term cannot be exactly predicted by statistics-based machine learning. Once the trigger data set is chaotic labeled, attackers cannot reproduce an alternative trigger set including the labels that match the trigger set. Therefore, the watermarking features meet the non-generalization capacity. Non-generalization of watermarks guarantees adversaries cannot create fake trigger sets even though they have got our original trigger set. In addition, the initial value X_0 and parameter μ of the Logistic map are designed as the keys in our scheme. We use these keys to generate the chaotic labeling trigger set, then extract the watermark and verify the ownership of the model by trigger set. Therefore, the secret keys cannot be leaked during the watermarking verification. The contributions of our work include: (1) It proposes a new watermarking scheme based on chaotic automatic annotation of trigger set; (2) It ensures the non-generalization of watermarking features and good function of the original model at the same time; (3) It can avoid the leakage of keys when verifying watermarks; (4) It can resist fine-tuning attacks, compression attacks, fraudulent ownership claim attacks and overwriting attacks.

II. CHAOTIC FEATURES AND ANNOTATION

A. CHAOTIC FEATURES

Chaos has superior features, such as the sensitivity of initial value, aperiodic behavior, and unpredictability of the chaotic sequence [10]. These features can be applied in cryptography: (1) Non-periodic: it is chaotic and irregular, hence it is suitable for encryption. (2) Sensitivity: the orbit of the system is extremely sensitive to the initial value. Subtle changes in the initial value will lead an entire difference on the chaotic behavior. (3) Unpredictable: it is unpredictable for the chaotic sequence in the long term. Thus, machine learning cannot predict its behavior statistically. In view of these features, chaos is very suitable for the automatic annotation of trigger set.

B. PRINCIPLE OF CHAOTIC ANNOTATION

The principle of chaotic annotation is to consider the chaotic sequence that generated by the chaotic system as the key sequence [4]. The chaotic system generates complex dynamic behavior [3, 14, 15]. Because chaos is extremely sensitive to the initial value, numerous uncorrelated and pseudo-random chaotic sequences can be generated. So, chaotic automatic annotation guarantees the possibility of commercialization of the model. In fact, even if the attacker has obtained the equations of generating the chaotic sequence, it is difficult to guess the corresponding coefficient parameters and the initial value of the chaotic sequence. Therefore, the initial value and parameters are designed as keys. The chaotic sequence is unique once the initial value and parameters are assigned. Without knowing the related initial value and parameters, the obtained chaotic sequence cannot reconstruct the unknown parameters and predict the forthcoming sequence in the long run. Thus, chaotic automatic annotation guarantees the non-generalization of watermarks. Without loss of generality, we mainly address the characteristics of one-dimensional Logistic map in this paper. In the following experiments, we will use Logistic chaotic map labeling our trigger set as well.

One dimensional Logistic map is a very simple chaotic map in mathematical form. However, the system has extremely complex dynamic behavior and is widely used in the field of secure communication. The mathematical expression is as follows:

$$X_{n+1} = \mu * X_n * (1 - X_n), \mu \in [0,4], X \in [0,1], \quad (1)$$

It is proposed by mathematical ecologist May in 1976 in an influential review published in the journal Nature [8]. The parameter μ is Logistic parameter. Researches have shown that when $X \in [0,1]$, the Logistic map is in a chaotic state. Fig.2 shows the bifurcation diagram of Logistic map. It indicates that the parameter μ which is close to 4 can lead a better chaotic behavior of X. Therefore, when labeling the trigger set, the parameter μ should be close to 4. In addition, the experimental results show that the chaotic sequence is extremely sensitive to the initial value X_0. Fig.3 shows the distribution of the 32 results after 2000 iterations of Logistic map is different after changing the initial value X_0 from 0.2001 to 0.2002. As can be observed, a subtle change in the initial value leads a significant difference between the two distributions.

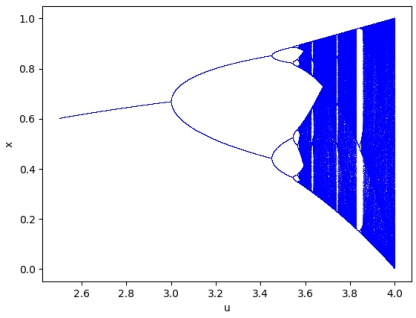
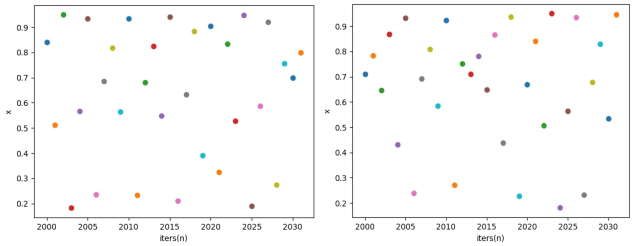


