Computers & Security 153 (2025) 104401

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Full Length Article

Data reduction for black-box adversarial attacks against deep neural networks based on side-channel attacks

Hanxun Zhou [a], Zhihui Liu [a], Yufeng Hu [a], Shuo Zhang [a], Longyu Kang [a], Yong Feng [a,*], Yan Wang [a,*], Wei Guo [b,*], Cliff C. Zou [c]

[a] Liaoning University, Shenyang, PR China
[b] Shenyang Aerospace University, Shenyang, PR China
[c] University of Central Florida, Orlando, FL, USA

ARTICLE INFO

Keywords:
Deep learning
Side-channel attack
Data reduction
Adversarial attack

ABSTRACT

Launching effective black-box adversarial attack against a deep neural network (DNN) without knowledge of the model's details is challenging. Previous studies involved performing numerous queries on the target model to generate adversarial examples, which is unacceptable due to the high query volume. Additionally, many of these queries are unnecessary as the dataset may contain redundant or duplicate data. To address these issues, we propose a two-stage black-box adversarial attack approach that combines side-channel attacks and a data reduction technique. In the first stage, we employ Long Short Term Memory (LSTM) to gather partial information about the target DNN through side-channel attacks, enabling us to obtain the class probability of the dataset. In the second stage, we utilize a new data reduction algorithm based on the class probability to enhance the efficiency of generating adversarial examples. Our approach is capable of precisely identifying the target model and the data reduction performs better than other reduction methods. Furthermore, when utilizing the reduced datasets to train the shadow model, the adversarial examples generated on this shadow model demonstrate a higher transferability success rate than SOTA data reduction methods.

1. Introduction

In recent years, researchers have paid more and more attention on deep learning neural network (DNN), which is widely used in various fields, such as image classification (Azizi et al., 2021), natural language processing (Wang et al., 2021), and autonomous driving (Li et al., 2022), etc. However, it takes tremendous resources to develop the above-mentioned commercial model. As a result, the security of the models becomes an important issue that cannot be ignored.

DNN are vulnerable to several types of attacks, especially adversarial attacks (Coqueret et al., 2023; Gupta and Drees, 2023; Acharya et al., 2022). In addition, there are other types of attacks including model extraction attacks (Rakin et al., 2022b), member inference attacks (Tang et al., 2022), model inversion attacks (Khowaja et al., 2022), and hyper-parameter stealing attacks (Rakin et al., 2022a), etc.

The mainstream adversarial attacks include black-box attacks and white-box attacks. Attackers have no prior knowledge of the neural network model in black-box attacks, while attackers have access to the complete model in white-box attacks. Black-box attacks, such as transfer attacks (Qi et al., 2020) and optimization attacks (Fnu et al., 2020), require querying the target model as a precondition, and then training the shadow model with similar decision boundaries to generate adversarial examples (AEs). However, the large number of queries to the target model is often extremely costly or even infeasible. Therefore, it is of great importance to reduce the queries to the target model (He et al., 2021) or even not to query the target model.

With the exploration of side-channel attack methodology, attackers can infer the network structure by exploiting memory and timing side channels (Hua et al., 2018). Researchers have successfully deciphered deep learning model by power consumption, computation time, and electromagnetic radiation (Xiang et al., 2020; Aldahdooh et al., 2022; Luo et al., 2022; Bhagoji et al., 2017), etc. As for deep learning model running on computers, most of the resources are consumed by GPU, so an attacker can also reverse the structure of a DNN model by collecting the resource consumption of GPU (Bhagoji et al., 2017). Most side-channel attacks use oscilloscopes or external data acquisition cards

* Corresponding authors.
E-mail addresses: fylnu9@163.com (Y. Feng), xiaotaiyang0567@163.com (Y. Wang), 18790329669@163.com (W. Guo).

https://doi.org/10.1016/j.cose.2025.104401
Received 22 March 2024; Received in revised form 24 November 2024; Accepted 24 February 2025
Available online 25 February 2025

to collect side channel signals. We believe that it is not feasible for an attacker to use an oscilloscope-like device to measure the signal directly, and a feasible method is to measure the signal by a Trojan script.

Since we do not have access to the complete structure of the target model, we construct multiple alternative models with similar structures. These models are then used to average the class probabilities obtained in each model, which enhances generalization ability. The alternative models are constructed successively, and the samples are ranked based on their class probabilities. The newly obtained probabilities are averaged with those calculated in previous alternative models, and the process is stopped when the difference between the class probabilities of two successive models is below a given threshold. Finally, we use the reduced dataset to train the shadow model, which is the last alternative model (**LAM**).

We make the following contributions in this paper:

We develop and design a new two-stage adversarial attack algorithm, called Data Reduction for Adversarial Attacks based on Side-Channel Attacks (**DRAASC**), which enables an attacker to launch a black-box attack.

We design a dataset reduction algorithm to simplify the attack process and generate adversarial samples based on the dataset to launch a black-box attack. The results show that our method outperforms other methods in terms of transferability.

2. Related work

The development of side channel technology poses a great challenge to the security of DNN model, and the architecture of DNNs were cracked by many researchers. Papernot et al. (2017) found that a side-channel attack can obtain the general structure of the DNN model, such as the activation function, the number of network layers, the number of output classes, and so on; Wei et al. (2018) proposed that the input image can be obtained by analyzing the power trajectory in the first convolutional layer. Yan et al. (2020) presented Cache Telepathy a fast and accurate mechanism to steal DNN s architecture by the cache side channel. There are also attacks that extract the architecture of deep neural networks (DNN), which enhances an adversary s capability to conduct black-box attacks against the model (Hong et al., 2018). Liu and Srivastava (2020) proposed GANRED, an attack approach based on the generative adversarial network (GAN) which utilizes cache timing side-channel to recover the structure of DNNs without memory sharing or code access. Luo et al. (2022) utilized electromagnetic (EM) side-channel leakage to learn the association between DNN architecture configurations and EM emanations comprehensively. Compared with previous work, we use LSTM to simplify the work to identify the model by conventional side channel techniques.

The training process of deep learning requires a large amount of data. However, collecting and labeling the data costs much time as well as resources, and the large amount of data also increases the probability of poisoning attacks on the dataset. Therefore, simplifying the data with good quality is also of great interest to researchers. Chitta et al. (2019) used AL to build subsets from large labeled training datasets to provide accurate DNNs in less training time. Kholidy and Erradi (2019) introduced VHDRA, a Vertical and Horizontal Data Reduction Approach, to improve the classification accuracy and speed of the NNGE algorithm and reduce the computational resource consumption. Tertytchny and Michael (2020) proposed a machine learning-based framework for instance-based dataset reduction applied for IFD model. He et al. (2021) proposed to use mutual information to measure the quality of the dataset to do data reduction. They obtain the confidence of the dataset based on the obtained model without querying the target model. We believe that the confidence of the data on the model reflects the sensitivity of the model to the dataset, and design the dataset reduction algorithm based on the confidence.

In this article, we focus on adversarial attacks. White-box access to model is usually not allowed for security reasons, so black-box attacks are more feasible. Chen et al. (2020) launched a general adversarial attack on commercial ASR systems. Cheng et al. (2019) proposed the P-RGF model to enhance black-box attack, which exploits both migration-based priority and query information. Fewer queries are used for black-box attack with high success rate. Ilyas et al. (2018) proposed to use NES as a black-box gradient estimation technique and construct adversarial samples using PGD with estimated gradients (for white-box attacks). Papernot et al. (2017) proposed to train an alternative model and have an alternative dataset with inputs from the hacker and the target model. The alternative model is used to form the adversarial sample.

3. Threat model

Scenario: The aim of this paper is to introduce a novel black-box adversarial attack on deep learning models, where an attacker can infer the model preliminarily through side-channel technology. By capturing leakage information during the reasoning operation of the neural accelerator, the attacker does not need to fully reconstruct the structure of the neural network model, but instead can construct numerous alternative models with varying network structures, layers, and other parameters using the partially known model. From the alternative models generated through queries, a simplified yet high-quality dataset is selected. This strategy not only saves time and computing resources during model training, but also enables successful adversarial attacks with a limited number of target model queries.

