Stealthy Inference Attack on DNN via Cache-based Side-Channel Attacks

Han Wang, Syed Mahbub Hafiz, Kartik Patwari, Chen-Nee Chuah, Zubair Shafiq, and Houman Homayoun
University of California, Davis, CA, USA
{hjlwang, shafiz, kpatwari, chuah, zshafiq, hhomayoun}@ucdavis.edu,

***Abstract*—The advancement of deep neural networks (DNNs) motivates the deployment in various domains, including image classification, disease diagnoses, voice recognition, etc. Since some tasks that DNN undertakes are very sensitive, the label information is confidential and contains a commercial value or critical privacy. This paper demonstrates that DNNs also bring a new security threat, leading to the leakage of label information of input instances for the DNN models. In particular, we leverage the cache-based side-channel attack (SCA), i.e., Flush+Reload on the DNN (victim) models, to observe the execution of computation graphs, and create a database of them for building a classifier that the attacker can use to decide the label information of (unknown) input instances for victim models. Then we deploy the cache-based SCA on the same host machine with victim models and deduce the labels with the attacker's classification model to compromise the privacy and confidentiality of victim models. We explore different settings and classification techniques to achieve a high attack success rate of stealing label information from the victim models. Additionally, we consider two attacking scenarios: binary attacking identifies specific sensitive labels and others while multi-class attacking targets recognize all classes victim DNNs provide. Last, we implement the attack on both static DNN models with identical architectures for all inputs and dynamic DNN models with an adaptation of architectures for different inputs to demonstrate the vast existence of the proposed attack, including DenseNet 121, DenseNet 169, VGG 16, VGG 19, MobileNet v1, and MobileNet v2. Our experiment exhibits that MobileNet v1 is the most vulnerable one with 99% and 75.6% attacking success rates for binary and multi-class attacking scenarios, respectively.**

***Index Terms*—Inference Attack, Deep Neural Network, Privacy Leakage, Side-Channel Attack**

I. Introduction

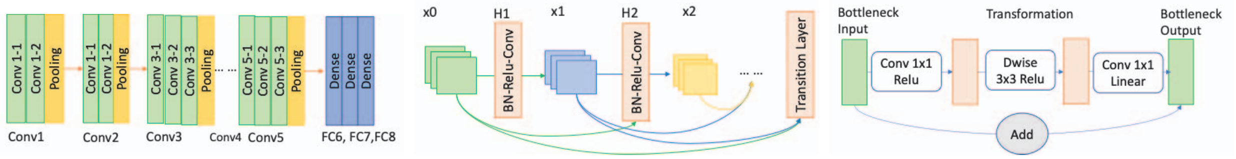
Deep neural networks (DNNs) have made significant progress in the past decade and gained increasing popularity in undertaking different tasks, including image classification [4], [9], [25], language processing [2], [5], security enhancement [23], etc. Several successful deep neural network models have been proposed and received notable success, including but not limited to VGG [25], DenseNet [14], etc. The advancement of DNNs magnifies the deployment on both servers and edge devices with different computation resources and energy restrictions [12], [14]. Since then, a number of DNN-enabled applications have been deployed in past decades across various critical domains, such as disease diagnosis [8], [21], intelligent surveillance [16], [32], financial decision [15], and so on. However, the DNN models also bring new security risks — the leakage of label information may cause financial loss and privacy compromise since the label information of such DNN-enabled applications is directly linked to users' crucial decisions and sensitive information. Taking the investment decision-related applications [15] as an example, the leakage of label information can expose the big financial decision to attackers who can take advantage of them and make an illicit profit out of it.

Hence, it is essential to investigate whether DNN models are vulnerable in terms of the leakage of label information. This paper presents a novel inference attack targeting to steal label information of DNN models by leveraging side-channel attacks (SCAs). In particular, we show that a cache-based SCA, e.g., Flush+Reload, can be employed to spy on DNN models' computations, and the extracted observations can deduce the label information of DNN models via machine learning (ML)-based classification techniques. The novelty of our demonstrated attack is to identify a correlation between the cache-based side channel of DNN models' computation trajectory and label information that may contain the users' medical, financial decisions, or other critical information.

Though prior works have demonstrated that using side-channel attacks can steal the architectures of DNN models [11], [13], [19], [30], our inference attack is more challenging than reconstructing the architectures of DNN models for the following reasons. a) The black-box DNN models make it almost impossible to infer the computation graphs based on the label information. b) The inference phase takes a much shorter time than loading DNN models, leaving less time for extracting computation traces for our attack. c) Extracting DNN model architectures uses multiple traces to remove noises while attacking label information demands the attacker obtains label information with only one trace. To address the challenge of noise from SCAs' observations and the incurred accuracy decrease, we explore the various threshold settings for removing repeated DNN models' computation observations and analyze the impact of using different ML techniques for the attacker to decide the optimal one. What is more, we select multiple models with both static neural networks (DenseNet and VGG) and dynamic neural networks (MobileNet) to illustrate the attacker's success rate, i.e., attacker's classification accuracy and highlight the vast existence of the introduced attack. Lastly, this work also identifies the most prominent computations of victim models for the attack, providing insights into the leakage source and future mitigation research. The contributions of our work are categorized as follows:

- To the best of our knowledge, this is the first work to introduce a stealthy attack that steals the label information of DNN models via cache-based SCAs.

