Detection of Cloned Recognizers: A Defending Method against Recognizer Cloning Attack

Yuto Mori*, Kazuaki Nakamura*, Naoko Nitta*, and Noboru Babaguchi*
* Graduate School of Engineering, Osaka University, Suita, Osaka, Japan
E-mail: mori@nanase.comm.eng.osaka-u.ac.jp Tel: +81-6-6879-7746

***Abstract*—With the development of machine learning technologies and the spread of mobile terminals, cloud-based image recognition services are getting popular in recent years. However, these services might suffer from a new type of attack called "recognizer cloning attack" (RCA), in which an attacker sends a lot of images to a recognition server and receives their recognition results to train a new recognizer that mimics the function of the server's original recognizer. We refer to the recognizers trained by RCA as "cloned recognizers" (CR). CRs allow attackers to analyze the weakness of their original recognizer and cause serious damage to the providers of the original service. To defend against RCA, we propose a method for detecting CRs in this paper. Our proposed method receives two recognizers as input and discriminates whether one of them is a CR of the other or not. We experimentally analyzed the properties of CRs and got the following two findings. First, CR and its original recognizer have the almost same recognition boundary. Second, CR provides a recognition confidence score that is almost same or quite higher than that provided by the original recognizer. Using these properties as clues, the proposed method was able to detect CRs with an accuracy of more than 80% in our experiments.**

I. Introduction

With the development of machine learning technologies and the spread of mobile terminals such as smartphones, cloud-based image recognition services are getting popular in recent years, whose typical examples include Google Cloud Vision [1] and Amazon Rekognition [2]. In these services, users first send an image to a recognition server from their own smartphones, and then the server recognizes the sent image and returns its recognition result to the users. This will be a mainstream form of image recognition services in the near future because it is difficult to install a high-end GPU, which is necessary for running modern image recognition processes, into smartphones.

On the other hand, some researchers warn about the possibility that cloud-based image recognition services are attacked by malicious users or attackers [3], [4]. In this paper, we particularly focus on the following type of attack: An attacker sends a lot of images to the recognition server and receives their recognition results, which are used as a training dataset to train a new recognizer that mimics the function of the server's original recognizer. We refer to this type of attack as *recognizer cloning attack* (RCA) and the trained new recognizer as a *cloned recognizer* (CR).

We believe that it is important to prepare for the future risk of RCA and develop its defending method, which is the goal of this work. To clearly specify our research interest, we start to describe a general process of RCA and the problem of trained CRs in detail in the subsequent subsections.

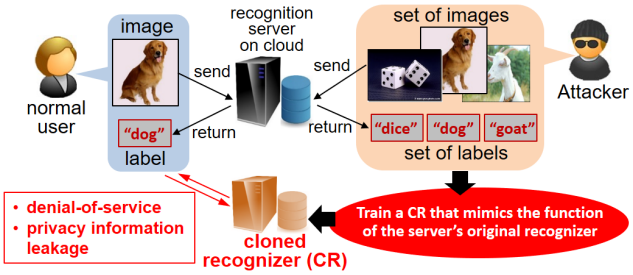

Fig. 1. General process of *recognzer cloning attack* and its problem.

A. Process of Recognizer Cloning Attack

For convenience of formulation, we only focus on single-label recognizers here. Let X be the whole set of image data and Y be a finite set of class labels. For an image $x \in X$ sent from a user, the server computes its class label y as $y = f^L(x) \in Y$, where $f^L : X \to Y$ is the server's image recognizer. In most image recognition services, the confidence score of the recognition result is also computed and returned to the user together with y. Let $z = f^S(x) \in [0,1]$ be the confidence score, where $f^S : X \to [0,1]$ is a map for measuring the confidence score from an input image x.

Sending a set of unlabeled images $\{x_i \in X \mid i = 1, 2, \cdots\}$ to the server, an attacker can obtain their recognition results $\{y_i \in Y \mid i = 1, 2, \cdots\}$ and the corresponding confidence scores $\{z_i \mid i = 1, 2, \cdots\}$. The ultimate goal of the attacker is to construct a CR $g = \left(g^L,\ g^S\right)$ that satisfies

$$\forall x \in X \quad f^L(x) = g^L(x) \tag{1}$$

by using $D = \{(x_i,\ y_i,\ z_i) \mid i = 1, 2, \cdots\}$ as a training dataset. Note that

$$\forall x \in X \quad f^S(x) = g^S(x) \tag{2}$$

is an optional condition; some attackers might attach importance to this condition while some other attackers might ignore it. Fig. 1 shows an overview of the above process of RCA.