Capabilities: Firstly, we assume the attacker has no prior knowledge of the targeted neural network s structure and parameters. Secondly, the attacker can monitor power consumption while the model is at runtime without causing any incorrect calculations or device malfunctions. Power data from the DNN accelerator can be obtained by utilizing power monitoring programs or Trojans. We believe these assumptions are reasonable, as it is feasible to implant Trojans to acquire power traces at runtime. Thirdly, the attacker s goal is to launch an adversarial attack against the deep learning model. Therefore, attackers can generate AEs by the reduced dataset to train the shadow model.

4. System overview

We collect leaked power consumption data from AI devices to analyze their behavior while running deep learning models. These activities may reveal crucial features about the model, which can be exploited for adversarial attacks. Fig. 1 illustrates the workflow of our methodology, which comprises five stages: (1) collecting computer hardware resource consumption; (2) inferring the target model based on the collected signals (3) constructing alternative models; (4) reducing the dataset based on alternative models; (5) conducting adversarial attacks.

In the first stage, a monitoring procedure is used to collect signals of the monitored device to analyze and estimate the resource consumption. In the second stage, we employ the LSTM model to infer the target model based on the collected signals. Since LSTM model is very successful to deal with various timing tasks, such as language emotion analysis (Cao and Gao, 2020), we choose it as our basic classification model. In the third stage, we construct several different alternative models which is used to reduce the dataset with the partly known model in the side channel attack phase. In the fourth stage, we use the reduced dataset to train the shadow model and generate adversarial samples. In the final stage, an adversarial attack is launched against the target model.

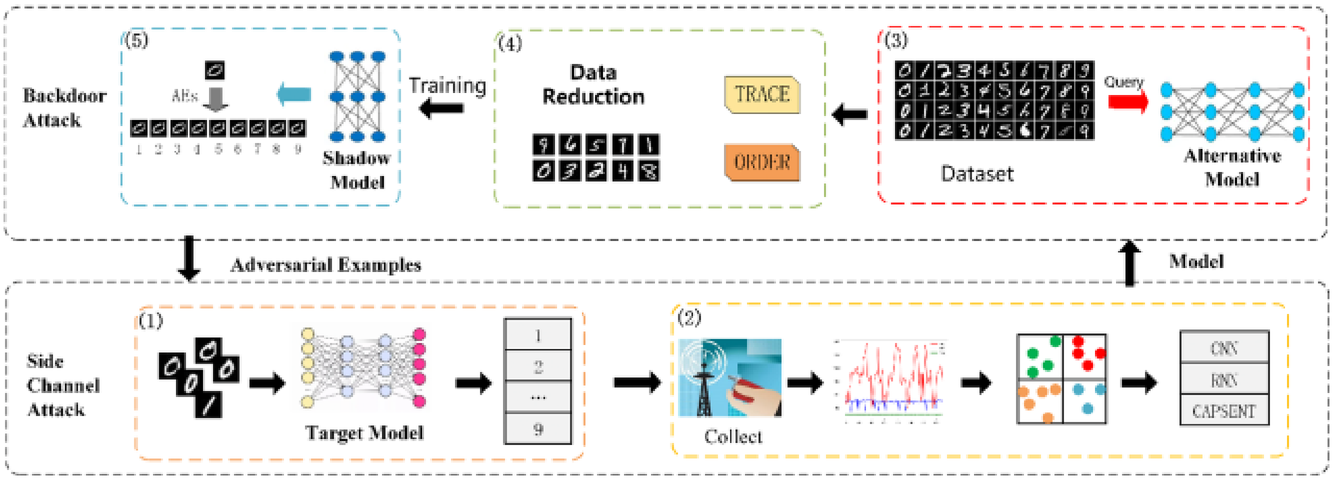


Fig. 1. The workflow of our work.

5. The proposed draasc approach

5.1. Side channel

5.1.1. Deep neural networks

In our proposed DRAASC approach, the objective of the side channel attack is to ascertain the type of the targeted model via the utilization of the LSTM model, which is based on monitored side channel signals. We selected Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and Transformer as the target models due to their strong performance in both image processing and natural language processing.

5.1.2. Collecting signals

We use the open-source monitoring software Openhardware-monitor (OpenHardwareMonitor) to measure the power usage of the target computer that runs a deep learning model. The software incorporates certain internal commands, such as 'nvidia-smi', which are utilized to monitor the status of the hardware. Since the resource consumption of a deep learning model on one computer may differ from that on another computer, we conducted experiments on multiple computers with different GPUs and CPUs.

5.1.3. Signals

Architectures of deep learning models (such as cell and fine-grained components) are different. Hence, they always impose various impacts on the power consumption of AI devices (Jha et al., 2020).

Taking GPU power consumption as an example. GPU power can be considered as the sum of Energy Per Frame (EPF), and EPF depends on MACs (multiply and accumulate operations) and energy efficiency. The former is determined by the DNN structure (filter parameters, activation function, and so on), and the latter is determined by the degree of data reuse.

We implement and test three different structures of deep learning models for image classification under the same condition, such as the same epochs and so on. In addition, we also collect the resource consumption when the computer is idle. The energy consumption is shown in Fig. 2. When deep learning models are running for classification, they require a lot of computing resources to perform calculations. Therefore, there is a significant increase on the power consumption when models are running. In Fig. 2, we eliminated the low power phases that occur at the beginning and end of the computational process, as they require minimal computing resources. We can observe that the power consumption of GPUs exhibits strong and consistent peaks for CNNs, which occur on average once every 10 s, while RNNs show regular dips in power consumption, occurring once approximately every 50 s.

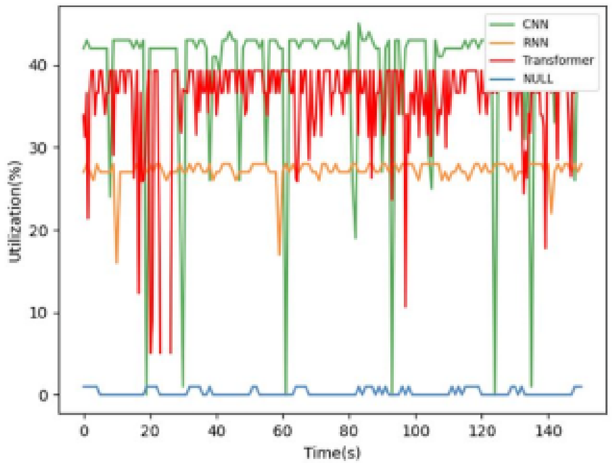


Fig. 2. GPU power consumption on 1060 for deep learning models.

Conversely, the power consumption of RNNs remains consistently low and stable. The power consumption of Transformers lies between that of CNNs and RNNs.

In another test, the GPU power consumption of three graphics cards is shown in Fig. 3 when we run the identical CNN model. As can be seen, although the GPU arithmetic power differs among the graphics cards,

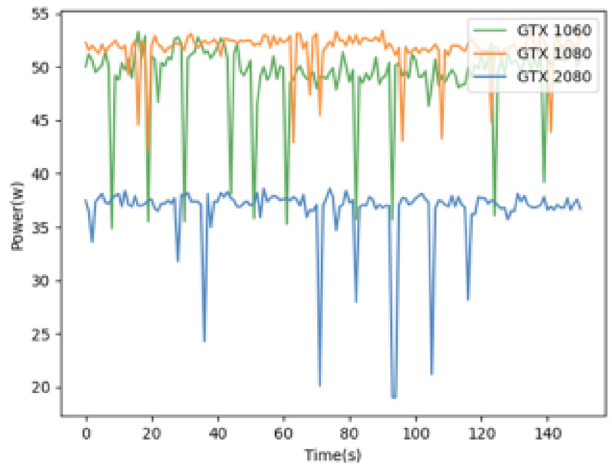


Fig. 3. GTX1060, GTX1080 and GTX2080 Power Consumption.

the traces display similar waveform patterns across all three devices, despite varying amplitudes. The difference in arithmetic power between GTX1080 and GTX1060 is small, so the wave curves of them almost merge. On the other hand, GTX2080, which is more powerful in arithmetic, expresses a very similar but more efficient power consumption pattern.

We can draw similar conclusion for other representative features collected from computers running deep learning model, such as GPU Core Load, GPU Frame Buffer, GPU Bus Interface Load, GPU Power, GPU PCIE RX and GPU PCIE TX.

5.1.4. *Model detection*

In stage 2 of Fig. 1, we utilize a deep learning model to identify the targeted model based on the collected side-channel signals. The primary objective at this stage is to classify deep learning models in real-time by analyzing the observed signals. Initially, since the signals in the time series are unbounded, we employ a Long Short-Term Memory (LSTM) network to address this issue.