FIGURE 2. Bifurcation diagram of Logistic map



(a) (b)

FIGURE 3. Data distribution under different initial values (a) The distribution of 32 results after 2000 iterations of Logistic map with $X_0 = 0.2001$, $\mu = 3.8$ (b) The distribution of 32 results after 2000 iterations of Logistic map with $X_0 = 0.2002$, $\mu = 3.8$

The trigger set is different from the original training dataset in both the data content and data annotation. The annotation of trigger set needs unique intentional labeling to be the fingerprint of a deep learning model. After such training with this intentional trigger set, the model can remember the unique fingerprint.

Since a subtle change in the initial values leads an entire difference in chaotic sequence, the chaotic sequence maps the trigger data sequence and obtains the data-related labels. Then, the labels and trigger data samples may cause the deep learning model over-fitting on this trigger set. An ideal watermarking scheme for deep learning models not only needs to maintain the function of the original model, but also needs good over-fitting on the trigger set. Over-fitting is usually considered as a negative result. However, in the watermarking, the over-fitting on trigger set can enclose the trigger set sample, thus avoiding generalization. The generalization of the watermarks is the key of fraudulent ownership claim attacks. Therefore, the trigger set based on chaotic labeling can confine the trigger set and offer large key space to watermarking models.

III. THE PROPOSED ANNOTATION SCHEME AND PERFORMANCE

A. THE PROPOSED ANNOTATION SCHEME

The proposed scheme labels the trigger set by Logistic chaotic map. Due to the uniqueness of chaotic sequence, the unique watermark of the model can be proved in the watermark extraction stage.

Assume that there is an m classification problem, the labels of the data are represented by $l_i, 1 \le i \le m$ and the collection of the m labels is $\bigcup_{i=1}^{m} l_i$. Firstly, we need to select n trigger samples $N_k, 1 \le k \le n$. The corresponding original labels of them are $L_k, 1 \le k \le n$, where $\bigcup_{k=1}^{n} L_k \subseteq \bigcup_{i=1}^{m} l_i$. It means that the union of all trigger set labels is a subset of the collection of the m labels. Our purpose is to divide these trigger samples into an entire different chaotic labels $L'_k, 1 \le k \le n$, where $L'_k \ne L_k, 1 \le k \le n$ and $\bigcup_{k=1}^{n} L'_k \subseteq \bigcup_{i=1}^{m} l_i$.

Next, we divide the chaotic value range $[0,1]$ of one dimensional Logistic map into m intervals $[y_{i-1}, y_i)$, where $1 \le i \le m$. Each interval $[y_{i-1}, y_i)$ corresponds to a specific category l_i. After that, we abandon the results of Logistic map for the beginning N' iteration times. Then, we obtain a sequence within n results after n iterations and the n chaotic values are assigned to the n trigger samples $N_k, 1 \le k \le n$. At this moment, each trigger sample corresponds to a Logistic mapping value. Therefore, we can classify the trigger samples into specific categories $l_i, 1 \le i \le m$ according to the match between the chaotic values and corresponding intervals. Suppose X_{Nk} represents the chaotic value corresponding to the trigger sample $N_k, 1 \le k \le n$, the annotation algorithms are as follows.

Algorithm1 Trigger Annotation

Input:
Original Trigger data: $D_{wm} = \{N_k, L_k\}_{k=1}^{n}$
Chaos Value of N_k: $X_{Nk}, 1 \le k \le n$
Label: $l_i, 1 \le i \le m$

Output:
Trigger Set: $D'_{wm} = \{N_k, L'_k\}_{k=1}^{n}$

function Trigger_Annotation1()
  for k **from** 1 **to** n
    if $X_{Nk} \in [0, y_1)$ **then** $L'_k = l_1$
    else if $X_{Nk} \in [y_1, y_2)$ **then** $L'_k = l_2$
    ...
    else if $X_{Nk} \in [y_{i-1}, y_i)$ **then** $L'_k = l_i$
    ...
    else if $X_{Nk} \in [y_{m-2}, y_{m-1})$ **then** $L'_k = l_{m-1}$
    else $X_{Nk} \in [y_{m-1}, 1]$ **then** $L'_k = l_m$
  end for
end function
function Trigger_Annotation2()
  for k **from** 1 **to** n
    if $L'_k = L_k = l_i$ **then** $L'_k = l_{i+1}$
  end for
  if $k = n$ **and** $L'_k = L_k = l_i$ **then** $L'_k = l_1$
end function

