Recommended Paper

Model Extraction Attacks on Recurrent Neural Networks

Tatsuya Takemura[1,a)] Naoto Yanai[1,b)] Toru Fujiwara[1,c)]

Received: January 31, 2020, Accepted: September 10, 2020

Abstract: Model extraction attacks are an attack in which an adversary utilizes a query access to the target model to obtain a new model whose performance is equivalent to the target model efficiently, i.e., fewer datasets and computational resources than those of the target model. Existing works have dealt with only simple deep neural networks (DNNs), e.g., only three layers, as targets of model extraction attacks, and hence are not aware of the effectiveness of recurrent neural networks (RNNs) in dealing with time-series data. In this work, we shed light on the threats of model extraction attacks on RNNs. We discuss whether a model with a higher accuracy can be extracted with a simple RNN from a long short-term memory (LSTM), which is a more complicated and powerful type of RNN. Specifically, we tackle the following problems. First, in case of a classification task, such as image recognition, extraction of an RNN model without final outputs from an LSTM model is presented by utilizing outputs halfway through the sequence. Next, in case of a regression task such as weather forecasting, a new attack by newly configuring a loss function is presented. We conduct experiments on our model extraction attacks on an RNN and an LSTM trained with publicly available academic datasets. We then show that a model with a higher accuracy can be extracted efficiently, especially through configuring a loss function and a more complex architecture different from the target model.

Keywords: model extraction attacks, deep neural networks, recurrent neural networks, long short-term memory

1. Introduction

1.1 Backgrounds

Deep learning is a state-of-the-art technology for machine learning and is known to provide various advantages in many areas. Deep learning requires a heavy computational load and therefore a business style called machine-learning-as-a-service (MLaaS), where a machine learning model is hosted via a public server, is recently the subject of a great deal of attention. Well-known MLaaS include AWS [*1] and Microsoft Azure [*2]. In such a situation where a machine learning model consists of two tasks, i.e., training and prediction, a trained model is stored in a public server, e.g., cloud server, and a client requests the model to run a prediction task via APIs.

However, the execution of prediction tasks via APIs may leak information about a model to a client. Model extraction attacks [27] have received attention in recent years from the standpoint of information leakage described above. In particular, an adversary who mimics a client trains their own model by utilizing APIs of a machine learning model hosted by a public server called an original model, and its prediction results. The trained model by the adversary is called a substitute model. The goal of the adversary is to obtain a local copy of a machine learning model with a higher accuracy even when the adversary owns less data than the public server of the original model [14], [21]. In general, benefits for the adversary are significant because gathering data and its training are tasks involving heavy costs. Moreover, according to Juuti et al. [14], transferable adversarial examples [25] to analyze misidentifiable predictions via a substitute model have been discussed as applications of model extraction attacks. Consequently, a model extraction attack is a serious problem for deep learning and its underlying machine learning.

In spite of the significance of model extraction attacks, only simple architectures such as a logistic regression model [27] or deep neural networks (DNNs) with simple architectures [14], [15] have been discussed in existing works. Thus, the features and feasibility of model extraction attacks on other architectures are unclear. For instance, threats of model extraction attacks are non-trivial for recurrent neural networks (RNNs) which are used in natural language processing and cybersecurity applications. In particular, the computational process of deep learning differs according to the architecture and therefore the success conditions and advantages of an adversary may differ according to the architecture as well. Hence, discussion on model extraction attacks for various architectures is an important research theme for avoiding many potential threats stemming from attacks such as the adversial example via a substitute model as described above.

[1] Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[a)] t-tatsuya@ist.osaka-u.ac.jp
[b)] yanai@ist.osaka-u.ac.jp
[c)] fujiwara@ist.osaka-u.ac.jp

The preliminary version of this paper was published at Multimedia, Distributed, Cooperative, and Mobile Symposium (DICOMO 2019), July 2019. The paper was recommended to be submitted to Journal of Information Processing (JIP) by the chief examiner of SIGCSEC.

[*1] https://aws.amazon.com/jp/aml/
[*2] https://azure.microsoft.com/ja-jp/services/machine-learning-studio/

1.2 Contribution

In this paper, we introduce model extraction attacks on RNNs and long-short term memory (LSTM) and show that RNNs with a higher prediction accuracy as a substitute model by an adversary, can be obtained from LSTM as an original model by a public server or other means. Our technical contributions include finding attacks based on features of RNNs and LSTMs for a classification task and a regression task. Substitute models with higher accuracy can be obtained based on the features of RNNs and LSTMs, which compute an output each time and give feedback for next usage and also take into account those loss functions to obtain a final output (See Section 4). Despite RNNs having simpler architectures than LSTMs, an adversary can train a substitute model with a high accuracy without final outputs from an original model by biasing the outputs halfway through the sequence.

We also conduct experiments with the MNIST dataset [*3] for a classification task and with the Air Quality dataset [*4] for a regression task to show the ability to extract RNNs as substitute models from LSTMs as original models. The results show that for MNIST datasets, a substitute model can be extracted with 97.5% accuracy with only 20% of the traning data compared to 97.3% for the original model. Next, on the Air Quality dataset, the substitute model with 87.1% accuracy, which is computed by the coefficient R^2 of determination, can be extracted with just three months of training data compared with the original model which achieves 86.8% accuracy with training data for nine months. We also discuss relationships between architectures and those features and shed light on the success factors for model extraction and countermeasures (See Section 5 for details).

1.3 Related Work

State-of-the-Art Attacks: As the latest results, Reith et al. [20] discussed model extraction on support vector regression. For works targeting neural networks, Juuti et al. [14] showed an attack in which an adversary generates queries for DNNs with simple architectures. Their attack might improve our results on RNNs but we leave this as an open problem for RNNs. Concurrently, Wang et al. [28] proposed model extraction attacks to steal hyperparameters from a simple architecture, e.g., a neural network with three layers. To the best of our knowledge, the most elegant attack was shown by Okada and Hasegawa [21]. They utilized *distillation* [3], [11], which is a technique for model compression described below to execute model extraction attacks against DNNs and convolution neural networks (CNNs) for image classification. In doing so, Okada and Hasegawa succeeded in extracting a model with a higher accuracy than the original model. We therefore consider their work as having the best results from the standpoint of the use of distillation.

Relationships between Architectures and Accuracy: One of the main discussions in this paper is how to clarify relationships between accuracy and architectures on an original model and a substitute model. Similar discussions have been provided by Juuti et al. [14], Pal et al. [23], Krishna et al. [16], and Okada et al. [21]. The papers in Refs. [14], [23] explain that the accuracy of the substitute model increases in general when the architecture of an original model is identical to that of a substitute model. Pal et al. [23] have claimed that their argument is true unless underfitting or overfitting is caused on a substitute model, whose architecture is more complicated than that of the original model, e.g., the use of a deeper network in the substitute model than in the original model. In contrast, the results in Ref. [16] showed that, when BERT [6] utilized in natural language processing is targeted, the accuracy of the substitute model is improved by use of a BERT model as a substitute model that is deeper than the original model. The results in Ref. [21] also showed a similar result in the case of DNNs and CNNs. Our results are identical to the results in Refs. [16], [21], except for the use of RNNs.

Distillation: As further related works, model compression, named distillation [3], is represented. Distillation is used for reducing learning information by multiple neural networks, which are called teacher models, to smaller neural networks, which are called student models. Hinton et al. [11] showed a method to distill a model by a *softmax function with temperature* which can control the convergence of training through temperature. While distillation allows a student model to extract a large amount of information from a teacher model, model extraction attacks require that an adversary has as little access as possible to a dataset and an original.

Additional Features on Model Extraction: As one of the latest features on model extraction attacks, the Knockoff nets attack [22] discusses how an adversary attempts model extraction based solely on observed input-output pairs, i.e., without any knowledge of the dataset of the original model. However, Atli et al. [10] showed that the performance of the Knockoff nets attack is limited. Meanwhile, Jagielski et al. [13] proposed a new feature named fidelity to measure the general agreement between an original model and a substitute model. Discussions on these features on RNNs remain an open problem.

Further Attacks for Assisting in Model Extraction: Naghibijouybari et al. [19] and Yoshida et al. [30] introduced side-channel attacks targeting models separated by hardware mechanisms. Model reverse-engineering attacks [2] where an original model is operated in an environment owned by an adversary have also been shown as advanced attacks. These attacks are stronger than the attack scenario in this work because we do not discuss such physical access to an original model for an adversary. Our attack against RNNs can potentially become stronger by utilizing the side-channel attack described above.

1.4 Paper Organization

The rest of this paper is organized as follows. First, the background required for understanding this paper is presented in Section 2. Next, an attack model and the proposed attacks against RNNs are presented in Section 3. Then, experiments are shown in Section 4, and considerations including potential countermeasures are shown in Section 5. Finally, the conclusion and future directions are presented in Section 6.

2. Preliminaries

In this section, we provide a background for understanding our

[*3] http://yann.lecun.com/exdb/mnist/
[*4] https://archive.ics.uci.edu/ml/datasets/air+quality

work.

2.1 Tasks Specified in Neural Networks

The mechanism of a neural network varies depending on the task to be solved and loss functions in particular differ for each task. Hence, it is necessary to discuss attack techniques separately. We describe the two types of typical tasks handled by neural networks below.

2.1.1 Classification

A classification task is a task where an input is classified into one of the categories specified in advance as a prediction output. For example, a neural network for classification categorizes a number from 0 to 9 when a handwritten digit is given as input. The number of neurons in the neural network required to solve such a classification task is identical to the number of candidates in an output layer, and decides which neuron has the largest calculation result. The training is often executed through the softmax function to return candidates along with their probablities.

2.1.2 Regression

A regression task outputs continuous values as a response to input. For example, a neural network for prediction of the air quality outputs a numerical value of a component of the air quality at a certain time of the given input for the previous several hours. The neural network that solves a regression task outputs a numerical value on an output layer. Here, the neural network has plural neurons in proportion to the number of candidates predicted on the output layer. Unlike the classification task, values of neurons in the output layer are computed without the softmax function.