5.1.5. *Feasibility*

Although side-channel attacks may not be effective in some well-protected environments, we focus on scenarios where side-channel attacks can be effectively executed. Specifically, we can exploit common Trojan horses, viruses, or backdoor vulnerabilities to deploy legitimate system power consumption monitoring programs onto the victim machine that hosts the target model. Since programs designed to gather system power consumption typically leverage legitimate system calls, they are not considered as malicious or viral by protection software.

5.2. *Data reduction*

We try to construct multiple alternative models with different model structures to launch black-box adversarial attacks, but the large amount of data used in queries leads to inefficiency, particularly if not all queries need to be executed and some data is duplicated and left unused. Therefore, it is necessary to reduce the dataset to make queries more efficient.

Data reduction is considered as a technique that removes duplicate and redundant data from a large dataset, but the remaining data is still representative.

We now describe the task of data reduction as follows:

Model stealing: In this section, the attacker utilizes an alternative model to imitate the functionality of the target model when the model is only partially known through side-channel attacks and when the dataset is available.

Target model: The purpose of the target model is to perform a specific task (e.g., image classification), and we do not query the target model directly because of limitations on the number of queries as well as efficiency. We deem the target model as a black box model which attackers cannot know.

Attacker behavior: The attacker knows the type of the deep learning model by the side channel attack and is not aware of the model details (e.g., internal structure, hyper parameters). Therefore, the attacker chooses to build classifiers with different structures and uses the obtained dataset to train them. Confidence of the classifiers can be obtained for all the data when they are output by the classifiers, and the results obtained by querying images on several classifiers are averaged to get the ranking of the images. We select images from the dataset based on this ranking and train a shadow model with these images.

Target: Our purpose is to train a shadow model that can launch a adversarial attack on the target model when the target model cannot be queried in large numbers and the structure of the target model is not available. This shadow model may improve the effectiveness of black-box attacks.

The details of our entire data reduction approach are as follows.

Algorithm 1 describes the overall process of reducing the dataset. It performs several iterations. For each iteration, it constructs an alternative model. F_c^t represents the t-th alternative model. Then the segmented sub-datasets are used to query the t-th alternative model. The subsequent step is to average sorting results of previous queries. After the loop terminates, our task is to select valuable data from the ordered results. We set the size of the reduced dataset by a reduction factor and use the reduction function $\boldsymbol{F_{sample}}$: $\boldsymbol{DATA_{sample}}$ DataReduction(**DATA** to reduce the dataset.

As shown in Algorithm 2, We classify the dataset into k classes based on the label $\boldsymbol{y_i}$ of DATA($\boldsymbol{x_i}$,$\boldsymbol{y_i}$), where k is the number of classes in the dataset, and save each sub-dataset as $\boldsymbol{N_j}$, j [1,k].

In this research, we commence by performing a side-channel attack, which enables us to acquire the model type c of the target model. This facilitates the identification of the architecture of the target model that requires attacking. Upon determining the model type, we select certain model parameters that are likely to impact performance and proceed to construct a series of alternative models F_c^t. These models are essentially variations of the target model architecture, designed to minimize data dimensionality and optimize their effect on the target model. To elaborate, we describe the steps involved in constructing these alternative models in detail below:

Consider a model A with several construction structures, denoted by A_1, A_2, , A_n. Using CNN as an example, A_i represents various parameters such as the number and size of convolution layers, the number of fully connected layers, and other relevant aspects in the convolution neural network. These parameters provide us with several options for constructing alternative models F_c^t.

For instance, A_1 is a parameter that represents the number of convolution layers in the convolution neural network. It offers a range of possible layer values, which vary from between a minimum of two to a maximum of four layers. Therefore, A_1 has three distinct options: A_{11}, A_{12}, and A_{13}. As presented in Fig. 4, in the process of constructing alternative models, this manuscript formulates the various parameter options into a directed acyclic graph (DAG), which bears resemblance to a neural network structure. Through the amalgamation of options from heterogeneous structures, the study develops novel alternative models.

To regulate the overall complexity of constructing alternative models, this paper assigns weights to different options based on their intricacy within the construction structure.

For precise model architecture, measurable structures, such as the number of layers, are normalized based on their numerical magnitude. Contrarily, imprecise structures, such as the convolution kernel size in a convolutional neural network, are sorted in ascending order according to their intricacy and subsequently normalized with respect to their position.

Following the determination of substitute model weights, this study employs a roulette wheel selection algorithm to better account for the contribution of each alternative model. The roulette wheel algorithm converts each individual s fitness value into a probability distribution

Algorithm 1
Data reduction.

```
Data: K is the number of classes of the dataset, DATA is dataset of the kind ($x_i$,$y_i$), c is
   Model Type, $S_t$ is the softmax output probability for each class of dataset on the t-th
   alternative model,$num_{move}$ is the sorting difference between $S_{t\ 1}$ and $S_t$.
1   N   SplitDATA (k,DATA $x_i$,$y_i$ )
2   $Weight$  $Struct$   ProduceAModel
3   While $num_{move}$   DATA  do
4   $S_t$  SelcetModel($F_c^t$)
5   S   Avg($S_t$)
6   $num_{move}$   CountOrder($S_{t\ 1}$ $S_t$)
7   End While
8       Reduction ratio
9   $Data_{simple}$   DataReduction(  S
```

Algorithm 2
Dataset split based on category.

```
Data: DATA is dataset of the kind ($x_i$,$y_i$);K is the number of classes of the dataset
procedure SplitDATA (k, DATA)
    k   Type of dataset
    for j   1,k do
        $N_i$   kth kind of data
    end for
return N($N_1$ $N_2$   $N_k$)
end procedure
```

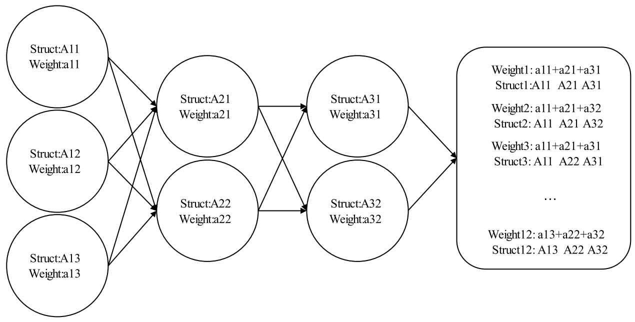


Fig. 4. Construction of alternative models.

and selects them according to the resultant probabilities.

By utilizing this method to select alternative models, the selection probability of each model is determined by its respective fitness value, which is directly proportional. The normalization of weights further reduces discrepancies between variable varying weights and encourages more equitable selection. The following steps elucidate the procedural process:

The process for selecting substitute models is outlined as follows:

(1) Compute the fitness value of each model, which serves as the probability for a roulette wheel selection. This approach utilizes a fitness function based on the weight of each model, with those possessing higher values being more likely to be selected.
(2) Normalize the fitness values to ensure an accurate probability distribution. The Softmax function is employed for this purpose, as demonstrated in the formula below:

$$Softmax \quad z_i \qquad \frac{e^{z_i}}{\sum_{c \quad 1}^{c} e^{z_c}} \tag{1}$$

(3) Generate a random number r within the range of 0 to 1.
(4) Arrange the models in descending order of probability and calculate the cumulative probability until it exceeds the randomly generated number from step 3. The corresponding substitute model is then selected.
(5) Return the selected substitute model and update the weight and structure sequences by removing the chosen model from them.

As shown in Algorithm 5, we obtain $S'_{t \quad 1}$ and S'_t from the alternative models $F^{t \quad 1}_{ctest}$ and F^t_{ctest}, respectively. And then we sort $s^k_{t \quad 1}$ and s^k_t in the k-th class as $S'_{t \quad 1}$ *and* S'_t, and record the different position of x_i between $s^k_{t \quad 1}$ and s^k_t,which is named as num_{move}.

In Algorithm 6, we fetch data every *span* samples, where *span* is calculated as *span* $\frac{subsize_i}{n}$.Then the fetched data is stored in array Data [].