According to the function *Trigger_Annotation1()*, assume that $X_{Nk} \in [y_{i-1}, y_i)$, the label of sample N_k will be $L'_k = l_i$, where $1 \le k \le n$ and $1 \le i \le m$. To ensure that each trigger is classified into an unpredictable category, the condition $L'_k \ne L_k, 1 \le k \le n$ is needed. Therefore, we relabel the trigger set by the function *Trigger_Annotation2()*.

Assuming that $1 \le k \le n\text{-}1$ and $L'_k = L_k = l_i$, we label the data N_k by $L'_k = l_{i+1}$, where $1 \le i \le m$. While $k = n$ and $L'_k = L_k = l_i$, we label the sample N_k by $L'_k = l_1$. At this point, all the labeling work of trigger set is completed. Then, by merging the training set and trigger set together to train a model, the watermark can be embedded in the intelligent model.

B. PERFORMANCE AND EXPERIMENTAL RESULTS

We carried out the experiments on two different models to verify our scheme: the vehicle steering control model of intelligent self-driving and the handwritten digit recognition model. The corresponding two datasets are the vehicle steering control dataset of intelligent self-driving and the

MNIST dataset. The vehicle steering control dataset of intelligent self-driving is a non-public dataset collected by ourselves. It contains a total of 10147 training images, and each image contains 160×120 pixels. There are four categories in total: stop, forward, rightward and leftward. MNIST [17] is a large handwritten digit recognition dataset containing 60,000 training images and 10,000 testing images. Each image has 28×28 pixels, and each pixel value is in the grayscale range between 0 and 255. There are totally 10 categories, from 0 to 9.

We verified the influence of the proportion of trigger set and training dataset on our scheme as well. Without loss of generality, we randomly use 16, 32, 64 trigger samples as our watermarks in the self-driving models and 20, 50, 100 trigger samples as our watermarks in the handwritten digit recognition models. These trigger samples are different from each other and there is no repetition between the trigger sets.

The watermark embedding and extraction process is shown in Fig.4. Through steps ❶ to ❸, we can embed the watermark into the model. Through steps ❹ to ❻, we can extract the watermark embedded in the model. Firstly, put the prepared *Trigger Samples* with salt-and-pepper noise into the *Chaotic Labeler* to relabel the samples, the annotation method can be gain from ***Algorithm1***. The image samples relabeled are our *Trigger Set*. Then, by putting the *Trigger Set* and the *Training Set* together into the *Data Set* for training, a model with a watermark will be generated. In the watermark extraction process, we only need to put the trigger set into the model to obtain the outputs of the model. Then, by comparing the *Output labels* of the model with the *Chaotic labels* of the trigger set, we can verify ownership of the model. The experimental results in Tab.V and Tab.VI indicate watermarks of various lengths can achieve good model accuracy and a 100% watermark extraction rate. We will analyze the results of the experiments in detail in the next section.

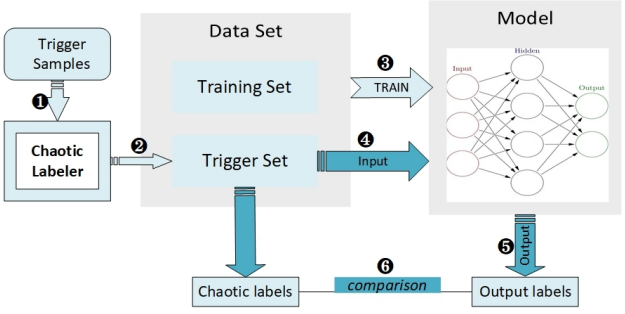


FIGURE 4. **Flow chart of watermark embedding and extraction**

IV. PERFORMANCE ANALYSIS

A FIDELITY ANALYSIS

The fidelity requires that the accuracy of the model will not be affected after embedding the watermark, so we evaluate the accuracy of the models before and after embedding the watermark.