B. Problems Caused by Cloned Recognizers

It is an urgent issue to develop a method for defending against RCA, because the CRs trained by RCA can cause the following serious problems:

- If an attacker trains a CR and makes it publicly available for free, it would be more preferred than its original recognizer by most users. This decreases the profit of the owner of the original recognizer. Moreover, privacy information of the users using the CR would be leaked to the attacker.
- The server's original recognizer and its CR would have a common weakness. Hence, the weakness of the original recognizer would be easily found by analyzing the property of its CR. For instance, once adversarial examples that can fool a CR are generated, they can also fool its original recognizer.

There are two strategies for defending against RCA: ex-ante defense and ex-post defense. More specifically, the former is to prevent attackers from training CRs, and the latter is to detect already trained CRs. However, the former, namely ex-ante defense, is difficult to realize. Hence, in this paper, we focus on the latter, namely ex-post defense, and propose its specific method. The contribution of this paper is summarized as follows: First, this is the first work deeply considering the threat of RCA, to the best of our knowledge. Second, we experimentally analyze the characteristics of CRs to realize a method for effectively detecting them.

The remainder of this paper is organized as below. In Section II, we briefly review the attacks on pattern recognition models including RCA. Next, in Section III, we describe our proposed method for detecting CRs in detail. Then, in Section IV, we report the results of the experiments conducted to examine the performance of the proposed method, and finally conclude this paper in Section V.

II. Related Work

There are many kinds of attacks on pattern recognition models constructed by machine learning techniques. Huang et al. [3] roughly divided them into two categories: causative attacks (CA) and exploratory attacks (EA). CAs directly influence their target recognition model by altering its training dataset, whereas EAs do not make any influence on the target model. Instead, an attacker analyzes the target model itself using pairs of its inputs and outputs in EAs.

A typical example of CAs is poisoning attack [5], which is generally carried out against a recognition model constructed by an online learning algorithm. Suppose that a spam filter updated in real time with continually received e-mails. The recognition model of this filter could be drastically changed when it receives new "unnatural" spams whose characteristic is totally different from that of past received ordinary spams. Based on this property, in poisoning attack, an attacker creates "unnatural" spams and sends them to the target filter in order to corrupt its recognition model. In the above process, the attacker's focus is how to efficiently create or select "unnatural" data, which depends on individual recognition model. Kloft et al. investigated a poisoning attack approach particularly focusing on nearest neighbor-based anomaly detection models [6], while Biggio et al. focused on support vector machine (SVM)-based malware detection models [7]. These model-specific methods can hardly work when the target recognition model is a black-box.

A typical example of EAs is model inversion attack (MIA), where an attacker analyzes the target recognition model in order to reconstruct its training examples. MIA could cause a serious privacy issue when it is carried out against biometric recognition models (e.g. face recognition). Fredrikson et al. investigated a method of MIA and experimentally demonstrated that it is possible to reconstruct an individual's face image only from a face recognition model by using its outputting confidential score [8]. Methods of generating adversarial examples [9], [10] are another example of EAs, where an adversarial example means the data leading a neural network model into incorrect outputs. The threat of these attacks is larger if the target recognition model is not a black-box.

Our focused RCA, which is the abbreviation of *recognizer cloning attack*, is a kind of EA, because it does not influence the target recognizer itself. RCA is different from the other attacks in that it can be carried out against black-box models. Moreover, for attackers, a CR constructed by RCA is totally a white-box as well as shares the same weakness with its original recognizer. Hence, RCA could form a hotbed of the other attacks, that is, it could make the other attacks much easily carried out.

III. Method for Detecting Cloned Recognizers

In this section, we first explain our assumed scenario for detecting a CR in subsection III-A, and then we describe the proposed method in detail with the result of a preliminary experiment in subsection III-B.