Algorithm 3
ProduceAModel.

```
Data: c is the Model,
Procedure ProducePModel(c)
    $A_i$   construction structure for the type c.
    $A_{ij}$   Options under the construction structure.
    $W_{ij}$   weight of $A_{ij}$ using equation 4.1
    $Weight_t$   weight of the substitute model
    return Weight   Struct
End Procedure
```

Algorithm 4
SelcetModel.

```
Data: Weight   Struct     DATA, N
Procedure SelectAModel(c)
    W   Initialize the fitness list.
    W   Normalize Weight[] using equation 4.2.
    r   random (0,1)
    acc   Initialize cumulative probability
    for 1 to n-1
        acc   acc   w[i]
        if(acc   r):
            Nextselect   Struct i
            delete Weight i , Struct i
        End if
    End for
    $F^t_c$   Nextselect
    $F^t_{ctest}$   SaveBEST($F^t_c$ DATA $x_i$,$y_i$ )
    for 1 to k:
        $s^t_i$   EVALUATE($F^t_{ctest}$ $N_i$)
    End for
    return $S_t$($s^t_1$ $s^t_2$     $s^t_k$)
End Procedure
```

Algorithm 5
Replacement optimization.

```
Data: $S_t$ is the softmax output probability for each class on the t-th alternative model
procedure Avg($S_t$)
    S'   $\frac{S_1 \quad S_2 \qquad S_t}{t}$
    S   order S')
    return S($s'_1$ $s'_2$   $s'_k$)
end procedure
procedure CountOrder($S_{t \quad 1}$ $S_t$)
    $S'_t$   Avg($S_{t \quad 1}$)
    $S'_{t \quad 1}$   Avg($S_t$)
    for i   1, k do
        Flag_before_i   order($s'_i$ $s'_i$   $S'_{t \quad 1}$)
        Flag_now_i   order($s'_i$,$s'_i$   $S'_t$)
        for j   1,  $s'_i$  do
            if Flag_before_i[j]   Flag_now_i
                $num_{move}$   $num_{move}$   1
            end if
        end for
    end for
return $num_{move}$
end procedure
```

5.3. Adversarial attack

Adversarial attacks are considered as the important way to threaten the security of deep learning model. Attackers usually use white-box attacks to generate adversarial examples by constructing shadow

Algorithm 6
Data reduction process.

```
Data:   is the Ratio of reduction; S is the current sub-dataset
procedure DataReduction(  S
    n   (   S  10
    subsize   size of S
    span   subsize  n
    Data[]   S (0, end, span)
    return Data
end procedure
```

model. And then the generated examples are input to the target model to evaluate the effectiveness of the attacks.

In this paper, attackers can obtain probability distribution of inputs on classes by querying the alternative models, which is black-box attack. Then attackers label each input as the class with the maximum probability. In this way, they construct their reduced training dataset. Finally, Project Gradient Descent (PGD) attacks are launched on LAM. It can be considered as another form of Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) attack. It is a projected gradient descent of a negative loss function. Unlike FGSM, which attacks with only one iteration in a large perturbation, PGD perturbs in a small range during each iteration and performs multiple iterations, in which the perturbation range can be defined by attackers.

5.4. *Simplifying attacks*

For clean neural network models published on third-party model marketplaces, Users can download and deploy the model for legitimate commercial or personal applications. Therefore, if the victim system is implemented by the model available online, attackers can launch adversarial attacks based on the data reduction without the side-channel attack and constructing shadow model. However, if the target model is not available, we can employ feature engineering techniques to select the top n significant features, which is crucial in the process of side-channel attacks. Furthermore, we can also simplify the architecture of the LSTM model by reducing the number of layers, hidden units, or other methods to further simplify our approach.

5.5. *Protection*

Adding power consumption noise is an effective strategy for preventing power side-channel attacks. Firstly, we can incorporate noise into the power consumption monitored by side-channel attacks to render it difficult for attackers to extract useful information. For example, a random power consumption pattern can be created by starting and stopping unrelated programs. Furthermore, reinforcement learning can be employed to learn how to add noise in a more sophisticated and adaptive manner. The system can dynamically adjust the type, intensity, and timing of the noise based on real-time observations of the power consumption.

6. Evaluation

In order to evaluate the performance of the proposed attack, attackers track the power consumption trajectory when the model is running on the computer, and use the collected signals to identify the model. Furthermore, attackers can reduce dataset based on the identified model and generate AEs on the shadow model. We use the transferability as the evaluation metric.

6.1. *Environment configuration*

Throughout this work, we conducted experiments on computers equipped with three different resources. Table 1 provides detailed information about each computer used in our experiments. The systems have significantly different computing capacity, bandwidth, and memory resources but with identical system configuration.

Table 1
Configuration of hardware used in the experiment.

	CPU	GPU	GPU memory
Desktop I	I5-3450	GTX1060	6144MB
Desktop II	I7-4770	GTX1080	8192MB
Server	E5-2678	GTX2080	16384MB

6.2. *Signals*

We use the open-source monitoring software Open hardware-monitor (OpenHardwareMonitor) to measure the power usage of the computer. The software integrates some internal commands used to monitor the hardware status, such as *nvidia-smi*. The supported hardware suppliers include NVIDIA, AMD and Intel, and the sampling rate of built-in hardware monitoring commands is different. Take *nvidia-smi* for example, the sampling rate depends on the built-in power sensor in GPU. Therefore, it is necessary to ensure accurate measurement under the same sampling rate.

6.3. *Dataset*

We employed the TensorFlow (TensorFlow) deep learning framework to construct CNN, RNN and Transformer models for classifying the MNIST, GTRSB and CIFAR-10 datasets. The MNIST dataset is a fundamental computer vision dataset that features various handwritten numerical images (YannLecun). The utilized MNIST dataset contains 60, 000 images for training and 10,000 images for testing. The GTSRB dataset is an image collection consisting of 43 traffic signs. Specifically, we reserved 35,000 images for the training set and 4000 for the validation set (from the available 39,209 samples), while the test set contains 10,000 images (from the available 12,630). The goal for the CIFAR-10 dataset was to create a cleanly labeled subset of Tiny Images. To this end, the researchers assembled a dataset consisting of ten classes with 6000 images per class. These classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The standard train / test split is class-balanced and contains 50,000 training images and 10,000 test images. We use three DNNs as the target model, and the target model is unknown by attackers.

6.4. *Signals preliminary*

Next, we utilized a script to gather hardware resource consumption of the computer during the execution of diverse DNNs and during periods of computer idleness. We collected several parameters related to the mainboard. CPU, GPU, memory, and hard disk. And we select representative features as follows: GPU Core Load, GPU Frame Buffer, GPU Bus Interface Load, GPU Power, GPU PCIE RX, GPU PCIE TX. Fig. 5 shows the resource consumption when the deep learning models work on NVidia GTX 1060 GPUs.

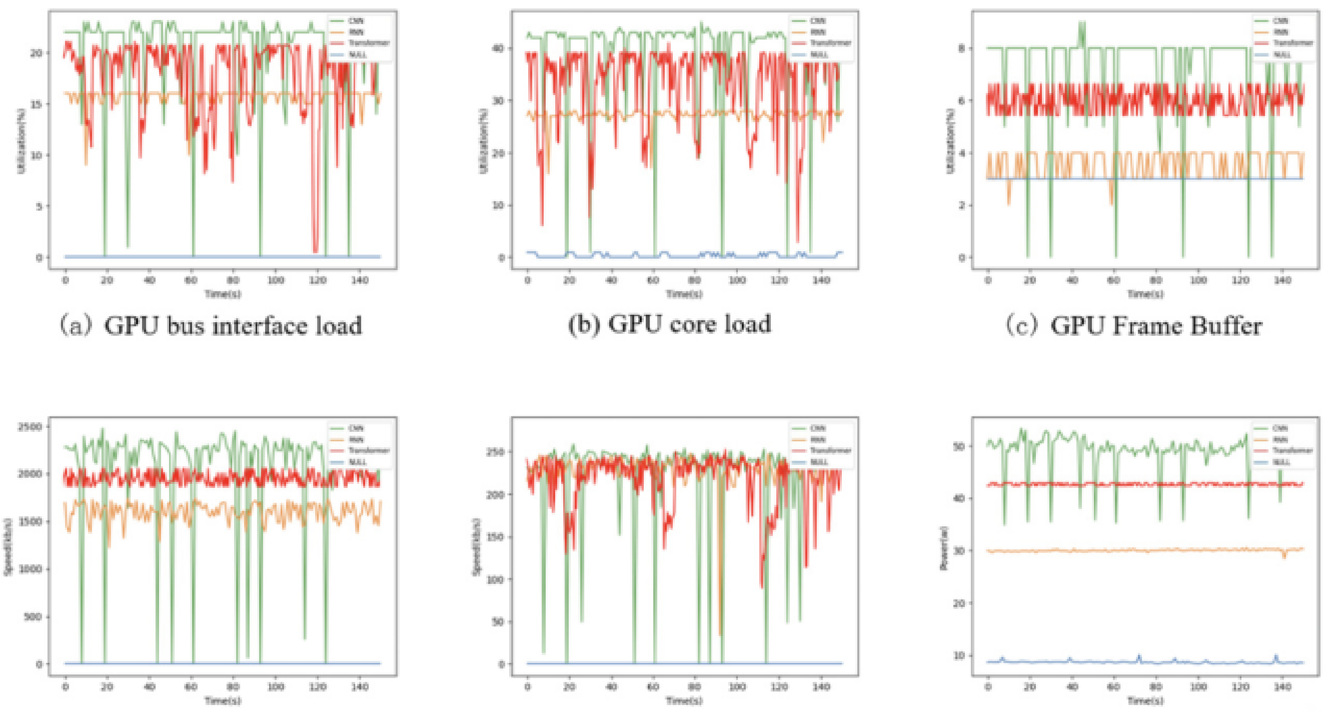
The green line corresponds to the CNN model, the orange line represents the RNN model, and the red line represents the Transformer model while the blue line denotes the computer in an idle state. Each model runs for 2.5 min, and it is evident that the GPU features exhibit a notable periodicity with multiple cycles of similar waveforms. The RNN curve remains stable owing to the large scale of the axis. However, we confirm that it remains distinguishable by scaling down the axis.