In Tab.I and Tab.II, before embedding the watermark, the accuracy of the steering control model of intelligent self-driving is 99.51% and the accuracy of the handwritten digit recognition model is 99.36%. After embedding the watermark, in the worst case, the accuracy of the self-driving model with 64 trigger samples can reach 99.43% and the accuracy of the MNIST model with 100 trigger samples can reach 99.30%. Therefore, the accuracy of the models only drops by 0.08% and 0.06% in the worst case. Compared with the original model, the accuracy of the self-driving model with 16 trigger samples and the MNIST model with 20 trigger samples has improved to a certain extent. The results clearly indicate that the embedding of the watermark has no significant impact on the performance of the models. Therefore, there is no doubt that our automatic annotation scheme for trigger set meets the requirements of fidelity.

As we obtained in Tab.I and Tab.II, the length of watermark has little effect on the fidelity of our model. The accuracy of self-driving model with 16 trigger samples and the handwritten digit recognition model with 20 trigger samples is even slightly higher than the original models.

TABLE I
FIDELITY ANALYSIS ON SELF-DRIVING MODEL

Model	Original Model	16-Trigger Model	32-Trigger Model	64-Trigger Model
Accuracy	0.9951	0.9971	0.9951	0.9943

TABLE II
FIDELITY ANALYSIS ON MNIST MODEL

Model	Original Model	20-Trigger Model	50-Trigger Model	100-Trigger Model
Accuracy	0.9936	0.9937	0.9933	0.9930

B EFFECTIVENESS AND INTEGRITY ANALYSIS

Effectiveness refers to whether the ownership of the model can be successfully verified under the protection of our chaotic automatic annotation scheme. It requires the watermarked model to be able to identify the corresponding trigger set embedded in the model with high accuracy. Integrity requires our solution does not falsely claim ownership of unwatermarked models.

We extract the watermarks of the four models(one original model and three watermarked models with watermarks of different lengths) by trigger sets of three different length in the experiments. As we mentioned in the previous section, there are no duplicate samples between trigger sets of different lengths. The experimental results are shown in Tab.III and Tab.IV. All trigger sets can achieve a watermark extraction rate of 100% on the model embedded with the corresponding trigger set, which meets the effectiveness requirements. All trigger sets have achieved a low watermark extraction rate of 0%-15% on models that do not embed the trigger set as a watermark, which meets the integrity requirements.

As can be seen in Tab.III and Tab.IV, the length of the watermark does not affect the effectiveness and integrity of our scheme.

TABLE III
EFFECTIVENESS AND INTEGRITY ON SELF-DRIVING MODEL

Watermark Extraction rate	Original Model	16-Trigger Model	32-Trigger Model	64-Trigger Model
16-Trigger	0.00	1.00	0.00	0.00
32-Trigger	0.00	0.00	1.00	0.00
64-Trigger	0.00	0.00	0.00	1.00

TABLE IV
EFFECTIVENESS AND INTEGRITY ON MNIST MODEL

Watermark Extraction rate	Original Model	20-Trigger Model	50-Trigger Model	100-Trigger Model
20-Trigger	0.00	1.00	0.15	0.10
50-Trigger	0.00	0.04	1.00	0.04
100-Trigger	0.01	0.07	0.05	1.00

C ROBUSTNESS ANALYSIS

1) ROBUSTNESS AGAINST FINE-TUNING

Fine-tuning and transfer learning [11] seem to be the normal method to improve the model avoiding the heavy work while training a model from scratch. An ideal scheme can still remain the watermark after fine-tuning. We developed the fine-tuning attacks on our original models by substituting the cross-entropy loss for mean-squared-error loss to retrain the models again. After the fine-tuning attacks, Tab.V and Tab.VI indicate the accuracy of the six watermarked models has increased with the growth rate of 0.01% - 0.14%. The watermark extraction rate of all the models is still 100% after fine-tuning. Therefore, our scheme would not affect the performance and the watermark extraction rate of the model after fine-tuning. Besides, the length of the watermark will not affect the robustness of our scheme against fine-tuning attacks as well.

TABLE V
ROBUSTNESS AGAINST FINE-TUNING ATTACKS OF SELF-DRIVING MODEL

Model		16-Trigger Model	32-Trigger Model	64-Trigger Model
Before fine-tuning	Accuracy	0.9971	0.9951	0.9943
	Extraction rate	1.0	1.0	1.0
After fine-tuning	Accuracy	0.9985	0.9961	0.9952
	Extraction rate	1.0	1.0	1.0

TABLE VI
ROBUSTNESS AGAINST FINE-TUNING ATTACKS OF MNIST MODEL

Model		20-Trigger Model	50-Trigger Model	100-Trigger Model
Before fine-tuning	Accuracy	0.9937	0.9933	0.9930
	Extraction rate	1.0	1.0	1.0
After fine-tuning	Accuracy	0.9938	0.9934	0.9937
	Extraction rate	1.0	1.0	1.0

2) ROBUSTNESS AGAINST COMPRESSION ATTACKS

Compression attack is also a common attack. We use the TensorFlow Model Optimization Toolkit to prune our model so that we can compress it. Pruning means eliminating unnecessary values in the weight tensor. The pruning step in the TensorFlow Model Optimization Toolkit is amplitude-based weight pruning. It gradually zeros the model weights from the weight which is closest to zero to achieve model sparsity.