A. Assumed Scenario

Since different image recognizers have different characteristics, it is difficult to detect CRs without any auxiliary information. Therefore we assume the scenario similar with near-duplicate detection of documents [11], [12] and that of program source codes [13], [14]; that is, the owner of an image recognition service compares his own recognizer $f = \left(f^L,\ f^S\right)$ and another recognizer $h = \left(h^L,\ h^S\right)$ to check whether h is a CR of f or not (see Fig. 2). Of course he knows f is not a CR of any other recognizers and h is not owned by himself. This means h is a black-box for the owner of f, and therefore only the following information can be used.

- A set of images for the comparison process, $U = \{u_i \in X \mid i = 1, 2, \cdots\}$
- Outputs of f for each u_i, i.e., $U_f^L = \{f^L(u_i) \mid i = 1, 2, \cdots\}$ and $U_f^S = \{f^S(u_i) \mid i = 1, 2, \cdots\}$
- Outputs of h for each u_i, i.e., $U_h^L = \{h^L(u_i) \mid i = 1, 2, \cdots\}$ and $U_h^S = \{h^S(u_i) \mid i = 1, 2, \cdots\}$

We believe that the relationship between U_f^L and U_h^L has a certain characteristic if and only if h is a CR of f, which is also the case with the relationship between U_f^S and U_h^S. Hence, we first experimentally investigate the characteristic.

Note that h could be constructed by the following two strategies when it is a CR, as mentioned in Section I-A:

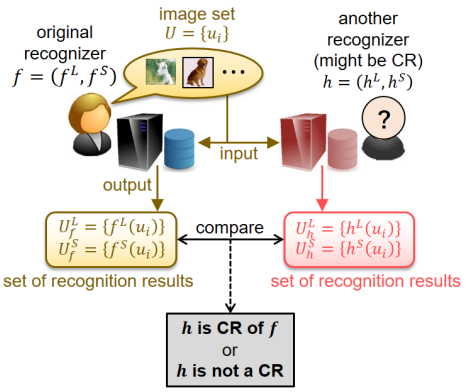

Fig. 2. Assumed scenario of detecting *cloned recognzer*.

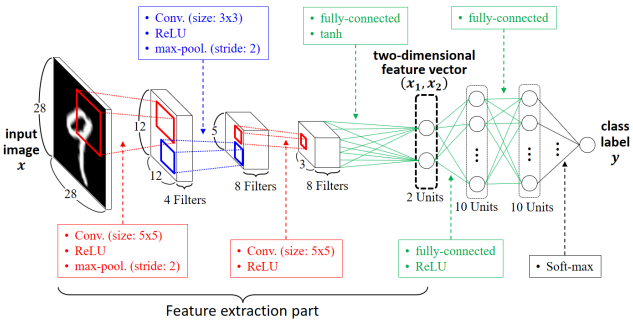

Fig. 3. Convolutional neural network used in preliminary experiment.

(a) Aiming to satisfy only $h^L(u_i) = f^L(u_i)$ for all $x \in X$.
(b) Aiming to satisfy not only $h^L(u_i) = f^L(u_i)$ but also $h^S(u_i) = f^S(u_i)$ for all $x \in X$.

The characteristic of the relationship between U_f^S and U_h^S highly depends on which strategy is employed by the constructor of h. Thus we have to consider both strategies.

B. Features for Detecting Cloned Recognizers

We made three hypotheses about the relationship between U_f^L and U_h^L and that between U_f^S and U_h^S. First, it can be naturally hypothesized that the output of f^L and that of h^L are consistent for most images if h is a CR of f. This would be satisfied in both the case (a) and the case (b) described above. Second, in the case (b), the output of f^S and that of h^S are also consistent (or quite similar with each other) for most images. Third, in the case (a), $h^S(x)$ tends to be much higher than $f^S(x)$ for most images $x \in X$. The reason can be explained as below.

Generally, two different classes often overlap with each other in a feature space in most pattern recognition systems because of outliers in the training dataset. Hence, the confidence score of the datapoint located near the decision boundary tends to be low. However, this is not satisfied in the case of CRs constructed with the strategy (a), because its training dataset was collected from a certain original recognizer and therefore has no outliers. Hence, the confidence score tends to be always high in the case (a).