We also observe that it is easy to distinguish deep learning model by the peaks in the signal. To ascertain the generalizability of our results, we replicate the experiment on the NVidia GTX 1080 GPU and NVidia GTX 2080 GPU. By visualizing the waveform plots of the collected signals, we find that the signals are not always regular and visually distinguishable.

6.5. *LSTM*

LSTM can resolve the problem that multiple power signals are difficult to distinguish. we divide the collected datasets into two subsets: the training dataset and the test dataset. The model can be evaluated quantitatively by metrics as follows.

We denote TP as the number of true positives, FN as the number of false negatives, FP as the number of false positives, TN as the number of true negatives. Accuracy is the ratio of all correctly identified samples.



(a) GPU bus interface load (b) GPU core load (c) GPU Frame Buffer

(d) GPU PCIE Received Quantity (e) GPU PCIE Transmit Quantity (f) GPU power consumption

Fig. 5. Resource consumption of GPU.

$$Accuracy = \frac{TP + FN}{TP + FP + TN + FN} \tag{2}$$

Precision indicates the number of actual positive samples among those predicted to be positive.

$Precision = \frac{TP}{TP+FP}$(3)

Recall rate indicates the proportion of samples that are actually positive that are judged to be positive.

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

F1-Score:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

After conducting experiments, we present the results in Table 2. By utilizing the LSTM model, the attacker is able to infer the neural network model running on the computer. It is evident from Table 2 that the LSTM model outperforms other models. This indicates that the attacker can successfully leverage side-channel signals to launch attacks and identify the aforementioned three deep learning models at runtime. These research findings reveal the potential of side-channel attacks and the ability to infer model types by analyzing their performance and behavior during runtime.

6.6. Data reduction

To validate the performance of our approach, we perform the experiments on the MNIST, GTRSB and CIFAR-10 dataset. Based on the classification of the dataset, MNIST and CIFAR-10 are divided into 10 sub-datasets and GTSRB is divided into

43 sub-datasets. Our purpose is to construct a simplified dataset in each category and recombine them into a new training dataset.

We firstly classify the DNN model (e.g., CNN, RNN, Transformer) based on the side channel attack, and then construct multiple alternative models with different structures and hyperparameters. CNNs, RNNs and Transformers are deemed as target models which are subjected to adversarial attacks.

Table 3 shows C3F2's model architecture. There are 3 convolutional layers, 2 pooling layers, and 2 fully connected layers. Table 4 details RNN's model architecture. There are 3 recurrent layers. Table 5 details Transformer's model architecture.

Regarding CNN models, this study uses different parameter settings, including the convolution kernel size, the number of convolution layers, the connectivity of convolution layers, and the number of fully connected (FC) layers, to construct various alternative models. We set the maximum number of convolution layers to five and the minimum to two. Moreover, the maximum number of FC layers is three, while the minimum is two. We carry out experiments using 3 × 3 and 5 × 5 convolution kernels. As for RNN models, we consider the number of layers and different cells as parameters to construct alternative models. The number of layers represents the number of cells that can be stacked on the time sequence. In this experiment, we set the minimum number of layers to two and the maximum to five. The relationship between the

Table 2
Classification by LSTM and ILSTM.

	Accuracy	F1-Score	Recall	Precision
SVM	87.68 %	87.54 %	87.65 %	87.95 %
RNN	84.73 %	84.72 %	84.75 %	85.03 %
LSTM	89.44 %	89.17 %	89.44 %	91.98 %

Table 3
Parameters of the CNN model.

Input	1 * 28 * 28
Convolutional layer	16 * 24 * 24
Convolutional layer	32 * 20 * 20
Max-Pooling layer	32 * 10 * 10
Convolutional layer	64 * 6 * 6
Max-Pooling layer	64 * 3 * 3
Fully connected layer	100
Fully connected layer	100

Table 4
Parameters of the RNN model.

Input	784
RNN1	128
RNN2	64
RNN3	32
Output	10

Table 5
Parameters of the Transformer model.

Input	1 * 28 * 28		
Encoder	8	128	300
Encoder	8	128	300
Encoder	8	128	300
Output	10		

number of layers and the number of cells is as follows. If the number of layers is five, the maximum number of cells is 256, which decreases to 128 for four layers, and so on. We choose both RNN and Long Short-Term Memory (LSTM) cells as the cell. For the transformer model, we consider the number of encoder layers as a parameter to construct alternative architectures. In this experiment, we set the minimum number of layers to three and the maximum to five. Each layer is equipped with multi-head self-attention, layer normalization, residual connections and feedforward network. The dimensionality of the multi-head attention is 300, comprising 8 heads.

We use the sub-datasets to query the alternative models to obtain the class probability. And then we sort the queried images based on category probability. We repeat the process above until the adjacent results are similar. (We set the num_{move} used in Algorithm 1 to 6,000)

Based on the size of the reduced dataset and the size of the current sub-dataset ($subsize_i$), we set the interval $span$ to fetch data as $span$ $\frac{subsize_i}{n}$, and then fetch one data every span samples in each sorted sub-dataset. We set used in Algorithm 6 to 1 % (600), 0.5 % (300), and 0.3 % (200), which means that we set n to 60, 30, and 20 for each category in the dataset.

We use the simplified dataset to train the shadow model, and set the batch size to 64, and use the cross-entropy loss function to calculate the loss, and set adaptive moment estimation as the optimizer, and set the learning rate to 0.001.

6.7. Adversarial attack

We evaluate the similarity of the decision boundary between the target model and the shadow model by the transferability of the Adversarial Examples (AEs) attack, which is named as transferability. Deep learning models are confronted with adversarial attacks, and training shadow model is one way to launch black-box attack. By obtaining class probabilities of inputs with different structures of the same model in the data reduction phrase, we can train a shadow model and generate adversarial examples. These AEs can then be used to attack the target model. Our evaluation of attacks is based on transferability

6.7.1. Target

PGD whose perturbation can be set by attacker is essentially projected gradient descent with a negative loss function. We use PGD to generate target AEs based on last alternative model (**LAM**) trained by reduction data. Then we take these AEs that can successfully attack LAM to attack the target model. The target model is trained on the overall training dataset and performs well on the test dataset.

We set the perturbation upper limit ( ) to 0.5, and the attack becomes increasingly better as increases. As for attacks, we select 5000 images randomly in the test dataset, and we generate nine additional target AEs for each image in addition to itself. As a result, 45,000 target AEs are generated. The results are shown in Table 5. Although the details of the target model is unknown, the AEs which can attack **LAM** can also attack the target model. That means **LAM** can help attackers to generate AEs.