2.2 Recurrent Neural Networks

2.2.1 Principle of Recurrent Neural Networks

Recurrent neural networks (RNNs) are a kind of neural network that deals with time-series data including contexts, e.g., speech recognition and language processing. For example, when gender is estimated by speech waveforms of human voices, a continuous waveform can be discretized by short time intervals as shown in **Fig. 1**. In this figure, inputs are set as $x_1, x_2, \cdots, x_t$, where x_i is the input at time i as time-series data.

RNNs have negative feedback in those networks as shown in **Fig. 2**. In comparison with architectures of typical neural networks such as deep neural networks (DNNs), RNNs have an input layer, hidden layers, and an output layer whereas they also have a feedback path to return the output of the hidden layer to the input itself.

We describe RNNs in detail below. An input to the RNN is made at each time t. An input given at time t propagates from the input layer to the hidden layer in a similar manner as in conventional neural networks. An output of the hidden layer with an activation function propagates to the output layer and returns in paralle with the input of the hidden layer itself as feedback. The signal propagated to the output layer is output as a prediction result at time t, whereas the feedback is given to the hidden layer as a part of an input at the next time $t+1$. Consequently, the output at $t+1$ is affected by the outputs of the hidden layer before time t, and hence is able to capture contexts of the time-series data. Unlike typical neural networks which approximate a mapping from a single input to a single output, an RNN approximates a mapping from a sequence to a sequence and returns outputs each time. RNNs are used in both classification and regression tasks, and often deal with the regression task that predicts continuous values such as for stock prices prediction [8].

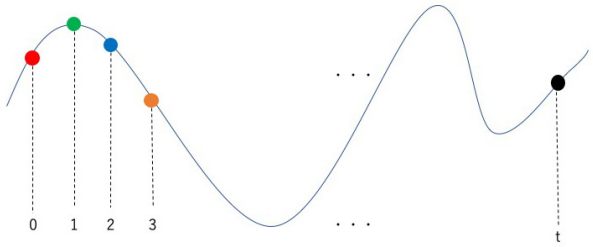

Fig. 1 Example of an audio waveform.

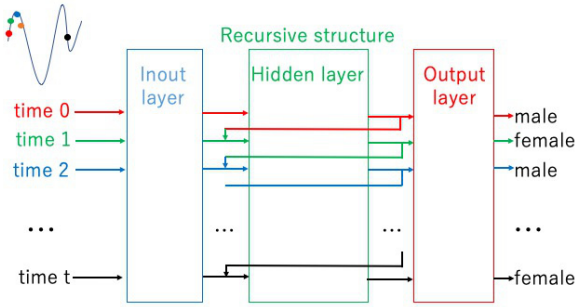

Fig. 2 Architecture of an RNN.

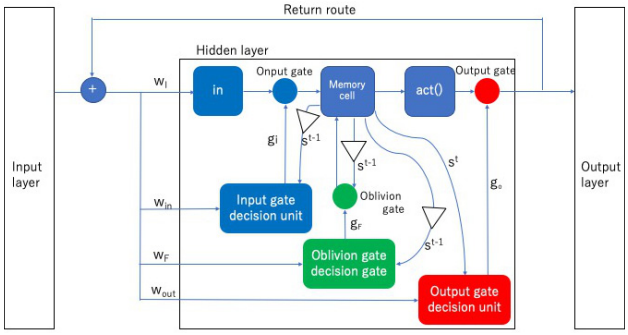

Fig. 3 Memory unit of an LSTM.

2.2.2 Long Short-Term Memory

In general, neural networks with deeper layers have a gradient loss problem [9]. Since RNNs have a recursive structure in a hidden layer, information propagates deeper when input time-series data serving as the input becomes longer, even for shallow networks. Consequently, RNNs are prone to the gradient loss problem or mainly the memory data is difficult to retain for a long period of time. Long short-term memory (LSTM) has been proposed to overcome the gradient loss problem in RNNs, which only store memory for a short period. The basic architecture of LSTMs is the same as that of RNNs, except that a hidden layer with a recursive structure of RNNs is replaced with a layer with an element called a memory unit which is shown in **Fig. 3**.

Compared to standard RNNs, LSTMs have greater computational complexity because they have more complicated architectures. Furthermore, LSTMs are different types depending on the

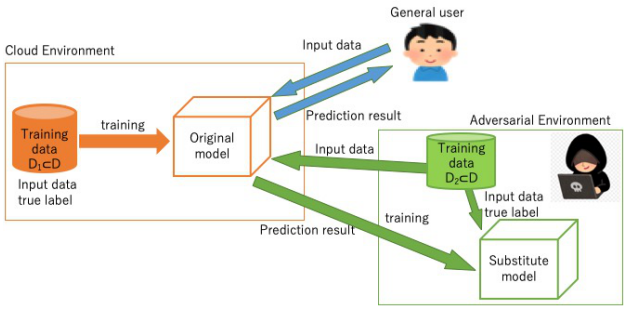

Fig. 4 Overview of model extraction attacks.

types of inputs and outputs. For instance, LSTMs have the following three types. The many-to-many type has inputs and outputs at each time. The many-to-one type has inputs given at each time and a single output at only the last time. The one-to-many type has a single input at only the first time and an individual output at each time. Typical neural networks have a single output for each given input and can therefore be considered a one-to-one type.

2.3 Model Extraction Attacks

We briefly describe an overview of model extraction attacks below. Suppose that a model stored on a public server is trained using training data $D_1 \subset D$. That trained model is thereafter called the original model. General users pay the server to be allowed to utilize its hosting service and give data as input via APIs of the server. This data is input to the original model, and then the original model returns prediction results to a user as a response from the service.

Based on this background, an adversary, who knows a part of the training dataset $D_2 \subset D$, executes the original model through the API to train their own model by utilizing prediction results from the original model for D_2. The model trained by the adversary is called a substitute model. For example, the adversary can train a substitute model by using prediction results and computational resources of the original model as a springboard to obtain the same or higher accuracy as that of the original model. This is the concept for an attack strategy for model extraction attacks. **Figure 4** shows an overview of the attacks.

The main advantage of model extraction attacks for an adversary is to obtain a model with significantly reduced costs for both data collection and its resulting training. In general, data collection and training tasks involve heavy costs, and hence resulting models become an important asset for a provider of a public server hosting an original model. In contrast, the adversary can obtain a substitute model whose performance can be the same as the original model without paying such expensive costs. Moreover, the state-of-the-art work in Ref. [21] has shown that model extraction attacks enable an adversary to obtain a substitute model with higher accuracy than the original model.

3. Model Extraction Attacks against Recurrent Neural Networks

3.1 Problem Setting

We describe the technical problems and conditions for model extraction attacks against RNNs as the main problem setting below.

Computational Resources An original model is provided on a public server with a rich computational resource, whereas an adversary, who executes model extraction attacks, needs to train a substitute model with fewer resources.

Input/Output In contrast to deep neural networks (DNNs), RNNs contain an input and an output for each time. These inputs and outputs are utilized halfway through the sequences in feedback to compute a final output. We discuss how an adversary obtains an advantage by way of model extraction attacks from the inputs and outputs halfway through the sequences.

Regression As described in Section 2.2, RNNs are often utilized for a regression task. Existing model extraction attacks have been discussed mainly about classification tasks such as image classification. A softmax function is utilized in neural networks for a classification task but it cannot be used for a regression task. Therefore, known techniques [21] that modify the softmax function to decrease the number of queries cannot be used. Moreover, since an adversary who owns several parts of a dataset used in an original model may know the correct result for each output in advance, the merit of the information obtained from the APIs may be of only limited value compare with DNNs. To effectively utilize information from RNNs, a loss function for a regression task should be constructed in detail.

In this paper, we evaluate the accuracy of model extraction attacks against RNNs from the standpoints described above.

3.2 Attack Strategy

In this section, we describe model extraction attacks based on features of RNNs.

As described in the previous section, the two features called computation resources and input/otuput should be considered for model extraction attacks on RNNs. First, the architecture of an LTSM is more complicated than that of an RNN. Hereafter, we simply denote RNNs with a simple architecture as RNNs. Second, in comparison with other neural networks such as DNNs or convolutional neural networks (CNNs), an output with variable length is generated for each time in RNNs.

Hence, we discuss model extraction attacks against RNNs from the following standpoints:

(1) Can a substitute model consisting of RNN be extracted from an original model consisting of LSTM?

(2) Can an adversary obtain any advantage by utilizing features of input/output for RNNs?

We describe the details of the attacks below. Let training data used in an original model be D_c and training data used in a substitute model be D_a. Here, the original model, i.e., an LSTM, is trained by utilizing D_c. Then, an adversary trains the substitute model, i.e., an RNN, with D_a whose distribution is close to D_c and possibly $D_a \subseteq D_c$, and then continues to train RNN using the obtained prediction results from the original model by giving input data.

One might think that an assumption in which an adversary has

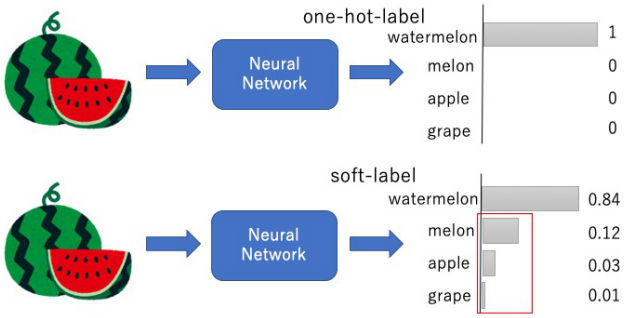

Fig. 5 Soft-label encoding and one-hot encoding.

a part of a dataset utilized in an original dataset, e.g., $D_a \subseteq D_c$, is very strong. However, according to Wang and Zhenqiang [28], this setting is rather realistic in machine-learning-as-a-service (MLaaS) such as Amazon Machine Learning and Microsoft Azure Machine Learning, where a machine learning model is hosted via a public server [*5]. In particular, in the MLaaS setting, a machine learning model hosted by a public server utilizes data provided by users as the training data for constructing a large amount of dataset. This means that an original model on the server is trained with data that each user owns in local. In doing so, an adversary or namely a malicious user can access to a part of training data because the user owns the data in local.