In Tab.VII and Tab.VIII, the accuracy and the watermark extraction rate of the six watermarked models are both very high when the pruning rate is smaller than 0.60, which means that our model can resist the pruning attack on these occasions. The MNIST model can even resist 70% pruning attacks. Even though the pruning attacks makes the watermark extraction failure when the pruning rate is bigger than 0.80, the attacked model suffers from a significant loss of accuracy. This means when the extraction of the watermark fails, the watermarked model has already lost its original function and becomes unavailable. Thus, our scheme can resist the compression attack.

Considering the impact of watermark length on compression attacks, the results in Tab.VII and Tab.VIII show that the number of samples in the trigger set has little effect on the robustness of our scheme against the compression attacks.

TABLE VII
ROBUSTNESS AGAINST COMPRESSION ATTACKS OF SELF-DRIVING MODEL

Model	Pruning Rate	Model Accuracy	Watermark Extraction Rate	Size (Mb)
16-Trigger Self-driving Model	-	0.9971	1.0	13.77
	0.10	0.9966	1.0	11.90
	0.20	0.9966	1.0	10.94
	0.30	0.9966	1.0	9.91
	0.40	0.9956	0.9375	8.87
	0.50	0.9951	0.9375	7.79
	0.60	0.9872	0.0625	6.74
	0.70	0.9848	0.0000	5.54
	0.80	0.9849	0.0000	4.22
	0.90	0.9818	0.0625	2.82
32-Trigger Self-driving Model	-	0.9951	1.0	13.77
	0.10	0.9995	1.0	11.87
	0.20	0.9995	1.0	10.92
	0.30	0.9995	1.0	9.86
	0.40	0.9995	1.0	8.80
	0.50	0.9995	1.0	7.70
	0.60	0.9951	0.9063	6.58
	0.70	0.9531	0.0000	5.48
	0.80	0.9648	0.0000	4.15

	0.90	0.9667	0.0625	2.71
	-	0.9943	1.0	13.77
	0.10	0.9952	1.0	11.92
	0.20	0.9952	1.0	10.98
64-Trigger	0.30	0.9952	1.0	9.95
Self-	0.40	0.9947	1.0	8.95
driving	0.50	0.9942	1.0	8.47
Model	0.60	0.9919	0.9063	7.50
	0.70	0.9694	0.0000	6.46
	0.80	0.8758	0.0000	4.24
	0.90	0.79	0.0625	2.85

TABLE VIII
ROBUSTNESS AGAINST COMPRESSION ATTACKS OF MNIST MODEL

Model	Pruning Rate	Model Accuracy	Watermark Extraction Rate	Size (Mb)
	-	0.9937	1.0	12.52
	0.10	0.9931	1.0	10.93
	0.20	0.9934	1.0	10.08
	0.30	0.9932	1.0	9.14
20-Trigger	0.40	0.9932	1.0	8.16
MNIST	0.50	0.9934	1.0	7.13
Model	0.60	0.9934	0.9000	6.05
	0.70	0.9932	0.7500	4.89
	0.80	0.9819	0.4500	3.72
	0.90	0.9780	0.0000	2.45
	-	0.9933	1.0	12.52
	0.10	0.9934	1.0	10.93
	0.20	0.9928	1.0	10.09
	0.30	0.9943	1.0	9.14
50-Trigger	0.40	0.9930	1.0	8.16
MNIST	0.50	0.9940	1.0	7.13
Model	0.60	0.9933	1.0	6.04
	0.70	0.9939	1.0	4.88
	0.80	0.9904	0.6000	3.72
	0.90	0.9461	0.1000	2.45
	-	0.9930	1.0	12.52
	0.10	0.9933	1.0	10.90
	0.20	0.9931	1.0	10.01
100-	0.30	0.9937	1.0	9.11
Trigger	0.40	0.9934	1.0	8.13
MNIST	0.50	0.9931	1.0	7.10
Model	0.60	0.9935	1.0	6.04
	0.70	0.9931	0.7800	4.90
	0.80	0.9926	0.5100	3.72
	0.90	0.9884	0.0500	2.45