As shown in Table 6, For the MNIST dataset, we train the LAM shadow model using 600 data instances, and the generated adversarial examples achieve transferability of 68.28 %, 65.34 % and 66.75 %, respectively, when attacking the target model trained on the complete dataset. When trained on 300 data instances, the transferability decrease to 56.9 %, 55.62 % and 57.64 %. Similarly, when trained on 200 data instances, the transferability drop to 45.67 %, 46.79 % and 45.58 %, and further decrease to 42.72 %, 40.17 % and 40.48 % when trained on 150 data instances. As for the GTSRB dataset, we also train the LAM shadow model using 600 data instances, and the generated adversarial examples achieve transferability of 62.42 %, 59.97 % and 60.17 % when used to attack the target model trained on the complete dataset. When the model is trained on only 300 data instances, the transferability decreases to 50.61 %, 48.65 % and 48.13 %. Similarly, the transferability decreases to 43.79 %, 44.57 % and 44.91 % when the model is trained on 200 data instances, and further decreases to 40.64 %, 39.76 % and 41.02 % when the model is trained on only 150 data instances. Finally, the transferability of the three deep learning models diminishes as the number of training data instances decreases on the CIFAR-10 dataset, which is similar to the performance observed on the previous two datasets.

The SOTA methods, such as PRADA (Goodfellow et al., 2014), Practical (YannLecun) and DRMI, generate target AEs based on the model whose structure is the same as the target model. Therefore, we evaluate **DRAASC** under the same condition with the SOTA methods. The results are shown in Table 7.

Under the same perturbation settings, we found that our method achieves higher transferability rates when compared to DRMI. Our results show that for the MNIST dataset, the transferability of AEs with 600

Table 6
Transferability of adversarial examples on target model generated by LAM.

Dataset	Queries	Target Model	Transferability
MNIST	600	LAM(CNN)	68.28 %
		LAM(RNN)	65.34 %
		LAM(Transformer)	66.75 %
	300	LAM(CNN)	56.9 %
		LAM(RNN)	55.62 %
		LAM(Transformer)	57.64 %
	200	LAM(CNN)	45.67 %
		LAM(RNN)	46.79 %
		LAM(Transformer)	45.58 %
	150	LAM(CNN)	42.72 %
		LAM(RNN)	40.17 %
		LAM(Transformer)	40.48 %
GTSRB	600	LAM(CNN)	62.42 %
		LAM(RNN)	59.97 %
		LAM(Transformer)	60.17 %
	300	LAM(CNN)	50.61 %
		LAM(RNN)	48.65 %
		LAM(Transformer)	48.13 %
	200	LAM(CNN)	43.79 %
		LAM(RNN)	44.57 %
		LAM(Transformer)	44.91 %
	150	LAM(CNN)	40.64 %
		LAM(RNN)	39.76 %
		LAM(Transformer)	41.02 %
CIFAR-10	600	LAM(CNN)	74.54 %
		LAM(RNN)	72.54 %
		LAM(Transformer)	74.75 %
	300	LAM(CNN)	68.94 %
		LAM(RNN)	66.25 %
		LAM(Transformer)	66.47 %
	200	LAM(CNN)	60.47 %
		LAM(RNN)	61.79 %
		LAM(Transformer)	61.75 %
	150	LAM(CNN)	54.23 %
		LAM(RNN)	55.17 %
		LAM(Transformer)	54.48 %

Table 7
Transferability of adversarial examples on target model generated by the shadow model whose structure is the same as the target model.

Queries	Target Model	MNIST	GTSRB	CIFAR-10
600	PRADA (Wei et al., 2018)	49 %	41 %	48 %
	Practical (Hong et al., 2018)	39 %	32 %	40 %
	DRMI	78.51 %	76.32 %	75.02 %
	DRAASC(CNN)	**88.35 %**	**87.37 %**	**86.42 %**
	DRAASC(RNN)	**89.19 %**	**86.16 %**	**85.06 %**
	DRAASC(Transformer)	**88.21 %**	**87.04 %**	**85.19 %**
300	PRADA	39 %	32 %	40 %
	Practical	33 %	27 %	33 %
	DRMI	76.37 %	73.12 %	71.05 %
	DRAASC(CNN)	**87.46 %**	**84.77 %**	**81.04 %**
	DRAASC(RNN)	**86.27 %**	**83.44 %**	**80.14 %**
	DRAASC(Transformer)	**87.43 %**	**83.74 %**	**81.95 %**
200	PRADA	31 %	30 %	29 %
	Practical	28 %	25 %	26 %
	DRMI	70.13 %	70.35 %	65.17 %
	DRAASC(CNN)	**80.88 %**	**80.15 %**	**74.58 %**
	DRAASC(RNN)	**69.84 %**	**78.03 %**	**75.94 %**
	DRAASC(Transformer)	**80.14 %**	**79.88 %**	**74.06 %**
150	PRADA	29	27 %	25 %
	Practical	27 %	23 %	23 %
	DRMI	69.64 %	64.33	63.54 %
	DRAASC(CNN)	**78.41 %**	**75.47 %**	**73.06 %**
	DRAASC(RNN)	**79.14 %**	**74.38 %**	**73.18 %**
	DRAASC(Transformer)	**79.85 %**	**74.04 %**	**73.85 %**

queries is 88.35 % with our method, while it is only 78.51 % with DRMI. Similarly, for the

GTSRB dataset, the transferability of AEs with 600 queries is 84.37 % with our method, as opposed to only 76.32 % with DRMI. Regarding the CIFAR-10 dataset, AEs with 600 queries exhibit a transferability rate of 85.19 %, significantly outperforming 75.02 % transfer rate of AEs crafted with the DRMI approach. Our method outperforms DRMI by nearly ten percentage points under the same circumstances, and it is notably superior to other methods, including PRADA. While the transferability rates of all methods decrease as the number of queries decreases, our method maintains a higher transferability rate. Taking Tables 6 and 7 into account, we find that even though we use LAM, we still achieve a higher transferability than PRADA and Practical.

Fig. 6 shows the confusion matrices of targeted AEs attacks against CNN under 150 and 600 queries. The value in i th row, j-th column represents the number of samples whose original label is i which is classified into j. The diagonal elements are the number of failed attacks. Other elements are the number of successful attack samples. The lighter the color is, the larger the value is. Obviously, the (3,3) element in 150 queries is the lightest, which means many adversarial samples generated by the samples with label 3 do not succeed in attacks. In 600 queries, the (3,3) element turns darker due to the larger dataset. Fig. 7 shows the confusion matrices of targeted AEs attacks against RNN under 150 and 600 queries. Similarly, Fig. 8 shows the confusion matrices of targeted AEs attacks against Transformer under 150 and 600 queries. Label 3 still performs the worst in both Figures. We can conclude that it is difficult to

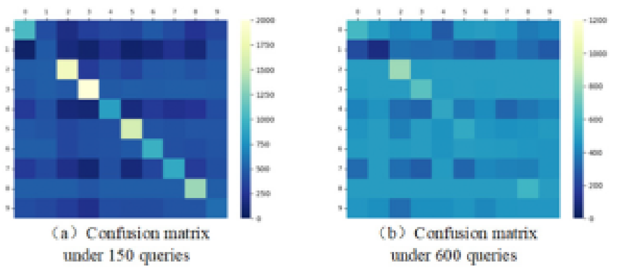

（a）Confusion matrix under 150 queries　（b）Confusion matrix under 600 queries

Fig. 6. Confusion matrices of targeted adversarial examples attacking the target CNN model.

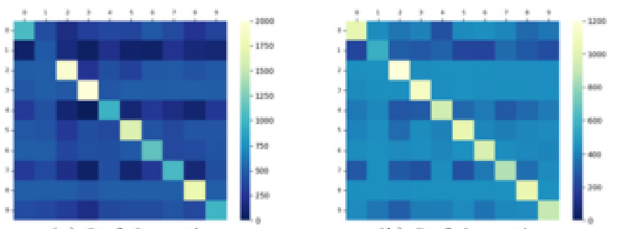

（a）Confusion matrix under 150 queries　（b）Confusion matrix under 600 queries

Fig. 7. Confusion matrices of targeted adversarial examples attacking the target RNN model.

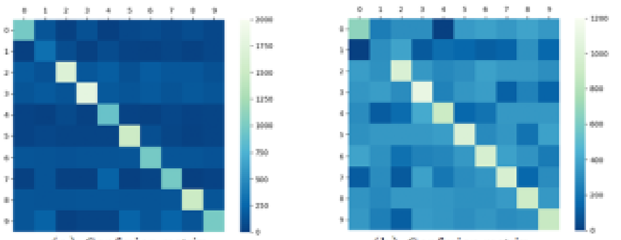

（a）Confusion matrix under 150 queries　（b）Confusion matrix under 600 queries

Fig. 8. Confusion matrices of targeted adversarial examples attacking the target Transformer model.

attack samples in label 3 successfully. However, the attacks for other labels are more effective. We can draw similar conclusion on confusion matrices in GTSRB and CIFAR-10 dataset. Therefore, we will not elaborate on that further due to space limitations.