3.2.1 Attack on Classification Task

In general, a neural network that solves a classification task returns an output without the use of a softmax function in its output layer for prediction. We call such an output or namely values not obtained through a softmax function as logits [9]. While logits are soft-label encoding, labeled data utilized in training an original model and a substitute model, i.e., D_c and D_a, are one-hot encoding that represents just the true value for each label. The concept of soft-label encoding and one-hot encoding is shown in **Fig. 5**. In this situation, an adversary executes the following attack procedure:

(1) Identification of Leaky Time for the Original Model: The adversary identifies the maximized index on each vector for logits returned from the original model from the first time to the last time and then evaluates the prediction accuracy for each time by comparing those labels. A time with high accuracy is defined as a leaky time.

(2) Intensive Extraction at Leaky Time: For the leaky time described in the previous item, the softmax function with temperature [11] is utilized. In particular, the adversary first trains a substitute model by computing a loss function with labeled data, i.e., one-hot labels included in D_a to update the parameters. Then, the adversary computes the loss function in which soft-labels through the softmax function with temperature are set as labeled data to update the parameters.

The softmax function with temperature is defined in Eq. (1) and its output is shown in **Fig. 6**. This function generally behaves identically to the original softmax function when $T = 1$, and a gradient becomes smaller in proportion to a temperature T, or namely so that convergence of training can become faster in proportion to T. In this paper, the softmax function with temperature is utilized and, in doing so, the effects on model extraction attacks are evaluated by changing T.

[*5] https://analyticsindiamag.com/10-machine-learning-service-mlaas-tools-data-scientists/

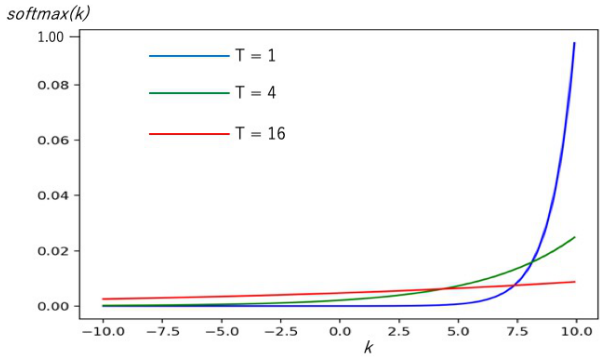

Fig. 6 Graphs for softmax function with temperature. Let T be temperature. Baseline is $T = 1$, which is identical to the original softmax function. In proportion to T, a derivative on each curve is different even with the same input k of the output layer.

$$\mathrm{softmax}(k) = \frac{e^{\frac{a_k}{T}}}{\sum_{i=1}^{n} e^{\frac{a_i}{T}}}. \quad (1)$$

Meanwhile, we focus on image classification with only RNNs although typical image classification with RNN is based on a combination with CNNs [29]. The purpose of our paper is to properly evaluate the accuracy of model extraction attacks to RNNs, and hence we focus only on features of the RNNs themselves. Namely, experiments and discussion described later are conducted only with RNNs and a combination of RNNs with CNNs is beyond the scope of this work.

3.2.2 Attack on Regression Task

A neural network that solves a regression task returns a predicted value computed by a model, and the output is drastically different from a model for a classification task whose output is labeled with probabilities. Consequently, due to the structure of an output layer, a softmax function cannot be used as a loss function. In general, norms are utilized in a loss function for a model to solve a regression task, e.g., $L_1 loss$ or $L_2 loss$.

The softmax function with temperature and $L_2 loss$ has been utilized in the distillation of neural networks [1], [5]. However, while the softmax function with temperature has been used in model extraction attacks by Okada and Hasegawa [21], the use of $L_2 loss$ in model extraction attacks is non-trivial. In this paper, we utilize $L_2 loss$ in the model extraction attacks for a regression task.

While an output of the softmax function with temperature is distributed within [0,1] even for a corrupted prediction, a loss function with the norm does not have any restriction in the output range. That is, a student model in distillation, or namely a substitute model for a model extraction attack, may be affected greatly by a corrupted prediction from a teacher model, i.e., an original model. To overcome the limitation described above, instead of the use of outputs from the teacher model, we focus on the method by Chen et al. [5] which utilizes the outputs as an upper bound to be achieved for the student model. As shown in Eq. (2), a penalty for adjusting the parameter is given for the output only when $L_2 loss$ between a predicted value R_t for the teacher model and labeled

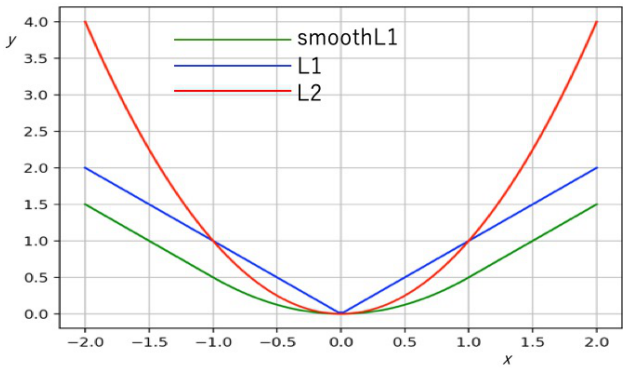

Fig. 7 Lossgen graphs for loss functions. We denote by $y = norm(x)$ the norm function with respect to each norm of $smoothL1$, $L1$ and $L2$, respectively.

data y is smaller than $L_2 loss$ between a predicted value R_s for the student model and y with respect to a parameter m designated in advance. The function is called teacher bound regression loss and is denoted by L_b. Here, m is a small value estimated by distribution for each dataset.

$$L_b(R_s, R_t, y) = \begin{cases} ||R_s - y||_2^2, & \text{if} \quad ||\mathrm{R_s} - \mathrm{y}||_2^2 + \mathrm{m} > ||\mathrm{R_t} - \mathrm{y}||_2^2 \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Furthermore, instead of the use of L_b as a loss function for the student model, L_b is embedded in the use of smooth L1 loss, L_{s1}, which is defined in Eq. (3). The smooth L1 loss L_{s1} can overcome the problem where the derivation is impossible by converging to zero on the L1 loss and the problem where a gradient becomes too large in proportion to the distance on L2. The distribution for each loss function is shown in **Fig. 7**.

$$L_{s1}(R_s, y) = \begin{cases} 0.5 \times |R_s - y|_1^2, & \text{if} \quad |\mathrm{R_s} - \mathrm{y}|_1 < 1 \\ |R_s - y|_1 - 0.5, & \text{otherwise.} \end{cases} \tag{3}$$

Then, the loss function L_{reg} of a student model for a regression task is defined in Eq. (4):

$$L_{reg}(R_s, R_t, y) = L_{s1}(R_s, y) + v \times L_b(R_s, R_t, y). \tag{4}$$

For model extraction attacks, the teacher model described above is dealt with by an original model stored in a public server while the student model is dealt with by a substitute model. In doing so, L_{reg} is utilized as a loss function to train the substitute model. Since an adversary does not know the outputs identical to the input data on typical neural networks, an output returned from an original model is important information for the adversary. Hence, discussion about the details in loss functions described above is often unnecessary.

Meanwhile, in the case of RNNs, the following discussion is necessary due to features of their input and output. For RNNs of the many-to-many type, an adversary can correctly guess the predicted data returned from an original model through its own data because both input data and output data are included in the same dataset as time-series data. In other words, an adversary who owns a part of a dataset can train a substitute model with its own input and labeled data. In doing so, to give a more generalized performance to the substitute model, knowledge of the original model trained with a larger amount of data should be extracted by the adversary. We evaluate the loss function L_{s1} with respect to the extraction of knowledge described above.

4. Experiment

In this section, we conduct experiments on model extraction attacks on the RNN described in the previous section to evaluate their effectiveness in terms of the accuracy of prediction in measuring the correctness of predictions on the test distribution. In particular, we discuss neural network architectures for both classification and regression tasks.

4.1 Experiment Setup

The experimental environment is shown in **Table 1**. We configured the environment on the Google Colaboratory [*6]. The training algorithm of neural networks used in the experiments is the Adam optimizer which is standard equipment for TensorFlow [*7] with a learning rate of 0.001.

4.1.1 Settings for the Classification Task

We utilize the MNIST dataset [*8] in an experiment on model extraction attacks against a many-to-many LSTM. The MNIST dataset used consists of 55,000 samples as training data and 11,000 samples as test data. Each sample represents a handwritten character from 0 to 9 and is represented as 28×28 pixels. **Figure 8** [*9] shows examples of the samples on the MNIST dataset. This dataset has been used in many works such as model extraction [13], [14] and distillation [11].

In the experiment described below, each sample on the MNIST dataset is converted into time-series data. As shown in **Fig. 9**, each sample of 28×28 pixels is divided into lines for each time sequentially, where we assume the t-th line is input at time t. In other words, there are 28 lines of input data, and each line is given to a model as time-series data for times 1 to 28.

As neural networks for classification of handwritten digits, DNNs and CNNs have been discussed by Okada and Hasegawa with respect to model extraction attacks based on the softmax function with temperature. To compare with the work by Okada and Hasegawa, we adopt the same experimental setting as in their work. In particular, as shown in **Fig. 10**, 55,000 samples as training data in the MNIST dataset are divided into five subsets, or namely 11,000 samples per subset. Let the subsets be denoted as D_1, D_2, D_3, D_4, and D_5 for convenience. Four of the subsets are utilized as training data D_c for an original model, and the remaining subset is utilized as training data D_a for a substitute model by an adversary. Experiments are conducted five times because there are five cases in which each subset is used as D_a, and then the final results of the entire experiment are the averages of the results of the five experiments, or namely 5-fold cross validation.