To test the above three hypotheses, we conducted a preliminary experiment. In this experiment, we first designed a convolutional neural network shown in Fig. 3 and trained it with MNIST database [15]. Next, we input 30,000 images in MNIST database into the feature extractor part of the network and collected a labeled set of two-dimensional feature vectors $(x_1,\ x_2) \in [-1, 1]^2$. Using the feature vector set, we constructed two original recognizers by SVM and Random Forest (RF). The SVM-based original recognizer was trained under the setting of $C = 1$ with RBF kernel, and the RF-based one was consisting of 100 trees. Then, we also constructed two CRs of the SVM-based original recognizer, using k-nearest neighbor method (k-NN) and RF, respectively. The k-NN-based CR was constructed with the strategy (a) under the setting of $k = 100$. The RF-based CR was constructed with the strategy (b), which was actually a pair of a class label estimator and a confidence score regressor trained separately. Both the estimator and the regressor was consisting of 100 trees. Note that we used 30,000 vectors randomly sampled from the range of $[-1, 1]^2$ to train the two CRs. After that, we compared these four recognizers from the aspect of their outputs. Fig. 4 shows the comparison result.

As shown in the first row in Fig. 4, decision boundaries of the two different original recognizers (i) and (ii) are not similar with each other even when their training dataset is totally same. On the other hand, decision boundaries of the two CRs (iii) and (iv) are almost same with that of their original recognizer (i). This result supports the first hypothesis; that is, the output of f^L and that of h^L are too consistent if h is a CR of f. For the confidence score, the result of the CR (iv) is almost same with that of its original recognizer (i), as seen in the second row in Fig. 4. This supports the second hypothesis; that is, the output of f^S and that of h^S are too consistent if h is a CR constructed with the strategy (b). Moreover, in the result of the CR (iii), we can see that the confidence score is almost always high. This supports the third hypothesis; that is, $h^S(x)$ tends to be higher than $f^S(x)$ for most images $x \in X$ if h is a CR constructed with the strategy (a).

Based on the above consideration and experimental results, we propose the following α, β, and γ as the features for detecting CRs.

- Consistency rate of recognition results between f^L and h^L, i.e.,

$$\alpha = \frac{1}{U}\left|\left\{u_i \mid h^L(u_i) = f^L(u_i)\right\}\right| \tag{3}$$

- The amount of images taking smaller confidence score from h^S than f^S, i.e.,

$$\beta = \frac{1}{U}\left|\left\{u_i \mid f^S(u_i) - h^S(u_i) > 0\right\}\right| \tag{4}$$

- Variance of the difference between the confidence score of f^S and that of h^S, i.e.,