3) ROBUSTNESS AGAINST OVERWRITING ATTACKS

Overwriting attack assumes that the attackers know our watermarking mechanism. Then, the adversary attacks our model according to our watermark generation algorithm to achieve the purpose of covering or eliminating our original watermark. Overwriting attacks are similar to fraudulent ownership claim attacks. Nevertheless, different from fraudulent ownership claim attacks, the intent of overwriting attacks is not simply to claim ownership of a model, but to erase the original watermark and add a new watermark. During the overwriting attack, the attacker cannot know our trigger set. However, we assume the worst case: the attacker obtains our trigger set and *Chaotic Labeler*. According to Kerchhoff principle, the adversaries do not know the secret keys, which are the parameter μ and the initial value X_0 of the *Chaotic Labeler* that generates the original trigger set labels.

By changing the parameter μ and the initial value X_0 of the Logistic mapping function, we obtain the new overwriting trigger sets corresponding to the six sets of trigger samples. The new overwriting trigger set is used to fine-tune the corresponding watermarked model to carry out the overwriting attack. For each watermarked model, we conducted four sets of overwriting attack simulations. We can obtain the experimental results in Tabs.IX-XIV. The third and fourth columns in the table represent the watermark extraction rate of the original trigger set and the new overwriting trigger set, respectively. On the self-driving models, the extraction rate of the new watermarks can reach 86.00% in Tab.XI in the best case. On the MNIST models, the extraction rate of new trigger set can reach 100%. However, when the extraction rate of the new watermark is high, most of the watermarked models have a high loss in accuracy, which have lost the model functionality. For instance, in the first overwriting attack simulation in Tab.IX, the accuracy of the model is reduced to 48.93% when the extraction rate of the new watermark reaches 75.00%. On the contrary, under the premise of ensuring the accuracy of the model, the extraction rate of the new overwriting watermark is very low. In the fourth overwriting attack simulation in Tab.XI, the extraction rate of the new watermark can only reach 39.00% when the accuracy of the model reaches 99.89%. Therefore, our chaotic scheme is robust to overwriting attacks and fraudulent ownership claim attacks.

Finally, let us consider the impact of the length of the watermark on the overwriting attacks. According to Tabs.IX-XI and Tabs.XII-XIV, we can get that the length of the watermark will not have a significant impact on the robustness against overwriting attacks of our scheme.

TABLE IX
ROBUSTNESS AGAINST OVERWRITING ATTACKS OF 16-TRIGGER SELF-DRIVING MODEL

Model	Model Accuracy	Extraction Rate (16-Original Trigger)	Extraction Rate (16-Overwriting Trigger)
Overwriting-16-001	0.4839	0.5000	0.7500
Overwriting-16-002	0.2980	0.5625	0.7500
Overwriting-16-003	0.8700	0.6875	0.3750
Overwriting-16-004	0.2400	0.1250	0.1875

TABLE X
ROBUSTNESS AGAINST OVERWRITING ATTACKS OF 32-TRIGGER SELF-DRIVING MODEL

Model	Model Accuracy	Extraction Rate (32-Original Trigger)	Extraction Rate (32-Overwriting Trigger)
Overwriting-32-001	0.6019	0.3750	0.8438
Overwriting-32-002	0.5958	0.3750	0.8438
Overwriting-32-003	0.8601	0.4370	0.8125
Overwriting-32-004	0.7126	0.5310	0.7188

TABLE XI
ROBUSTNESS AGAINST OVERWRITING ATTACKS OF 64-TRIGGER SELF-DRIVING MODEL

Model	Model Accuracy	Extraction Rate (64-Original Trigger)	Extraction Rate (64-Overwriting Trigger)
Overwriting-64-001	0.9121	0.4843	0.8600
Overwriting-64-002	0.9100	0.4843	0.8590
Overwriting-64-003	0.9952	0.8906	0.5000
Overwriting-64-004	0.9989	1.000	0.3900

TABLE XII
ROBUSTNESS AGAINST OVERWRITING ATTACKS OF 20-TRIGGER MNIST MODEL

Model	Model Accuracy	Extraction Rate (20-Original Trigger)	Extraction Rate (20-Overwriting Trigger)
Overwriting-20-001	0.7490	0.1000	1.0000
Overwriting-20-002	0.4770	0.1000	1.0000
Overwriting-20-003	0.5490	0.0500	0.9500
Overwriting-20-004	0.5280	0.2500	0.9000