Since the MNIST dataset is an academic benchmark for a classification task, as described in Section 3.2.1, we utilize cross-entropy error through the softmax function with temperature as a

[*6] https://colab.research.google.com
[*7] https://www.tensorflow.org/
[*8] http://yann.lecun.com/exdb/mnist/
[*9] https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/

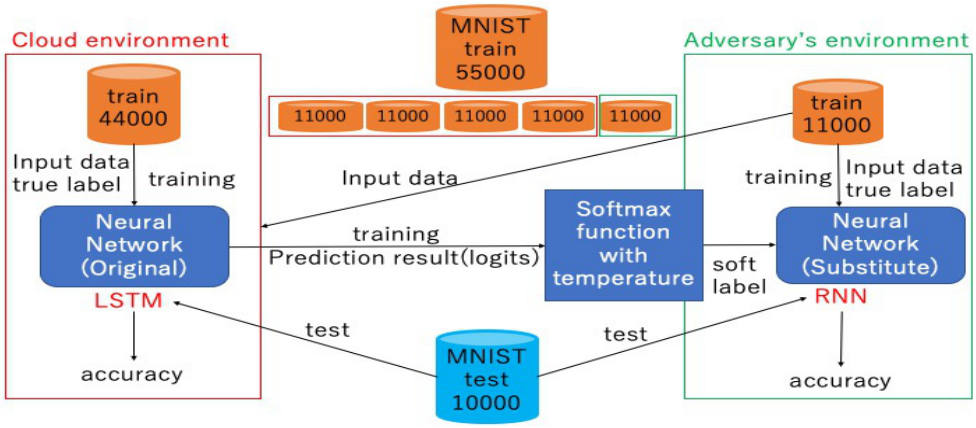

Fig. 10 Environments of original and substitute models with MNIST dataset.

Table 1 Experimental environment.

Development Platform	Tensor Flow 2.0.
OS	Ubuntu 18.04
GPU	NVIDIA Tesla K80 12 GB
Memory	13 GB RAM
Storage	360 GB

Fig. 8 Example of MNIST dataset.

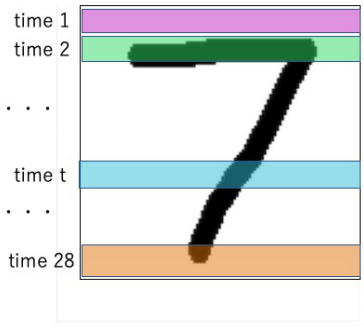

Fig. 9 Conversion of MNIST dataset to time-series data.

loss function. Here, let temperatures be $T = 1, 4, 16$. For $T = 1$, the softmax function with temperature is exactly identical to the original softmax function as described above. After training the substitute model with labeled data using the cross-entropy error, the substitute model is trained as a loss function utilizing the softmax function with temperature by using logits values obtained from the original model.

The accuracy of prediction is evaluated with 10,000 samples as test data of the MNIST dataset with respect to both the original model of LSTM and the substitute model of RNN. According to our pre-experiment, the accuracy of the original model is 97.3% although we omit the details. The accuracy is the attack goal of the substitute model for an adversary. The setting for training in this experiment is shown in **Table 2**.

Table 2 Setting for training in case of classification task.

Epoch	220
Iterations in Each Epoch	$\frac{\lvert D_a \rvert}{50}$
Batch Size	50

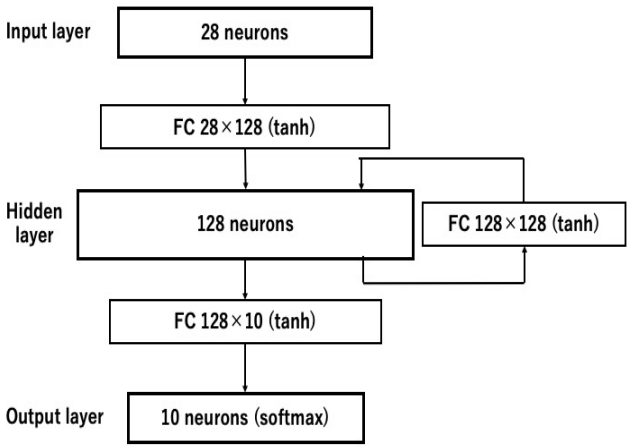

Fig. 11 The architecture of the neural network used on the experiment in classification task.

Figure 11 shows the architecture of the neural network used on the experiment classification task. The hidden layer is a single RNN cell. The outputs of the RNNcell neurons at time t and the inputs to the RNNcell neurons at time $t + 1$ are connected by a full connection, i.e., FC. We use tf.contrib.rnn.BasicLSTMCell and tf.contrib.rnn.BasicRNNCell as a hidden layer of RNN and LSTM, respectively, and use tf.contrib.rnn.static_rnn for network input/output.

Baseline of the Experiment on the Classification Task

The following setting is utilized as baselines to compare the performance of our attack.

- Temperature $T = 1$: This is identical to the original softmax function where the distillation is not used.
- Final output at time 28: This is precisely identical to an output on the output layer.

4.1.2 Settings for the Regression Task

For experiments on model extraction attacks against the LSTM to solve the regression task, we deal with an Air Quality

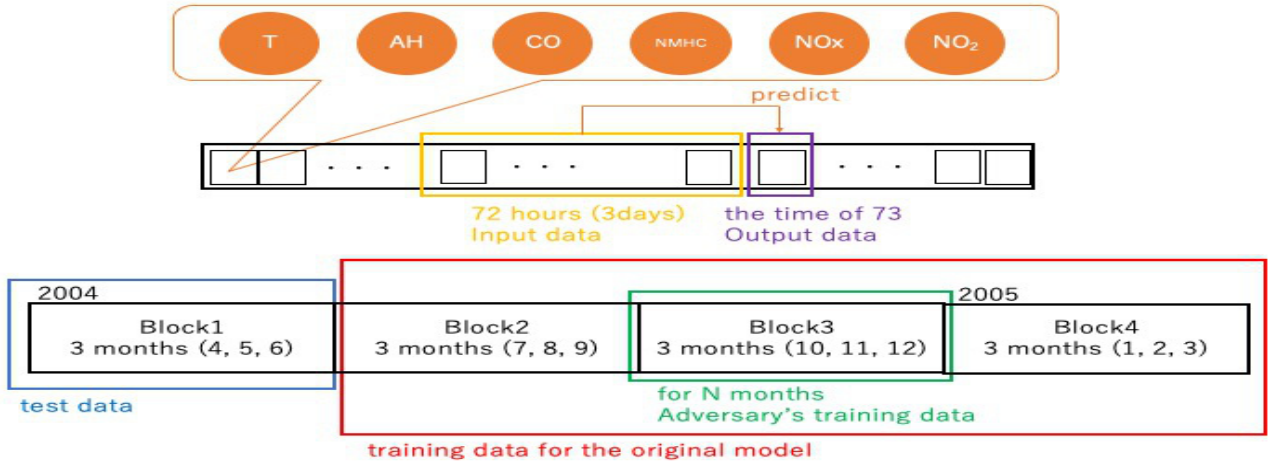

Fig. 13 Dataset of test, original and substitute models with Air Quality dataset.

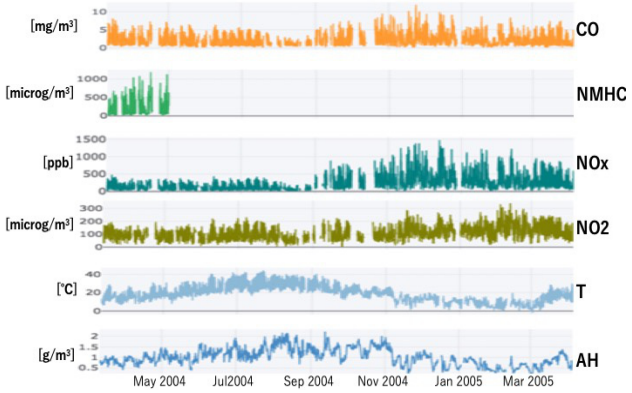

Fig. 12 Examples of the Air Quality dataset.

dataset [*10], which consists of the amount of materials contained in the atmosphere and temperatures collected by sensor devices. The Air Quality dataset was measured every hour from 18:00 on 03/10/2004 to 14:00 04/04/2005 and each record per hour consists of 13 kinds of values as the amount of materials and temperature. Examples of the Air Quality dataset are shown in **Fig. 12** [*11]. This dataset has been utilized in several works on analysis of the air quality [12], [17], [18].

A dataset that minimizes missing values is desirable because we handle time-series data, and thus a reliable experimental result can be expected by using the Air Quality dataset as follows. We utilize six values, i.e., temperature T, absolute humidity AH, time average value CO of oxide for carbon monoxide, time average value $NMHC$ of titanium oxide for non-methane hydrocarbons, time average value NO_x of tungsten oxide for nitrogen oxide, and time average value NO_2 of tungsten oxide for nitrogen dioxide. As shown in **Fig. 13**, we let an RNN receive the six values described above for 72 hours as time-series data of the input and then we predicted those values as values measured on the 73rd hour as the output.

We run multi-fold cross validation using data from 04/01/2004 to 03/31/2005 of the Air Quality dataset. As shown in Fig. 13, it is divided into four blocks consisting of three months. Here, one block among them is used as the test data whereas the remaining three blocks are used as the training data D_c for the original model. It is also assumed that the adversary possesses data $D_a \subseteq D_c$ for n months, e.g., the one block per three months and the two blocks per six months. There are four ways to select the test data D_c as shown in **Table 3**, and furthermore there are six ways to choose D_a for each D_c. In other words, we conduct twenty-four experiments to evaluate the original model and the substitute model.