$$\gamma = \frac{1}{U}\sum_i \left\{f^S(u_i) - h^S(u_i)\right\}^2 \tag{5}$$

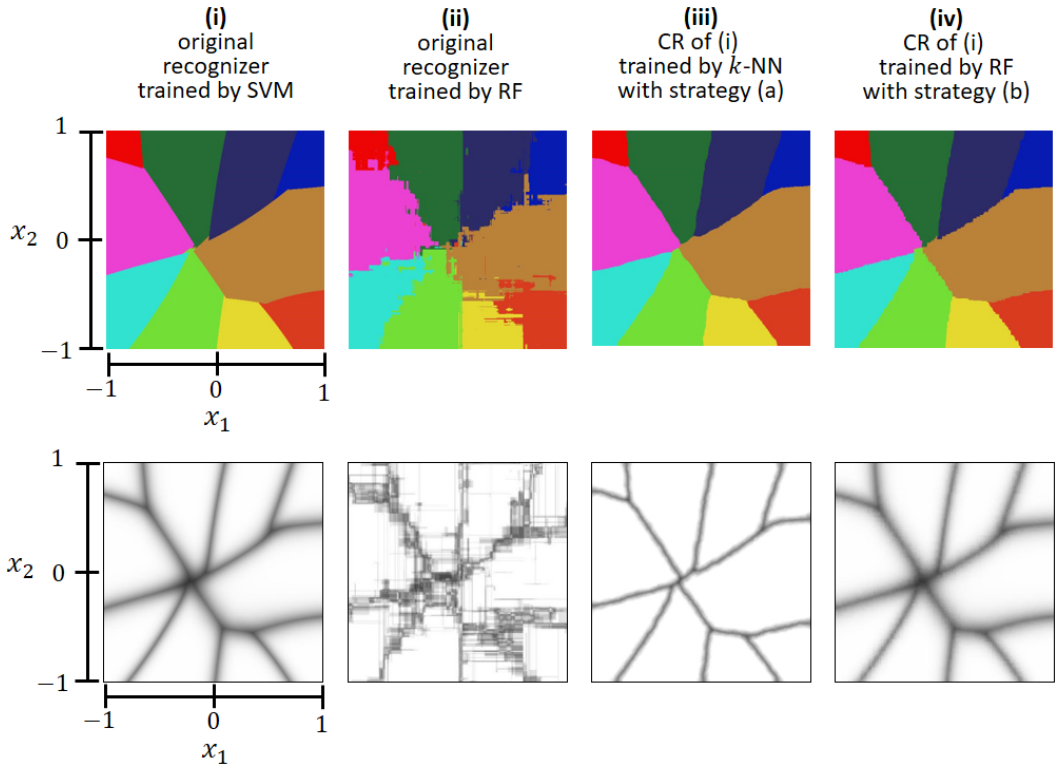

Fig. 4. Result of the preliminary experiment. Each square in this figure visualizes the recognition results and their confidence score output by the corresponding recognizer. The horizonral axis and the vertical axis of the square represents the value of x_1 and that of x_2, respectively. The color of the squares in the first row means the estimated class label ID for each $(x_1,\ x_2)$, while the color of the squares in the second row means the confidence score (white: high confidence, black: low cofidence).

The feature α would be too large if h is a CR of f. The feature β would be small if h is a CR constructed with the strategy (b). The feature γ would be too small if h is a CR constructed with the strategy (a). Using these three features as a three-dimensional feature vector, the proposed method trains a decision tree that judges whether h is a CR of f or not, which is used for detecting CRs afterward.

We also found from the preliminary experiment that it makes α and γ much effective to sample each u_i near the recognition boundary of the original recognizer f. This is because the datapoints far from the recognition boundary take almost same label and confidence score from f and h even when the h is not a CR of f, as shown in Fig. 4. Hence, we propose a way to collect the image set U as follows. First, for the feature β, we randomly sample each u_i from a uniform distribution. In practice, we can use an image set randomly collected from the Web as U. Next, for the features α and γ, we randomly sample two datapoints $v_i^{(1)}$ and $v_i^{(2)}$ from a uniform distribution to obtain each u_i so that $f^L(v_i^{(1)}) \neq f^L(v_i^{(2)})$ is satisfied. Then we repeat the following procedure:

1) Claculate $v_i = \frac{1}{2}(v_i^{(1)} + v_i^{(2)})$ and $f^L(v_i)$.
2) Compare $f^L(v_i)$ with $f^L(v_i^{(1)})$ and $f^L(v_i^{(2)})$.
3) If $f^L(v_i^{(1)}) = f^L(v_i)$, update $v_i^{(1)}$ as $v_i^{(1)} \leftarrow v_i$.

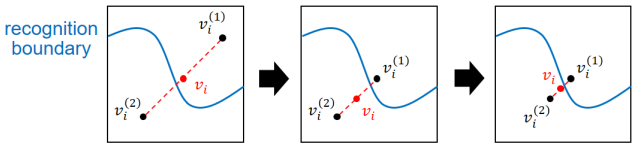

Fig. 5. Way to obtain datapoint near recognition boundary.

Otherwise, update $v_i^{(2)}$ as $v_i^{(2)} \leftarrow v_i$

After repeating the above procedure several times, we finally use v_i as u_i. Theoretically, this is the binary search of the recognition boundary between the initial $v_i^{(1)}$ and $v_i^{(2)}$ (see Fig. 5). In practice, we can use two images randomly collected from the Web as $v_i^{(1)}$ and $v_i^{(2)}$.

To further verify our three hypotheses, we actually computed the values of α, β, and γ between the following recognizer-pairs: [(i), (ii)], [(i), (iii)], and [(i), (iv)]. We note again that the recognizers (i) and (ii) are original, the recognizer (iii) is a CR of (i) constructed with the storategy (a), and the recognizer (iv) is a CR of (i) constructed with the storategy (b). The results are shown in TABLE I. As seen in this table, the value of α is more than 0.7 for the pairs of [(i), (iii)] and [(i), (iv)]; the class labels resulted from a CR is almost same with those resulted from its original, as

TABLE I
ACTUAL VALUES OF FEATURES α, β, AND γ COMPUTED BETWEEN THREE PAIRS OF RECOGNIZERS.