6.7.2. *Untargeted*

We also use the PGD approach to generate untargeted adversarial images with LAM to attack the target model. We set ε (max perturbation) as 0.3, and test the transferability of 1000 untargeted AEs for each experiment. The results are shown in Table 8. For the MNIST dataset, our method uses 150 queries to generate 1000 non-target AEs, and we achieve a transferability of 70.1 %, and we can generate an AE in about 0.09 s. For the GTSRB dataset, our method employs 150 queries to generate 1000 non-target adversarial examples, achieving the transferability of 66.95 %, and we are able to generate an adversarial example within approximately 0.131 s. Finally, our approach employs 150 queries to produce 1000 untargeted AEs) for the CIFAR-10 dataset. We achieve a transferability rate of 70.95 % while generating each AE in approximately 0.126 s.

Furthermore, we conduct experiments to generate target AEs based on the model with the same structure as the target model using SOA methods and DRAASC. With ε (max perturbation) set to 0.3, we test the transferability rate of 1000 untargeted AEs for each experiment and present the results in Table 9. For the MNIST dataset, DRMI requires 150

Table 8
Transferability of untargeted adversarial examples on target model generated by shadow model (LAM).

Dataset	Method	Transfer-ability	Queries	time
MNIST	**LAM(CNN)**	**70.1 %**	**150**	**0.089**
	LAM(RNN)	**65.82 %**	**150**	**0.089**
	LAM(Transformer)	**68.13**	**150**	**0.089**
GTRSB	**LAM(CNN)**	**66.95 %**	**150**	**0.131**
	LAM(RNN)	**61.03 %**	**150**	**0.131**
	LAM(Transformer)	**65.08**	**150**	**0.131**
CIFAR-10	**LAM(CNN)**	**70.95 %**	**150**	**0.126**
	LAM(RNN)	**64.03 %**	**150**	**0.126**
	LAM(Transformer)	**68.08**	**150**	**0.126**

Table 9
Transferability of untargeted adversarial examples on target model generated by shadow model whose structure is the same as the target model.

Dataset	Method	Transfer-ability	Queries	time
MNIST	GE (Papernot et al., 2017)	61.5 %	196	0.011
	DRMI (Xiang et al., 2020)	73.2 %	150	0.113
	DRAASC	77.1 %	150	0.089
GTSRB	GE (Papernot et al., 2017)	55.72 %	196	0.021
	DRMI (Xiang et al., 2020)	69.64 %	150	0.154
	DRAASC	73.18 %	150	0.131
CIFAR-10	GE (Papernot et al., 2017)	68.47 %	196	0.027
	DRMI (Xiang et al., 2020)	73.64 %	150	0.143
	DRAASC	78.87 %	150	0.126

model queries and two minutes to generate 1000 non-target AEs with a transferability rate of 73.2 % against the target model. Our method, DRAASC, also generates 1000 non-target AEs with 150 queries and achieves a higher transferability rate of 77.1 % with an AE generation time of approximately 0.09 s. Our method outperforms DRMI by 3.95 % while reducing the execution time. For the GTSRB and CIFAR-10 datasets, our approach remains effective in attacking.

6.8. *A real-world case*

We conduct a test using facial images system (Fig. 9.a) of individuals not enrolled in the system, and verify that the system does not unlock the door (Fig. 9.b). Then, we apply our proposed attack approach on this system. Firstly, a signal collection program is embedded into the control system. Secondly, the LSTM model is used to identify the type of facial recognition model (CNN). Thirdly, a shadow model is constructed to generate images which is capable of deceiving the target model. Finally, we employ the shadow model to generate adversarial examples corresponding to the facial image used in Fig. 9.b, which successfully tricks the system into unlocking the door (Fig. 9.c). To safeguard personal privacy, we apply mosaics to the facial images in Fig. 9 for illustration purposes; however, no mosaics are used during the experiments.

7. Conclusion

In this paper, we propose a novel adversarial attack combined with side-channel attack, which can be launched without querying the target model to achieve complete black-box attack. Firstly, we use LSTM to obtain the partial target model by the side-channel attacks. Then a novel dataset reduction technique is utilized to improve the attack efficiency. Instead of querying to the target model, attackers construct alternative models to select high-quality representative data to launch adversarial attack. Our approach outperforms SOTA methods in terms of transferability for both targeted and untargeted AE attacks, by nearly 10 % and 3 % respectively.

CRediT authorship contribution statement

Hanxun Zhou: Writing – review & editing, Validation, Supervision, Resources, Investigation, Funding acquisition, Conceptualization. **Zhihui Liu:** Writing – review & editing, Validation, Supervision, Formal analysis, Data curation, Conceptualization. **Yufeng Hu:** Writing – original draft, Validation, Resources, Formal analysis, Data curation. **Shuo Zhang:** Validation, Supervision, Investigation. **Longyu Kang:** Validation, Formal analysis. **Yong Feng:** Supervision, Project administration, Funding acquisition, Formal analysis. **Yan Wang:** Supervision, Resources, Project administration. **Wei Guo:** Resources, Project administration, Funding acquisition. **Cliff C. Zou:** Writing – review & editing, Validation, Supervision.

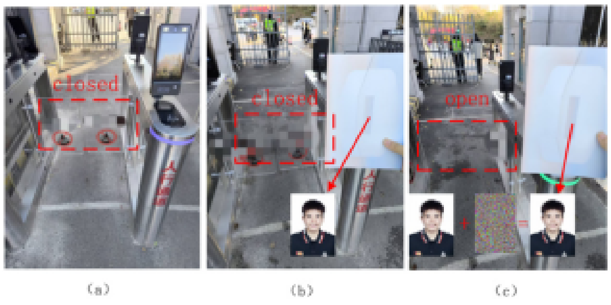


Fig. 9. Example of attacking campus access control system.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge constructive comments by the anonymous reviewers. This work was supported in part by the National Key Research and Development Program under Grant 2019YFB1406002, in part by the National Science Foundation of China under Grant 51704138, in part by the Key Scientific Research Project of Liaoning Provincial Department of Education under Grant LZD202002, in part by the Liaoning Education Department under Grant JYT19053, in part by the National Natural Science Foundation of Liaoning under Grant 2020-MS-239, in part by Teaching Reform Project of Liaoning University under Grant JG2020YBXW127.

Data availability

Data will be made available on request.

References

Acharya, R.Y., Ganji, F., Forte, D., 2022. Information theory-based evolution of neural networks for side-channel analysis. In: IACR Transactions on Cryptographic Hardware and Embedded Systems.
Aldahdooh, A., Hamidouche, W., Fezza, S.A., Déforges, O., 2022. Adversarial example detection for dnn models: a review and experimental comparison. Artif. Intell. Rev. 55 (6), 4403–4462.
Azizi, S., Mustafa, B., Ryan, F., Beaver, Z., Freyberg, J., Deaton, J., Loh, A., Karthikesalingam, A., Kornblith, S., Chen, T., et al., 2021. Big self-supervised models advance medical image classification. In: *Proc. IEEE/CVF Int. Conf. Comput. Vis., August*, pp. 3478–3488.
Bhagoji, A.N., He, W., Li, B., Song, D., 2017. Exploring the space of black-box attacks on deep neural networks. arXiv preprint. *arXiv:1712.09491*. Dec.
Cao, S., Gao, P., 2020. Lstm-gate cnn network for aspect sentiment analysis. In: 2020 5th Int. Conf. Inf. Sci. Comput. Tech. Transp., Computer Technology and Transportation, November, pp. 443–447. https://doi.org/10.1109/ISCTT51595.2020.00084.
Chen, Y., Yuan, X., Zhang, J., Zhao, Y., Zhang, S., Chen, K., Wang, X., 2020. Devil's whisper: a general approach for physical adversarial attacks against commercial black-box speech recognition devices. In: *USENIX Secur. Symp., January*, pp. 2667–2684.
Cheng, S., Dong, Y., Pang, T., Su, H., Zhu, J., 2019. Improving black-box adversarial attacks with a transfer-based prior. Adv. Neural Inf. Process. Syst. 32.
Chitta, K., Alvarez, J.M., Haussmann, E., Farabet, C., 2019. Training Data Distribution Search with Ensemble Active Learning. May.
Coqueret, B., Carbone, M., Sentieys, O., Zaid, G., 2023. When side-channel attacks break the black-box property of embedded artificial intelligence. In: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, pp. 127–138.
Fnu, S., Chi, J., Evans, D., Tian, Y., 2020. Hybrid batch attacks: finding black-box adversarial examples with limited queries. In: USENIX Secur. Symp..
Goodfellow, I.J., Shlens, J., Szegedy, C., 2014. Explaining and harnessing adversarial examples. arXiv preprint. *arXiv:1412.6572*. December.
Gupta, P., Drees, J.P., ˈullermeier, E.H, 2023. Automated side-channel attacks using black-box neural architecture search. In: Proceedings of the 18th International Conference on Availability, Reliability and Security, pp. 1–11.