Table 3 Combination of test data and training data owned by adversary. The B*i* shows the Block*i* where *i* is 1, 2, 3, 4.

test data	original model	adversary	distance
B1	B2, 3, 4	B2 (3 months)	1
		B3 (3 months)	2
		B4 (3 months)	3
		B2, 3 (6 months)	1.5
		B3, 4 (6 months)	2.5
		B4, 2 (6 months)	2
B2	B1, 3, 4	B1 (3 months)	1
		B3 (3 months)	1
		B4 (3 months)	2
		B1, 3 (6 months)	1
		B3, 4 (6 months)	1.5
		B4, 1 (6 months)	1.5
B3	B1, 2, 4	B1 (3 months)	2
		B2 (3 months)	1
		B4 (3 months)	1
		B1, 2 (6 months)	1.5
		B2, 4 (6 months)	1
		B4, 1 (6 months)	1.5
B4	B1, 2, 3	B1 (3 months)	3
		B2 (3 months)	2
		B3 (3 months)	1
		B1, 2 (6 months)	2.5
		B2, 3 (6 months)	1.5
		B3, 1 (6 months)	2

Since the Air Quality dataset has a general context, an adversary who owns a block of data away from the one selected as the test data may not have any advantage. Accordingly, the experimental results are grouped by the distance between D_a and the size of the block related to the test data. The distance is also shown in Table 3. For example, when the test data is B1 and the adversary has B2 as D_a, the distance is 1. Likewise, if the adversary has B3 and B4 as D_a, the distance is 2.5 by taking the average of both.

[*10] https://archive.ics.uci.edu/ml/datasets/air+quality
[*11] https://www.atmarkit.co.jp/ait/articles/1804/26/news150.html

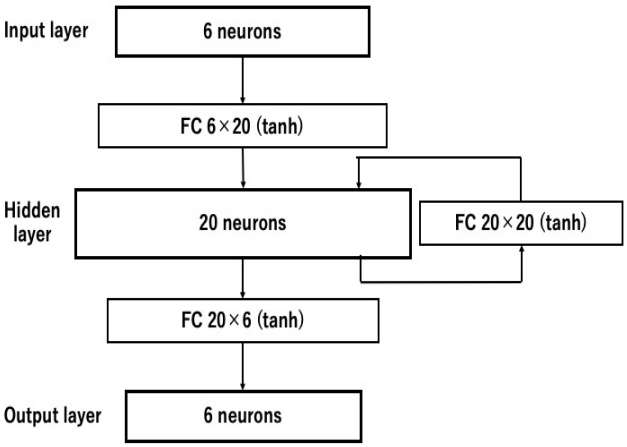

Fig. 14 The architecture of the neural network used on the experiment in regression task.

As described above, the softmax function with temperature cannot be used in the regression task. Therefore, the L_{reg} described in Section 3.2.2 is used instead. Note again that L_{reg} is used in the distillation of neural networks as well as the softmax function with temperature and thus it is expected to prove effective in model extraction attacks. A substitute model is trained 10,000 times using $L_2 loss$ of the predicted values and labeled data as D_a, and then it is trained 10,000 times again by using L_{reg} with the predicted values via querying to the original model. The batch size is 16.

The coefficient R^2 of determination, which is commonly used for regression analysis, is utilized in the evaluation of the accuracy. The coefficient R^2 is defined as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - pre)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}, \tag{5}$$

where y is the labeled data, $\overline{y}$ is the average of y, and pre are the predictions from the original or substitute model. The coefficient R^2 is preferably very close to 1. According to Rubin [24], if R^2 exceeds 0.8, the correlation is very strong and, for example, $R^2 = 0.8$ means that 80% of the variation in the dependent variable has been explained. Based on our pre-experiment, R^2 of the original model is 0.8992 although we omit the details. In this experiment, we refer to the value of R^2 as the accuracy, and an R^2 of 0.8992 is the attack goal of the substitute model for an adversary.

Figure 14 shows the architecture of the neural network used on the experiment in regression task. We use tf.nn.rnn_cell. BasicRNNCell and tf.nn.rnn_cell.BasicLSTMCell as hidden layers and use tf.nn.dynamic_rnn for network input/output.

Baseline of the Experiment on the Regression Task

The following setting is utilized as a baseline to compare the performance of our attack.

- True label only: This is identical to the least square error where the distillation is not used.

4.1.3 Evaluation Terms in Experiments

An evaluation of the experiments is conducted as described below.

4.1.3.1 Classification Task

The evaluation terms in the experiments for the classification task are as follows:

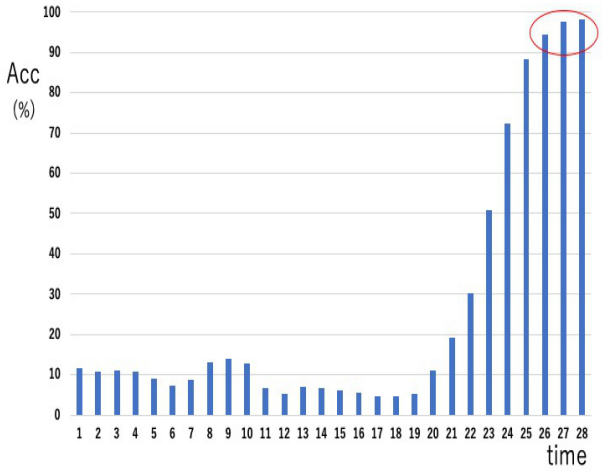

Fig. 15 Results for identification of leaky time on LSTM.

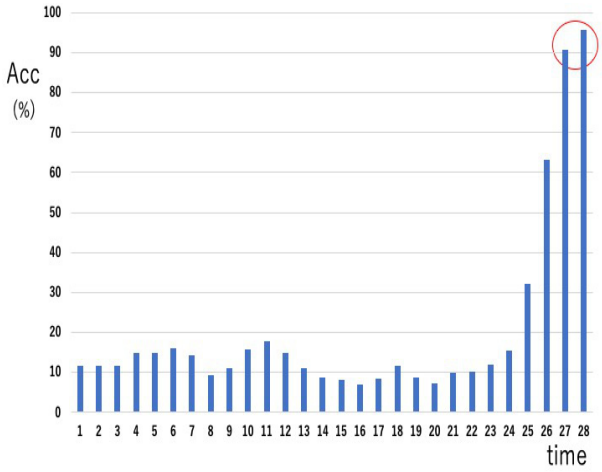

Fig. 16 Results for identification of leaky time on RNN.

- Leaky time.
- Difference in accuracy depending on each architecture of the substitute model.
- Difference in accuracy depending on the number of training data owned by an adversary, i.e., D_a.
- Difference in accuracy depending on temperature T of the softmax function with temperature.

4.1.3.2 Regression Task

The evaluation terms in the experiments for the regression task are as follows:

- Difference in R^2 depending on each architecture of the substitute model.
- Difference in R^2 depending on the ratio of training data owned by an adversary, i.e., D_a.
- Effects of loss functions using L_b.

4.2 Experimental Result

4.2.1 Results of Classification Task

First, the results for identification of leaky time are shown in **Figs. 15** and **16**.

According to Figs. 15, 16, increments of accuracy for LSTM started from $t = 21$ and reached more than 90% after $t = 26$. On the other hand, in an additional experiment, whereby RNN is

utilized in the original model instead of LSTM, increments of accuracy for RNN started from $t = 25$ and reached more than 90% after $t = 27$. Therefore, the accuracy of the final results of LSTM is greater than that of RNN, and LSTM is potentially leakier even at an early time.

Next, the results for intensive extraction of LSTM at $t = 21$ and later times are shown in **Fig. 17**. According to Fig. 17, a model with 95% accuracy is extracted at $t = 21$. Notably, for $T = 16$ in the softmax function with temperature, the accuracy reaches 97.5%, which is higher than the 97.3% accuracy of the original model. Note that, in the case in which the original model is a many-to-one type of LSTM and only the final prediction result at $t = 28$ is returned, an adversary will only know the final result and cannot extract with 97.5% accuracy for the substitute model. Furthermore, the accuracy in the case of the baseline, i.e., with the temperature $T = 1$ and the use of only the final output at time 28, is lower than 93%. This fact is evidence of the effectiveness of our attack.

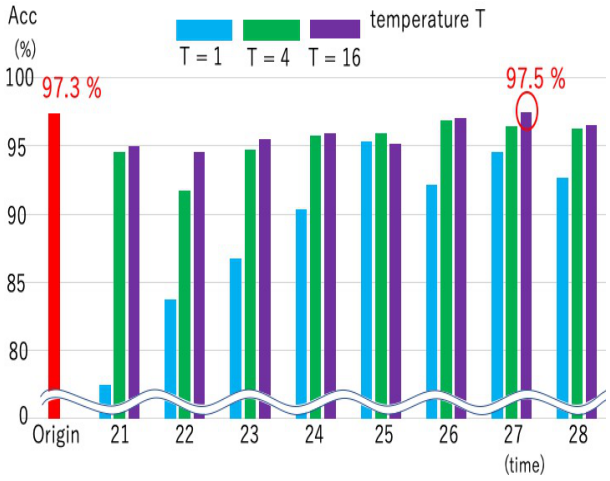

Fig. 17 Intensive extraction during leaky times.

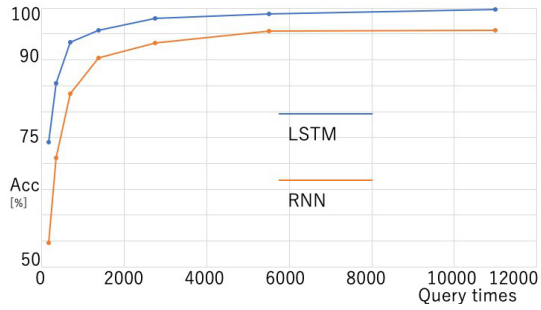

Fig. 18 Accuracy of substitute model depending on the number of queries.

Finally, we investigate how the accuracy of the substitute model is related to the number of queries at time $t = 27$ and temperature $T = 16$. The result is shown in **Fig. 18**. As a result, accuracy of the substitute model is rapidly deteriorates when the amount of training data D_a owned by an adversary decreases.

4.2.2 Results of the Regression Task

The results on the Air Quality dataset are shown in **Fig. 19**. We also conduct an experiment where an adversary owns an LSTM as a substitute model and the results are shown in **Fig. 20**. When the parameter m is 10, $\|R_s - y\|_2^2$ in Eq. (2) is selected as L_b almost every time. Therefore, m no longer has an effect in the experimental results, and thus we adopt $m \leq 10$.