Recognizer pair	α	β	γ
(i) and (ii)	0.580	0.217	0.342
(i) and (iii)	0.866	0.050	0.172
(i) and (iv)	0.752	0.118	0.026

mentioned above. The value of β is less than 0.07 for the pair of [(i), (iii)]. Furthermore, the value of γ is less than 0.05 for the pair of [(i), (iv)], which indicates the confidence scores resulted from a CR is too consistent with those resulted from its original.

IV. EXPERIMENTS

We experimentally evaluated the proposed CR detection method using MNIST database [15] and CIFAR10 database [16]. For both databases, we first prepared a set of two-dimensional feature vectors using the same way with the above preliminary experiment. Next, with the feature vectors of MNIST images, we constructed four original recognizers, each of which was trained by SVM, k-NN, RF, and multi-layer perceptron (MLP) using randomly selected 30,000 samples. The parameter settings of SVM, k-NN, and RF were totally same with those used in the preliminary experiment. For MLP, we employed a network structure only containing 10 hidden fully-connected layers, each of which was consisting of 100 units with ReLU activation gate. This network was trained for 200 epochs. After that, for each of the four original recognizers, we constructed its CRs using k-NN, RF, and MLP. Note that we employed both of the strategies (a) and (b) to train the CRs, also using randomly selected 30,000 samples. In the case of strategies (b), we separately constructed a class label estimator and a confidence score regressor for each of the four machine learning models, i.e., k-NN, RF, and MLP, and the pair of the estimator and the regressor was used as a CR. The parameter settings of k-NN, RF, and MLP were totally same as above. As a result, we prepared six ($= 3 \times 2$) CRs for each original recognizer. With this procedure, we obtained 12 pairs of original recognizers and 24 pairs of a original recognizer and its CR. We conducted the same proceedure for the feature vectors of CIFAR10 images, thus finally obtained 24 pairs of original recognizers and 48 pairs of a original recognizer and its CR. At last, we equally divided a set of the above recognizer-pairs into two parts, one of which was used for training the decision tree for CR detection and the other was used for evaluating its accuracy, under the cross validation procedure. As the image set U that was used to compute the three features α, β, and γ, we used randomly selected 1,000 samples.

When dividing the set of the recognizer-pairs, we employed the following three division criteria, where the term "model" means one of SVM, k-NN, RF, and MLP.

(I) Divide the set of the recognizer-pairs so that the models of the original recognizers in the first subset are different those in the second subset.

TABLE II
DETECTION ACCURACY OF CRS BY PROPOSED METHOD. TP, TN, FP, AND FN ARE ABBREVIATION OF TRUE POSITIVE, TRUE NEGATIVE, FALSE POSITIVE, AND FALSE NEGATIVE.

Criterion for dividing the recognizer-pair set	(I)	(II)	(III)
Precision (TP/(TP+FP))	95.5%	90.9%	86.4%
Recall (TP/(TP+FN))	87.5%	83.3%	79.2%
Accuracy ((TP+TN)/(TP+TN+FP+FN))	88.9%	83.3%	80.6%

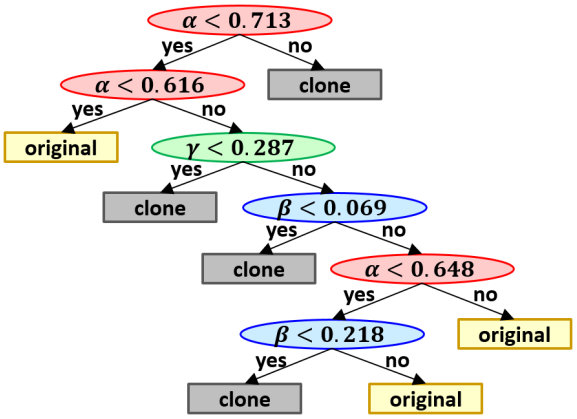

Fig. 6. Example of trained decision trees for detecting *cloned recognizers*.

(II) Divide the set of the recognizer-pairs so that the models of the CRs in the first subset are different those in the second subset.

(III) Divide the set of the recognizer-pairs so that both the condition of (I) and that of (II) are satisfied.

TABLE II shows the result of the experiment. As seen in this table, more than 80% of detection accuracy was achieved for all division criteria. We think this is a little insufficient for protecting cloud-based image recognition services in the real world. However, interestingly and importantly, the accuracy of the proposed method is not so degraded even when a recognizer trained by an unknown model is input to the decision tree-based detector. This fact shows the effectiveness of the proposed method, especially the three features α, β, and γ.