TABLE XIII
ROBUSTNESS AGAINST OVERWRITING ATTACKS OF 50-TRIGGER MNIST MODEL

Model	Model Accuracy	Extraction Rate (50-Original Trigger)	Extraction Rate (50-Overwriting Trigger)
Overwriting-50-001	0.2700	0.1200	1.0000
Overwriting-50-002	0.4500	0.1200	1.0000
Overwriting-50-003	0.6640	0.3200	0.7400
Overwriting-50-004	0.5090	0.2200	0.5800

TABLE XIV
ROBUSTNESS AGAINST OVERWRITING ATTACKS OF 100-TRIGGER MNIST MODEL

Model	Model Accuracy	Extraction Rate (100-Original Trigger)	Extraction Rate (100-Overwriting Trigger)
Overwriting-100-001	0.9328	0.1400	1.0000
Overwriting-100-002	0.8780	0.1400	1.0000
Overwriting-100-003	0.6290	0.2000	0.9900
Overwriting-100-004	0.7535	0.2000	0.9800

D SECURITY AND PRACTICALITY ANALYSIS

The security requires that the key of our scheme is unknown to the public and is separated from the trigger set. The practicality means our solution has a large amount of key space, thus promoting the commercialization process of intelligent models.

Security In the various solutions previously proposed, the creation of the trigger set is to add a certain understandable badges or noise to the trigger samples, so the key of the model protection scheme is inseparable from the trigger set. When verifying model ownership, the key will inevitably be leaked. The leak of the key will cause some attackers to forge trigger samples based on the leaked trigger set characteristics, so fraudulent ownership claim attacks may occur. This will lead to the model being claimed by others, which is very detrimental to the original author. Our scheme uses Logistic map to chaotically label the trigger set, so the initial value X_0 and the parameter μ of Logistic map are considered as keys. The key of our scheme is separated from the trigger set. When verifying the identity of the model, the leakage of the trigger set will not lead to the leakage of secret keys. Therefore, chaotic labeling of trigger set has great security advantages compared with other black-box watermarking protection schemes.

Practicality First of all, the chaotic automatic labeling scheme can save a lot of time and manpower. Secondly, the initial value X_0 and the parameter μ of chaos will provide a large amount of key space for our scheme. Finally, providing each user with a unique key can help quickly finding the source of leaky model when the model is leaked. The above points show that the chaotic automatic annotation scheme can promote the commercialization of intelligent models and has very good practicality.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we propose a new black-box watermarking scheme based on chaotic automatic labeling of trigger set. It effectively makes up for the shortcomings of existing watermarking technology and solves the automatic labeling

problem of the black-box watermarking trigger set. Firstly, the application of chaos in labeling the trigger set ensures the non-generalization of the watermark. Due to the superior characteristics of chaos, machine learning cannot predict the behavior of chaos in the long run. Attackers cannot find other trigger sets that match the characteristics of our trigger set, thus resisting overwriting attacks and fraudulent ownership claim attacks. Besides, this method not only effectively saves time and manual works, but also helps promoting the commercialization of the model. Assigning each user with a unique watermark by our scheme is also effective to find the source of the leaky model quickly in case of model leakage. Another significance of our scheme is that the secret keys are separate from the trigger set. When extracting the watermark, the leakage of the trigger set will not lead to the leakage of secret keys. We conducted experiments on two datasets and watermarks of six different lengths by our chaotic automatic annotation scheme. We also evaluate our model through fidelity, effectiveness, integrity, robustness, security and practicality. The results show that the method has great advantages in model protection.

In the experimental part, we verify our scheme through experiments with six different length watermarks. At present, our watermark capacity can reach 200 trigger samples when the training data reaches 60,000 samples. In future works, the exact watermark capacity of our scheme still needs further research.