According to Figs. 19, 20, in comparison with a substitute model trained with only labeled data without querying the original model, a substitute model with L_{reg} presented in Eq. (4) provides a large value for R^2 by leveraging the predicted values from the original model. Thus, the coefficient of R^2 on utilizing L_{reg} is higher than that in the case of the baseline, which uses only the true label. This fact is evidence of the effectiveness of our attack.

Furthermore, we show the results of the coefficient of R^2 for each range in accordance with the baseline. The case of RNNs is shown in **Fig. 21** and the case of LSTMs is shown in **Fig. 22**. These results indicate that when the better performance is obtained only by the use of true labels, the benefit of our attack is relatively lost. In other words, an adversary is more likely to benefit from our attack when having unfavorable data or namely maintaining a poor accuracy with only the true labels.

Several substitute models whose coefficient R^2's are larger than those of their original models are obtained. For instance, according to Table 3, when an adversary has a dataset such that for a hyperparameter $m = 0.5$ in Fig. 19, the coefficient R^2 becomes larger when the size is six months and the distance from the test data is 1.

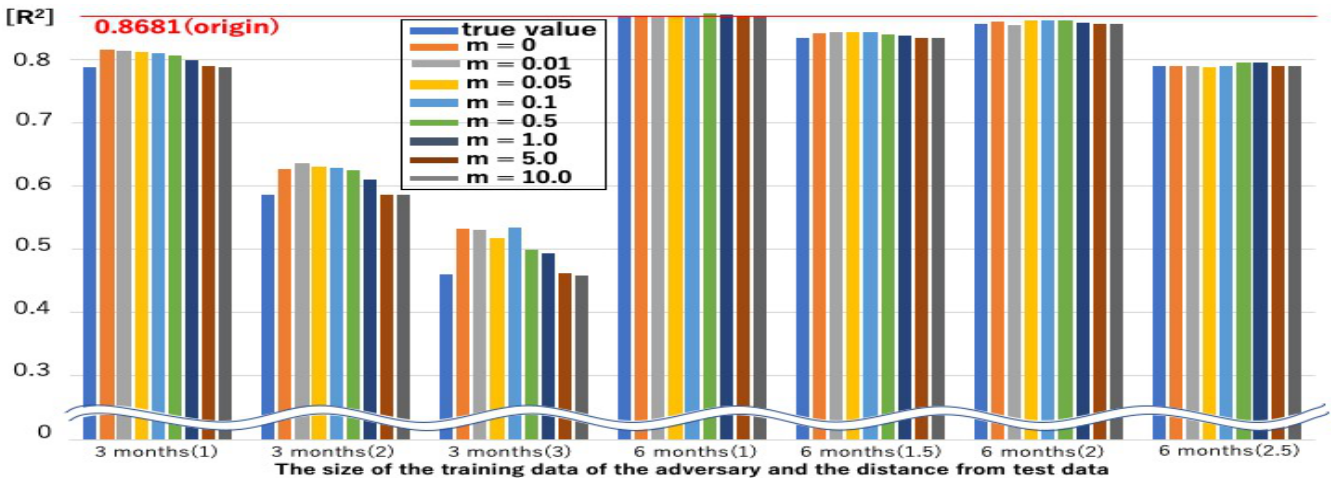

Fig. 19 Accuracy of the substitute model with Air Quality dataset.

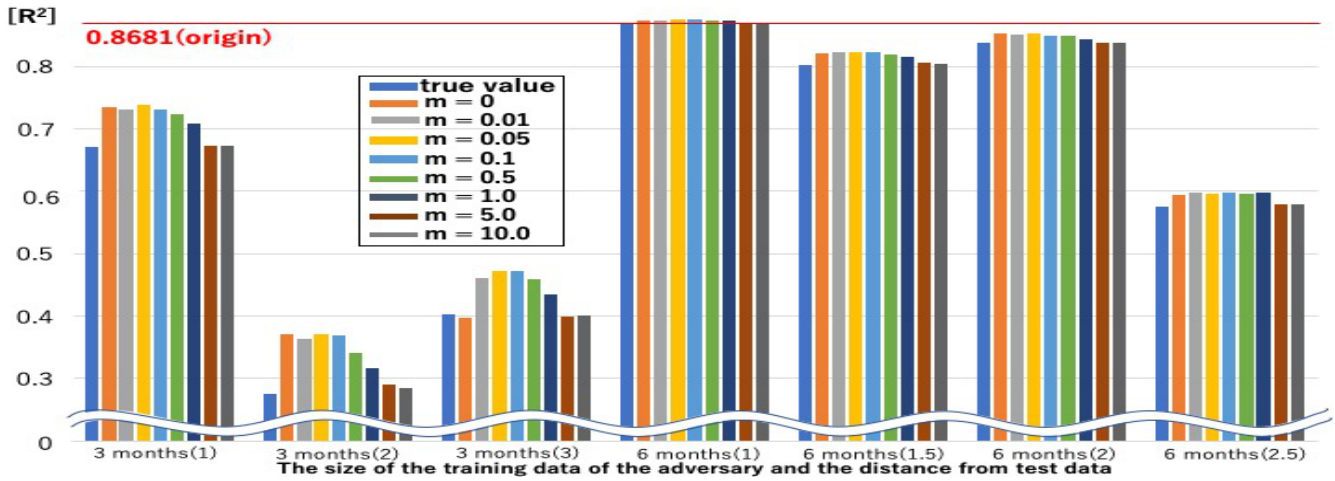

Fig. 20 Accuracy in Air Quality dataset when adversary owns LSTM.

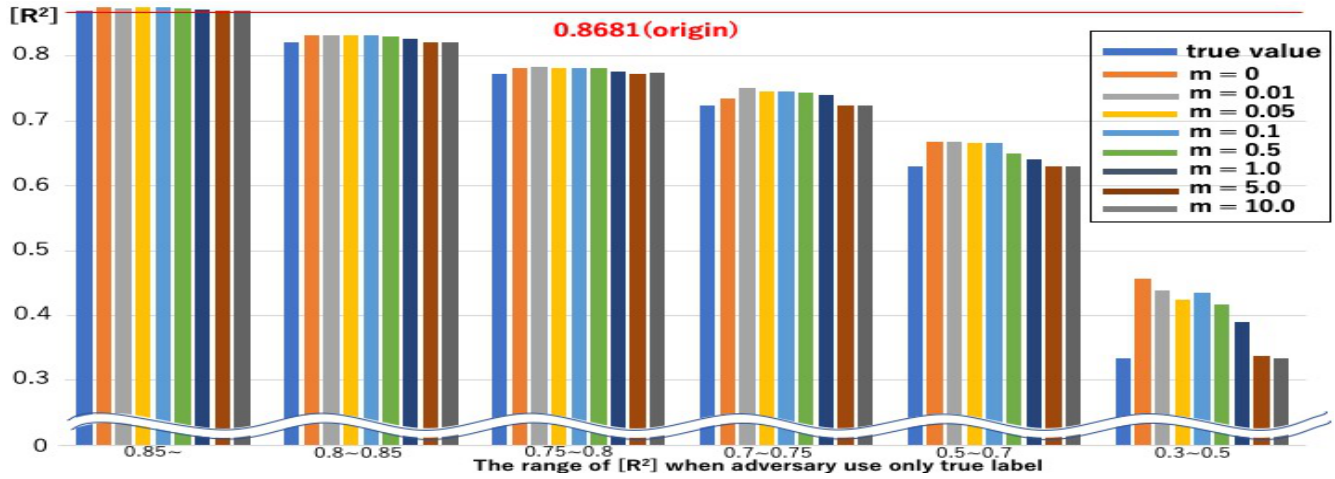

Fig. 21 Accuracy of substitute model with Air Quality dataset for each condition according to performance when an adversary uses true label only.

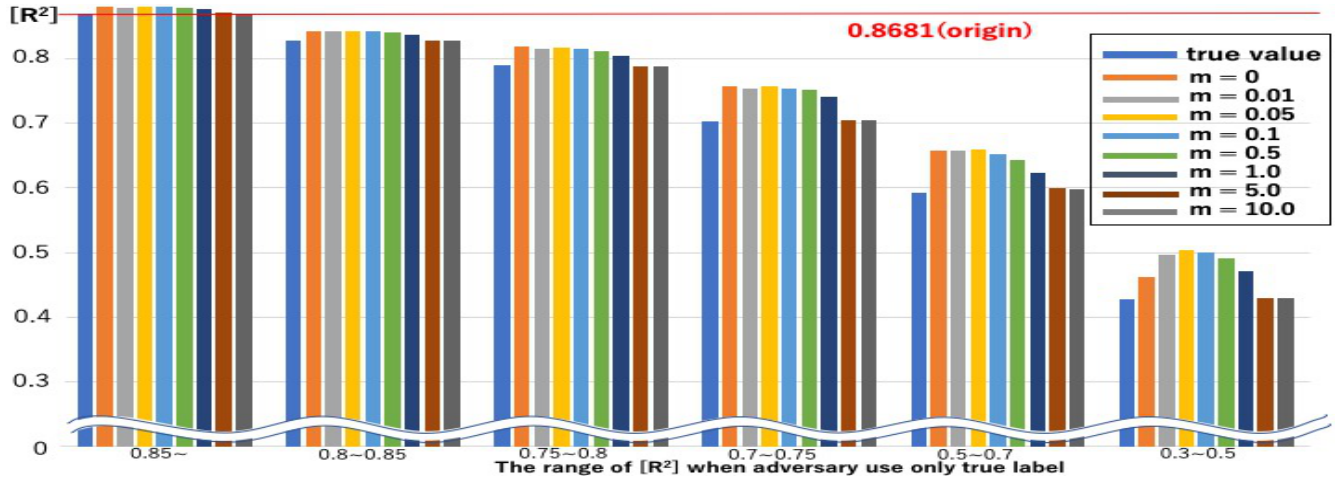

Fig. 22 Accuracy in Air Quality dataset for each condition according to performance when an adversary uses true label only and owns LSTM.