He, Y., Meng, G., Chen, K., Hu, X., He, J., 2021. Drmi: a dataset reduction technology based on mutual information for black-box attacks. In: USENIX Secur. Symp., August, pp. 1901 1918.

Hong, S., Davinroy, M., Kaya, Y., Locke, S.N., Rackow, I., Kulda, K., Dachman-Soled, D., Dumitras, T., 2018. Security analysis of deep neural networks operating in the presence of cache side-channel attacks. arXiv preprint. *arXiv:1810.03487*. October.

Hua, W., Zhang, Z., Suh, G.E., 2018. Reverse engineering convolutional neural networks through side-channel information leaks. In: Proc. 55th Annu. Des. Autom. Conf., September, pp. 1 6.

Ilyas, A., Engstrom, L., Athalye, A., Lin, J., 2018. Black-box adversarial attacks with limited queries and information. In: Int. Conf. Mach. Learn. PMLR, April, pp. 2137 2146.

Jha, N.K., Mittal, S., Kumar, B., Mattela, G., 2020. Deeppeep: exploiting design ramifications to decipher the architecture of compact dnns. ACM J. Emerg. Tech. Comput. Syst. 17 (1), 1 25.

Kholidy, H.A., Erradi, A., 2019. Vhdra: a vertical and horizontal intelligent dataset reduction approach for cyber-physical power aware intrusion detection systems. Secur. Commun. 19.

Khowaja, S.A., Lee, I.H., Dev, K., Jarwar, M.A., Qureshi, N.M.F., 2022. Get your foes fooled: proximal gradient split learning for defense against model inversion attacks on iomt data. IEEE Trans. Netw. Sci. Eng. 1 10. https://doi.org/10.1109/TNSE.2022.3188575.

Li, Z., Du, Y., Zhu, M., Zhou, S., Zhang, L., 2022. A survey of 3d object detection algorithms for intelligent vehicles development. Artif. Life Robot. 1 8.

Liu, Y., Srivastava, A., 2020. Ganred: Gan-based reverse engineering of dnns via cache side-channel. In: Proc. 2020 ACM SIGSAC Conf. Cloud Comput. Secur, August, pp. 41 52.

Luo, Y., Duan, S., Gongye, C., Fei, Y., Xu, X., 2022. Nnrearch: a tensor program scheduling framework against neural network architecture reverse engineering. In: *2022 IEEE 30th Annu. Int. Symp. Field-Programmable Custom Comput. Mach., March*, pp. 1 9.

OpenHardwareMonitor. [Online]. Available: https://openhardwaremonitor.org.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A., 2017a. Practical black-box attacks against machine learning. In: Proc. 2017 ACM Asia Conf. Comput. Commun. Secur., pp. 506 519.

Qi, F., Chen, Y., Li, M., Yao, Y., Liu, Z., Sun, M., 2020. Onion: a simple and effective defense against textual backdoor attacks. arXiv preprint. *arXiv:2011.10369*. November.

Rakin, A.S., Chowdhuryy, M.H.I., Yao, F., Fan, D., 2022a. Deepsteal: advanced model extractions leveraging efficient weight stealing in memories. In: 2022 IEEE Symp. Secur. Priv., November, pp. 1157 1174. https://doi.org/10.1109/SP46214.2022.9833743.

Rakin, A.S., Chowdhuryy, M.H.I., Yao, F., Fan, D., 2022b. Deepsteal: advanced model extractions leveraging efficient weight stealing in memories. In: 2022 IEEE Symp. Secur. Priv., August, pp. 1157 1174. https://doi.org/10.1109/SP46214.2022.9833743.

Tang, X., Mahloujifar, S., Song, L., Shejwalkar, V., Nasr, M., Houmansadr, A., Mittal, P., 2022. Mitigating membership inference attacks by Self-Distillation through a novel ensemble architecture. In: 31st USENIX Secur. Symp., October, pp. 1433 1450.

TensorFlow. [Online]. Available: http://www.tensorflow.org/.

Tertytchny, G., Michael, M.K., 2020. dataset reduction framework for intelligent fault detection in iot-based cyber-physical systems using machine learning techniques. In: 2020 Int. Conf. Omni-layer Intell. Syst., March, pp. 1 6.

Wang, X., Kou, L., Sugumaran, V., Luo, X., Zhang, H., 2021. Emotion correlation mining through deep learning models on natural language text. IEEE Trans. Cybern. 51 (9), 4400 4413. https://doi.org/10.1109/TCYB.2020.2987064.

Wei, L., Luo, B., Li, Y., Liu, Y., Xu, Q., 2018. I know what you see: power side-channel attack on convolutional neural network accelerators. In: Proc. 34th Annu. Comput. Secur. Appl. Conf., March, pp. 393 406.

Xiang, Y., Chen, Z., Chen, Z., Fang, Z., Hao, H., Chen, J., Liu, Y., Wu, Z., Xuan, Q., Yang, X., 2020. Open dnn box by power side-channel attack. IEEE Trans. Circuit. Syst. II: Express Brief. 67 (11), 2717 2721. https://doi.org/10.1109/TCSII.2020.2973007.

Yan, M., Fletcher, C., Torrellas, J., 2020. Cache telepathy: leveraging shared resource attacks to learn dnn architectures. In: USENIX Secur. Symp., August.

YannLecun. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

Hanxun Zhou was born in 1981. He received the master degree in computer application technology from Northeastern University in 2006. He received the PhD degree in computer application technology from Northeastern University in 2009. Now he is an Associate Professor at Liaoning University. His research interests include network security, especially malcode.

Zhihui Liu was born in 1998. He graduated from Tianjin University of Science and Technology in 2021. Now he is a master s student at Liaoning University. Follow Dr. Zhou Hanxun to learn the direction of network security based on machine learning, etc.

Yufeng Hu was born in 2000. He graduated from Henan University of Technology in 2022. Now he is a master s student at Liaoning University. Follow Dr. Zhou Hanxun to learn the direction of network security based on machine learning, etc.

Shuo Zhang was born in 1998. He graduated from Qufu Normal University of technology in 2021. Now he is a master s student at Liaoning University. Follow Dr. Zhou Hanxun to learn the direction of network security based on machine learning, etc. CCF member: O1135G.

Longyu Kang was born in 1998. He graduated from TaiShan University in 2020. Now he is a master s student at Liaoning University. Follow Dr. Zhou Hanxun to learn the direction of network security based on machine learning, etc.

Yong Feng was born in 1973. He received the Ph.D. degree in management science and engineering from Northeastern university in 2007 year. He is a professor and M.S. supervisor at Liaoning university. His research interests include network security, data mining, personal recommendation etc. CCF member: E200030948M.

Yan Wang was born in 1978. She received the Ph.D degree. Now she is a professor and M. S. Supervisor at Liaoning University, and the member of CCF. Her research interests include network security, big data, internet of things and blockchain technology.

Wei Guo was born in 1983. She received the master degree in pattern recognition and artificial intelligence from Northeastern University in 2008. She received the PhD degree in pattern recognition and artificial intelligence from Northeastern University in 2011. Now she is a lecturer at Shenyang Aerospace University. Her research interests include network security and graph processing.

Cliff C. Zou (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the Department of Automation, University of Science and Technology of China, in 1996, 1999, and 2005, respectively. He is currently an Associate Professor with the University of Central Florida. His research interests include computer and network security, computer networking and network modeling, and performance evaluation.