REFERENCES

[1] C. Y. Chang, S. J. Su, "A neural-network-based robust watermarking scheme," In ICSMC, vol. 3, pp. 2482-2487, 2005.
[2] E. L. Merrer, P. Perez, G. Tredan, "Adversarial Frontier Stitching for Remote Neural Network Watermarking," Neural Computing & Applications, 2017.
[3] F. Yang, J. Mou, J. Liu, C. Ma, H. Yan, "Characteristic analysis of the fractional-order hyperchaotic complex system and its image encryption application," Signal processing, vol. 169, 2020.
[4] H. Zhang, S. Zhou, "Application of Chaos Theory in Cryptography," Journal of Chongqing University (Natural Science Edition), pp. 39-43, 2004.
[5] J. Guo, M. Potkonjak, "Watermarking deep neural networks for embedded systems," In Proceedings of the 2018 International Conference On Computer Aided Design, pp. 1-8, 2018.
[6] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pp. 159-172, 2018.
[7] L. Luo, Z. Chen, M. Chen, X. Zeng, Z. Xiong, "Reversible Image Watermarking Using Interpolation Technique," In IEEE Transactions on Information Forensics and Security, vol. 5, no. 1, pp. 187-193, 2010.
[8] R. May, "Simple mathematical models with very complicated dynamics," Nature, vol. 261, pp. 459–467, 1976.
[9] R. Namba, J. Sakuma, "Robust Watermarking of Neural Network with Exponential Weighting," In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, 2019.
[10] S. H. Strogatz, "Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering," Perseus Books Publishing, LLC, Westview, 1994.
[11] S. Liu, H. Yao, G. Wen, "Neural network based steganalysis in still images," In Proceedings of ICME, 2003.
[12] S. Rizzo, F. Bertini, D. Montesi, "Fine-grain watermarking for intellectual property protection," EURASIP Journal on Information Security, 2019.
[13] T. Wang, F. Kerschbaum, "Attacks on Digital Watermarks for Deep Neural Networks," In IEEE International Conference on Acoustics Speech and Signal Processing, pp. 2622-2626, 2019.
[14] X. Ma, J. Mou, J. Liu, C. Ma, F. Yang, X. Zhao, "A novel simple chaotic circuit based on memristor-memcapacitor," Nonlinear Dynamics, vol. 100, no. 3, pp. 2859-2876, 2020.
[15] X. Ye, J. Mou, C. Luo, Z. Wang, "Dynamics analysis of Wien-bridge hyperchaotic memristive circuit system," Nonlinear Dynamics, vol. 92, no. 3, pp. 923-933, 2018.
[16] Y. Adi, C. Baum, M. Cisse, B. Pinkas, J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," In 27th Usenix Security Symposium, pp. 1615–1631, 2018.
[17] Y. LeCun, C. Cortes, C. J. Burges, "The mnist database of handwritten digits," 1998.
[18] Y. Liu, S. Ma, Y. Aafer, W. C. Lee, J. Zhai, W. Wang, X. Zhang, "Trojaning attack on neural networks," In NDSS, 2017.
[19] Y. Uchida, Y. Nagai, S. Sakazawa, S. Satoh, "Embedding Watermarks into Deep Neural Networks[J]," In Proceedings of the ACM on International Conference on Multimedia Retrieval, 2017.
[20] Y. Nagai, Y. Uchida, S. Sakazawa, S. Satoh, "Digital watermarking for deep neural networks," International Journal of Multimedia Information Retrieval, vol. 7, no.1, pp. 3-16, 2018.
[21] Z. Li, C. Hu, Y. Zhang, S. Guo, "How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN," In Proceedings of the 35th Annual Computer Security Applications Conference, 2019.

Ying-Qian Zhang is a professor in Xiamen University Tan Kah Kee college. He received the Ph.D degree in the Dalian University of Technology in 2015. He is interested in nonlinear systems, chaos theory and information security.

Yi-Ran Jia received the B.S. degree in IoT engineering from Xuzhou Medical University in 2018. She is currently a third-year graduate student majoring in computer technology at Xiamen University. Her research interests focus on digital watermarking, chaotic encryption, artificial intelligence and information security.

Xing-Yuan Wang was born in Liaoning, China, in 1964. He received his B.S. degree in application physics and his M.S. degree in optics from Tianjin University, Tianjin, China, in 1987 and 1992, respectively, and his Ph.D. degree in computer software and theory from Northeastern University, Shenyang, China, in 1999.
From 1999 to 2001, he was a Postdoctoral Fellow with the Department of Automation, Northeastern University. He is currently a Professor with the

Faculty of Electronic Information & Electrical Engineering, Dalian University of Technology, Dalian, China. His research interests include biomedical information, computer graphics, image processing, Complex network and chaos control and synchronization.

Qiong Niu received the B.S. degree in communication engineering from Xiamen University Tan Kah Kee College in 2019. Her research interest focuses on machine learning, big data algorithm model and application.

Nian-Dong Chen received the Ph.D. degree from the Fujian agriculture and Forestry University, in 2008. He is currently a Professor with the New Huadu Business School of Minjiang University. He is interested in big data, artificial intelligence, intelligent security, and Financial security.