5. Considerations

In this section, we discuss the difference in behavior of model extraction attacks in accordance with the distinction of architectures on a classification task to determine how simple RNNs and complex LSTMs affect the model extraction. Moreover, we consider the loss function used in a regression task to clarify how the designed loss function works in the model extraction, and then show future prospects. Finally, we discuss countermeasures against model extraction attacks.

5.1 Classification Task

We discuss the impact of structural architectural differences in the original model. LSTM can stochastically control information not only for the near past but also for an earlier time as feedback, and thus the range of leaky times can be potentially expanded. Likewise, a substitute model with a high accuracy at $t = 21$ was extracted because the bottom lines of the samples on the MNIST dataset contain less information. Thus, the substitute model was able to learn sufficient knowledge from the original model at the procedure on the 21st line. In our experiment, we converted the MNIST dataset to time-series data to identify the range of leaky times. This implies the existence of other time-series datasets where an adversary can extract a model even at earlier times, such as at the beginning time of a model extraction attack. In such a case, extraction becomes easier and thus more stringent restrictions on the use of APIs are necessary.

Moreover, as an additional experiment, we used LSTM as a substitute model. In comparison with the 90.3% accuracy obtained by RNN as a substitute model with 1,375 training data, the same accuracy is obtained by LSTM as a substitute model with only 500 training data. Furthermore, in the case of LSTM as a substitute model, the model with 99.69% accuracy can be extracted when 11,000 training data is used in the substitute model. Thus, in RNNs, an adversary using LSTM can extract a substitute model with a higher accuracy even with fewer queries than the use of RNN.

Okada et al. [21] utilized DNN and CNN as substitute models with the MNIST dataset, respectively, in their experiments. In doing so, they showed that models with more than 90% accuracy can be extracted even with 684 training data on DNN and 171 training data on CNN for the substitute models. In other words, according to Okada et al., the use of CNN is more effective for an adversary to extract a model with a high accuracy in an image classification task. Likewise, Krishna et al. [16] showed that an adversary can obtain a higher accuracy when a more complicated architecture of BERT [6] is used. In contrast, we show that in the case of time-series data, the use of LSTM as a substitute model enables an adversary to extract a model with a higher accuracy even with fewer queries than during use of RNN. Our results are identical to those of Okada et al. and Krishna et al., except for differences in the architectures among RNN, CNN, and BERT.

Finally, although a substitute model in our experiment was trained by utilizing labeled data and soft labels separately, Chen et al. [5] defined a loss function using both at the same time. The use of this loss function can reduce the number of epochs in training. In particular, the loss function is defined in Eq. (6) below, where P_s is the predicted value of a substitute model, P_t is a predicted value of an original model, μ is a hyperparameter, L_{hard} is the value of a loss function obtained by labeled data, and L_{soft} is the value of a loss function obtained by soft labels.

$$L = \mu L_{hard}(P_s, y) + (1 - \mu)L_{soft}(P_s, P_t). \quad (6)$$

The convergence of a substitute model with high accuracy becomes more effective by setting an appropriate value for μ. We leave the determination of an appropriate value for μ as an open problem.

5.2 Regression Task

Next we consider the loss function L_b. We discuss the reason why performance of a substitute model is independent of a parameter m when the model extraction attacks are performed with high performance, e.g., R^2 exceeds 0.85. The parameter m affects the case where a non-zero value is selected for L_b, or namely the frequency of training such that weights are significantly updated. Since training for the substitute model by an adversary was sufficiently converged in our experiment given a sufficient number of epochs, the case described above did not become a problem.

Next, we consider the reason why the loss function L_{reg} defined in Eq. (4) is ineffective in the case where training for the substitute model with labeled data is unsuccessful, e.g., R^2 is lower than 0.3. In that case, at the phase of training with the predict values from an original model, a value of R_s is expected to be close enough to that of R_t. Consequently, if R_s is mismatched and underfitting at the phase of L_b in Eq. (2), the inequality cannot be evaluated as expected. Thus, its resulting loss function L_{reg} will become unworkable.

Finally, if an adversary who owns data from April to June can produce valid data from July to September, then it has the same ability as an adversary who originally owns data from April to September. This is possible for at least time-series data such as the Air Quality dataset in which an output value at a certain time is identical to an input at a later time. For instance, if an adversary who owns data from April to June makes queries from 0:00 on June 28th to 23:00 on June 30th, then it may be able to precisely predict data about 0:00 on July 1st, which was unknown to the adversary. By iterating such an operation recursively, the adversary can obtain a larger and pseudo dataset. The intuition of the methodology is shown in **Fig. 23**. We plan to verify the validity of the methodology in a future work by utilizing features of input/output for RNNs to handle time-series data.

5.3 Countermeasures

As countermeasures to model extraction attacks, Kesarwani et al. [15] proposed extraction warning, wherein a model trained by an adversary is emulated as another model by a proxy and extraction will be alerted if the emulated model achieves some threshold designated in advance. Although the extraction warning can potentially be useful, settings about thresholds have never been discussed. In addition, although there is a method to detect model extraction attacks [14] whereby a cloud server checks if the distribution of API queries deviates from general and legitimate users, such an approach is ineffective against collusion between adversaries who execute the attacks according to Kesarwani et al. [15]. Moreover, according to Atli et al. [10], approaches to monitoring and alerting behavior of users are ineffective against model extraction attacks on complex neural networks.

Szyller et al. [26] proposed an approach based on digital watermarking [4], [31] to claim cloud ownership after a model extraction attack is made. Such an approach is expected to detect model extraction attacks by verifying the watermarking in a substitute model when an adversary publishes the model. However, the model extraction attacks based on distillation shown in this work (and the work of Okada et al. [21]) enable an adversary to remove

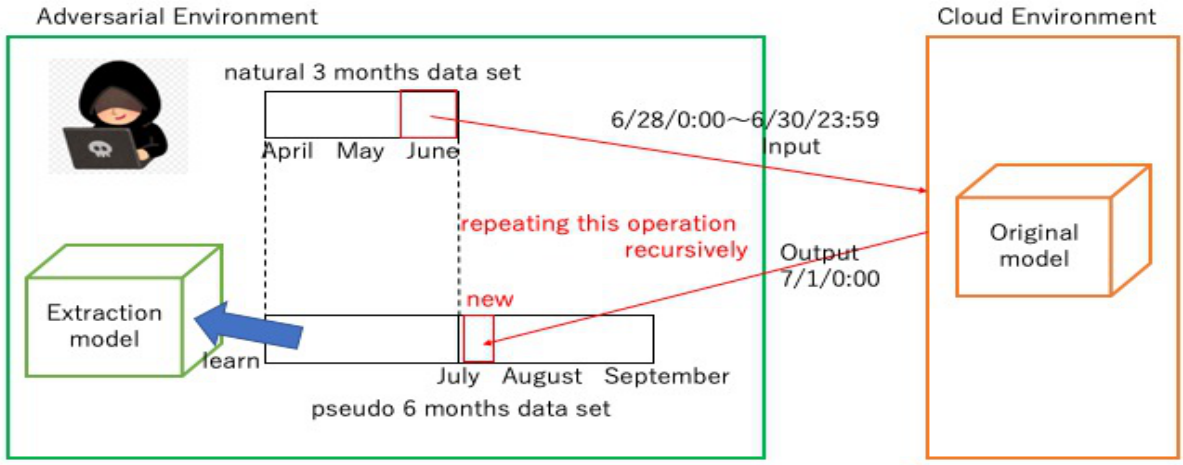

Fig. 23 Idea for improvement of adversary's capability.

watermarking. Moreover, according to Krishna et al. [16], watermarking can only verify whether an original model has been stolen through a substitute model and cannot prevent the extraction itself. Therefore, extraction can go unnoticed if an adversary keeps its own substitute model private, rendering the digital watermarking useless.

Another countermeasure is the use of differential privacy [7] as proposed by Huadi et al. [32]. However, such an approach will lower the performance of an original model hosted by a cloud server. Consequently, to the best of our knowledge, providing practical and effective countermeasures remains an open problem.

Finally, we discuss an alternative way to mitigate model extraction attacks, especially against the attacks discussed in this paper. The use of the softmax function with temperature [11] was the key idea regardless of the architecture from the viewpoint of model extraction attacks for a classification task. In doing so, soft-labels provide the probability for each label as a prediction result and are given to the softmax function with temperature as input. Since accuracy heavily depends on the maximized value in the prediction result, we consider manipulating the effects by the second maximized value and the lower values to mitigate the softmax function with temperature. This can prevent an adversary from obtaining more information than prediction results alone by disturbing an output without affecting a precise prediction result.

5.4 Limitations

We describe three limitations on this work below.

The first limitation is the assumption about knowledge of an adversary. In the current discussion, an adversary needs to have knowledge of a dataset utilized in the training of an original dataset. Namely, we have not solved whether our attacks are applicable to a completely private dataset or not. For instance, data distributions based on private photos such as the CIFAR10 dataset are potentially wider than that of the MNIST dataset, which was utilized in the current experiments. In other words, in the case of a dataset whose distribution is completely private, an adversary may no longer own a part of a dataset utilized in an original dataset and thus threats in such a situation are an open problem. We have also considered that such an evaluation is possible via the approach of the knockoff nets [22]. Further studies, which take the knockoff nets setting into account, will need to be undertaken.

The second limitation is the image classification task with RNNs. As described in Section 3.2.1, image classification with RNNs is often based on a combination of the RNNs with CNNs [29]. To understand threats of model extraction attacks for an image classification task with RNNs in the real world, the model extraction with a combination of RNNs with other neural networks, e.g., DNNs and CNNs, should be discussed. These topics are reserved for another future work.

The final limitation is that the current experiments are conducted on only the MNIST and Air Quality datasets. To further clarify the generalization ability of the model extraction on RNN, additional experiments with more datasets, e.g., CIFAR10, should be conducted. We hope that further tests will confirm our findings.