We showed an example of the decision trees obtained in this experiment in Fig. 6. The trained splitting rules seen in this figure are consistent with the hypotheses described in the previous section as well as the feature values shown in TABLE I. This also shows the effectiveness of the proposed three features.

V. CONCLUSIONS

In this paper, we proposed a method for detecting already trained CRs as an ex-post defense against RCA on cloud-based image recognition services. Our method is assumed to be used by a service owner: when he inputs his own original recognizer f and another (suspicious) one g owned by another person, the proposed method judges whether g is a CR of f or not by comparing them. With the preliminary experiment,

we found three characteristics of CRs, which are used as clues for CR detection. The effectiveness of the proposed method was shown through the experiments in which it was able to detect CRs with the accuracy of more than 80%.

However, since this is a pilot study, we experimentally evaluated the proposed methods only in the limited situations. Therefore, we will test the proposed method by further experiments and update them in a future work.

This work was supported by JSPS KAKENHI Grant Numbers JP17K00235 and JP16H06302.

REFERENCES

[1] Google Cloud Vision: https://cloud.google.com/vision/
[2] Amazon Rekognition Image: https://aws.amazon.com/rekognition/image-features/
[3] L. Huang, A.D. Joseph, B. Nelson, B.I.P. Rubinstein, and J.D. Tygar: "Adversarial Machine Learning," in Proc. of 4th ACM Workshop on Security and Artificial Intelligence, pp.43–58, 2011.
[4] F. Tramer, F. Zhang, A. Juels, M.K. Reiter, and T. Ristenpart: "Stealing Machine Learning Models via Prediction APIs," in Proc. of 25th USENIX Security Symposium, pp.601–618, 2016.
[5] B. Nelson, M. Barreno, F.J. Chi, A.D. Joseph, B.I.P. Rubinstein, U. Saini, C. Sutton, J.D. Tygar, and K. Xia: "Exploiting Machine Learning to Subvert Your Spam Filter," in Proc. of 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats, pp.1–9, 2008.
[6] M. Kloft and P. Laskov: "Online Anomaly Detection under Adversarial Impact," in Proc. of 13th Int'l Conf. on Artificial Intelligence and Statistics, pp.405–412, 2010.
[7] B. Biggio and P. Laskov: "Poisoning Attacks against Support Vector Machines," in Proc. of 29th Int'l Conf. on Machine Learning, 8 pages, 2012.
[8] M. Fredrikson, S. Jha, and T. Ristenpart: "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures," in Proc. of 22nd ACM SIGSAC Conf. on Computer and Communications Security, pp.1322–1333, 2015.
[9] A. Kurakin, I. Goodfellow, and S. Bengio: "Adversarial Examples in the Physical World," arXiv:1412.6572v3, 2016.
[10] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z.B. Celik, and A. Swami: "Practical Black-Box Attacks against Machine Learning," in Proc. of 2017 ACM Asia Conf. on Computer and Communications Security, pp.506–519, 2017.
[11] H. Yang: "Near-Duplicate Detection by Instance-level Constrained Clustering," in Proc. of 29th ACM Conf. on Research and Development in Information Retrieval, pp.421–428, 2006.
[12] S. Raveena and V. Nandini: "Near Duplicate Document Detection Using Document-Level Features and Supervised Learning," in Proc. of Int'l Conf. on Global Innovations in Computing Technology, 9 pages, 2014.
[13] I.D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier: "Clone Detection Using Abstract Syntax Trees," in Proc. of Int'l Conf. on Software Maintenance, 10 pages, 1998.
[14] Z. Li, S. Lu, S. Myagmar, and Y. Zhou: "CP-Miner: Finding Copy-Paste and Related Bugs in Large-Scale Software Code," IEEE Trans. on Software Engineering, Vol.32, No.3, pp.176–192, 2006.
[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner: "Gradient-Based Learning Applied to Document Recognition," IEEE Trans. on Software Engineering, Vol.86, No.11, pp.2278–2324, 1998.
[16] A. Krizhevsky: "Learning Multiple Layers of Features from Tiny Images," Master's thesis, Department of Computer Science, University of Toronto, 2009.