6. Conclusion

Model extraction attacks enable an adversary to extract a machine learning model via prediction queries to a model. In this paper, we discussed model extraction attacks based on features of recurrent neural networks (RNNs). In the case of the classification task, we were able to extract a substitute model without the final output from an original model by utilizing outputs halfway through the sequence. In the case of the regression task, we presented a new attack by newly configuring a loss function.

In the classification task, we conducted experiments by converting the MNIST dataset to time-series data. Our experimental results show that a substitute model can be effectively extracted by utilizing prediction results from the original model after identifying the leaky time via training with true labels. In particular, by utilizing the softmax function with temperature [11] in prediction results from the original model, a substitute model with a higher prediction accuracy than the original model could be extracted.

In the regression task, we proposed a new extension of model extraction attacks by using the teacher-bounded regression loss function [5] as a loss function. In experiments with the Air Qual-

ity dataset for our attack, we showed a substitute model whose correlation is strongly similar to an original model even when the substitute model was trained with only a small amount of data.

We also considered relationships between the accuracy and complicated architectures for a substitute model. We conclude that the use of a complex architecture contributes to obtaining a higher accuracy for a substitute model. These results corroborate the findings of Okada and Hasegawa [21] and Krishna et al. [16].

To further our research, we intend to conduct experiments under the knockoff nets setting [22] as described in Section 5.4. Future work will also look into experiments with more detasets such as the CIFAR10 as well. We are now investigating model extraction on a combination of RNNs and CNNs for an image classification task in the real world as well. Finally, we also plan to discuss the threshold of restrictions on the use of APIs as a countermeasure to model extraction attacks.

Acknowledgments This work is supported by the Cabinet Office (CAO), Cross-ministerial Strategic Innovation Promotion Program (SIP), Cyber Physical Security for IoT Society (funding agency: NEDO). We would also like to thank Jason Paul Cruz for helpful comments.

References

[1] Ba, J. and Caruana, R.: Do Deep Nets Really Need to be Deep?, *Proc. NIPS*, pp.2654–2662, Curran Associates, Inc. (2014).
[2] Batina, L., Bhasin, S., Jap, D. and Picek, S.: CSI Neural Network: Using Side-channels to Recover Your Artificial Neural Network Information, *Proc. USENIX Security*, pp.515–532, USENIX Association (2018).
[3] Bucilâ, C., Caruana, R. and Niculescu-Mizil, A.: Model Compression, *Proc. KDD*, pp.535–541, ACM (2006).
[4] Carlini, N., Liu, C., Erlingsson, Ú., Kos, J. and Song, D.: The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks, *Proc. USENIX Security*, pp.267–284, USENIX Association (2019).
[5] Chen, G., Choi, W., Yu, X., Han, T. and Chandraker, M.: Learning Efficient Object Detection Models with Knowledge Distillation, *Proc. NIPS*, pp.742–751, Curran Associates, Inc. (2017).
[6] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding (2019) (online), available from ⟨https://arxiv.org/abs/1810.04805⟩.
[7] Dwork, C.: Differential Privacy, *Proc. ICALP*, Lecture Notes in Computer Science, Vol.4052, pp.1–12, Springer (2006).
[8] Ghosh, A., Bose, S., Maji, G., Debnath, C.N. and Sen, S.: Stock Market Prediction Using LSTMs, *Proc. CAINE 2019, EPiC Series in Computing*, Vol.63, pp.101–110, EasyChair (2019).
[9] Goodfellow, I., Bengio, Y. and Courville, A.: *Deep learning*, MIT press (2016).
[10] Atli, B.G., Szyller, S., Juuti, M., Marchal, S. and Asokan, N.: Extraction of Complex DNN Models: Real Threat or Boogeyman? (2019) (online), available from ⟨https://arxiv.org/abs/1910.05429⟩.
[11] Hinton, G., Vinyals, O. and Dean, J.: Distilling the knowledge in a neural network (2015) (online), available from ⟨https://arxiv.org/abs/1503.02531⟩.
[12] Iskandaryan, D., Ramos, F. and Trilles, S.: Air Quality Prediction in Smart Cities Using Machine Learning Technologies Based on Sensor Data: A Review, *Applied Sciences*, Vol.10, No.7, p.2401 (2020).
[13] Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A. and Papernot, N.: High-Fidelity Extraction of Neural Network Models (2019) (online), available from ⟨https://arxiv.org/abs/1909.01838⟩.
[14] Juuti, M., Szyller, S., Marchal, S. and Asokan, N.: PRADA: Protecting against DNN Model Stealing Attacks, *Proc. EuroS&P*, pp.512–527, IEEE (2019).
[15] Kesarwani, M., Mukhoty, B., Arya, V. and Mehta, S.: Model Extraction Warning in MLaaS Paradigm, *Proc. ACSAC*, pp.371–380, ACM (2018).
[16] Krishna, K., Gaurav, S.T., Parikh, P.A., Papernot, N. and Iyyerm, M.: Thieves on Sesame Street! Model Extraction of BERT-based APIs (2019) (online), available from ⟨https://arxiv.org/abs/1910.12366⟩.
[17] Li, L. and Ngan, C.-K.: A Weight-adjusting Approach on an Ensemble of Classifiers for Time Series Forecasting, *Proc. ICISDM 2019*, pp.65–69, ACM (2019).
[18] Munkhdalai, L., Munkhdalai, T., Park, K.H., Amarbayasgalan, T., Erdenebaatar, E., Park, H.W. and Ryu, K.H.: An end-to-end adaptive input selection with dynamic weights for forecasting multivariate time series, *IEEE Access*, Vol.7, pp.99099–99114 (2019).
[19] Naghibijouybari, H., Neupane, A., Qian, Z. and Abu-Ghazaleh, N.: Rendered Insecure: GPU Side Channel Attacks are Practical, *Proc. CCS*, pp.2139–2153, ACM (2018).
[20] Reith, R.N., Schneider, T. and Tkachenko, O.: Efficiently Stealing your Machine Learning Models, *Proc. WPES*, pp.198–210, ACM (2019).
[21] Okada, R. and Hasegawa, S.: Gazoubunruishinsogakushukinitaisuru Model Extraction kogekinokensho, *Proc. CSS* (in Japanese), pp.201–208 (2018).
[22] Orekondy, T., Schiele, B. and Fritz, M.: Knockoff Nets: Stealing Functionality of Black-Box Models, *Proc. CVPR*, pp.4954–4963, IEEE (2019).
[23] Pal, S., Gupta, Y., Shukla, A., Kanade, A., Shevade, S. and Ganapathy, V.: A framework for the extraction of Deep Neural Networks by leveraging public data (2019) (online), available from ⟨https://arxiv.org/abs/1905.09165⟩.
[24] Rubin, A.: *Statistics for Evidence Based Practice and Evaluation*, chapter 13, Brooks/Cole Pub Co. (2012).
[25] Ŝrndić, N. and Laskov, P.: Practical Evasion of a Learning-Based Classifier: A Case Study, *Proc. IEEE S&P*, pp.197–211, IEEE (2014).
[26] Szyller, S., Gul Atli, B., Marchal, S. and Asokan, N.: *DAWN: Dynamic Adversarial Watermarking of Neural Networks* (2019) (online), available from ⟨https://arxiv.org/pdf/1906.00830.pdf⟩.
[27] Tramér, F., Zhang, F. and Juels, A.: Stealing Machine Learning Models via Prediction APIs, *Proc. USENIX Security*, pp.601–618, USENIX Association (2016).
[28] Wang, B. and Zhenqiang Gong, N.: Stealing Hyperparameters in Machine Learning, *Proc. IEEE S&P*, pp.36–52, IEEE (2018).
[29] Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C. and Xu, W.: CNN-RNN: A Unified Framework for Multi-Label Image Classification, *Proc. CVPR* (2016).
[30] Yoshida, K., Kubota, T., Shiozaki, M. and Fujino, T.: Model-Extraction Attack against FPGA-DNN Accelerator Utilizing Correlation Electromagnetic Analysis, *Proc. FCCM*, pp.318–318, IEEE (2019).
[31] Zhang, C., Bengio, S., Hardt, M., Recht, B. and Vinyals, O.: Understanding deep learning requires rethinking generalization (2016) (online), available from ⟨http://arxiv.org/abs/1611.03530⟩.
[32] Zheng, H., Ye, Q., Hu, H., Fang, C. and Shi, J.: BDPL: A Boundary Differentially Private Layer Against Machine Learning Model Extraction Attacks, *Proc. ESORICS*, Lecture Notes in Computer Science, Vol.11735, pp.66–83, Springer (2019).

Editor's Recommendation

This paper proposes a model extraction attack on a network (RNN) and shows that the attack can be successfully carried out with relatively high accuracy by conducting experiments using the MNIST dataset and other data sets. Since the boundary field between deep learning and security is not still very cobbled together, and the threats to the RNNs have not yet been sorted out, this paper's achievement is very interesting.

(Chief examiner of SIGSCEC, Toshihiro Yamauchi)

Tatsuya Takemura received his B. Eng. degree in Engineering Science from Osaka University, Japan, in 2019. He has recently joined the M.S. course in the Graduate School of Information Science and Technology in Osaka University, Japan. His research interests include machine learning and information security.

Naoto Yanai received his B. Eng. degree from The National Institution of Academic Degrees and University Evaluation, Japan, in 2009, the M.S. Eng. from the Graduate School of Systems and Information Engineering, University of Tsukuba, Japan, in 2011, and his Dr. E. degree from the Graduate School of Systems and Information Engineering, the University of Tsukuba, Japan, in 2014. He is an assistant professor at Osaka University, Japan. His research interests include cryptography and information security.

Toru Fujiwara received his B.E., M.E., and Ph.D. degrees in Information and Computer Sciences from Osaka University in 1981, 1983, and 1986, respectively. In 1986, he joined the faculty of Osaka University. Since 1997, he has been a Professor at Osaka University. He is currently with the Graduate School of Information Science and Technology. His current research interests include coding theory and information security.