978-3-9819263-6-1/DATE22/©2022 EDAA



a) VGG b) DenseNet c) MobileNet v2

Fig. 1: Architectures of VGG, DenseNet, and MobileNet v2

- To simulate the attack in the real world, we prototype two variants of the demonstrated attack: a) binary attacking: identifying the sensitive and non-sensitive label information; and b) multi-class attacking: deducing the exact classes of victim models.
- We also identify the most prominent computations of victim DNN models for the attacker to achieve a high success rate, providing insights into the leakage source and future mitigation research.
- We evaluate the attack with two types of victim DNN models, i.e., dynamic neural networks and static neural networks, to manifest the broad existence of the vulnerability.

II. Background and Motivation

A. Deep Neural Network (DNN)

DNN models can be categorized into *static* neural networks with the same architectures for all inputs and *dynamic* neural networks with an adaption of architectures and parameters for different inputs. This work selects three DNN families for both static and dynamic DNN models, detailed in the following.

1) Static Neural Networks: VGG was firstly proposed by Simonyan and Zisserman in 2014 [25] for localization and classification tasks. As shown in Figure 1-a), VGG has five blocks of convolutional layers initially and is followed by three fully connected layers. Each convolutional layer has a small kernel size of 3×3 with a stride and padding to maintain the same spatial dimensions as the last layer. The depth of VGG varies from 16 to 19 layers, and each of them is a variant DNN model: VGG 16 and VGG 19.

Dense Convolutional Network (DenseNet), as shown in Figure 1-b), employs a feed-forward fashion to connect each layer to every other layer in the network. It maximizes the information flow to alleviate the vanishing gradient problem and strengthen feature propagation [14]. Specifically, for lth layer, there are l inputs from all preceding convolutional blocks, and its output sends to the $L-l$ subsequent layers. This work chooses two variants of the DenseNet family: DenseNet 121 and DenseNet 169.

2) Dynamic Neural Networks: MobileNet v1 [12] and MobileNet v2 [22] were designed for conducting classification, detection, and other computer vision-related tasks in mobile devices. They enable applications installed in mobile devices to equip more functionalities with human and real-world interaction based on deep learning neural networks. MobileNet v1 [12] mainly leverages depthwise separable filters, width multiplier, and resolution multiplier to balance the accuracy loss and computation size. As depicted in Figure 1-c), MobileNet v2 includes another two techniques, a) linear bottlenecks between layers, and b) shortcut connections between the bottlenecks.

B. Cache-based Side-Channel Attacks

To bridge the latency between memory and CPU, cache hierarchies are introduced and shared among applications. The shared cache gives the attacker opportunity to influence applications' memory access and infer their cache access pattern by measuring its accessing latency, termed as cache-based side-channel attacks (SCAs). The existing cache-based SCAs, including [18], [27], [28], [31], spy on shared cache activities and steal critical information, including passwords, secret keys, etc. In this work, we leverage *Flush+Reload* to observe the computation behaviors of victim DNN models. This attack targets the *Last-Level Cache* in the CPU, flushes out victim applications' data in the cache and waits for the victim application to execute. After flushing the cache, the attacker tries to access the data and measures the accessing time (latency). Shorter accessing time denotes that the victim application has accessed the data; otherwise, it has not been accessed.

C. Motivation of the Attack

DNNs have been increasingly used for critical domains, from financial decisions [15], energy control [24], autonomous systems [26], medical treatment [7], etc. Their label information either contains critical information or impacts significant decisions, which attackers can steal and make an undesired profit out of them or conduct crimes based on them. In the finance domain, attackers can misuse the investment suggestions stolen from victim DNNs. Energy controlling systems [24] leverage DNNs to design the optimal online power control policy while the attacker with label information can take advantage of the policy information to deliberately overload the energy network and cause a denial-of-service attack on customers. Hence, we believe that the leakage of such label information of sensitive DNNs-enabled applications can allow attackers to cause undesirable damages.

III. Overview of Attack

This work demonstrates a novel attack that stealthily spies side-channel information to deduce labels of inputs. As depicted in Figure 2, the whole attack contains two parts: *offline preparation* and *online deployment*. Offline preparation firstly mimics the victim model based on architecture and weights knowledge. And then, it collects Flush+Reload traces during the inference phase of the mimicked model to build the

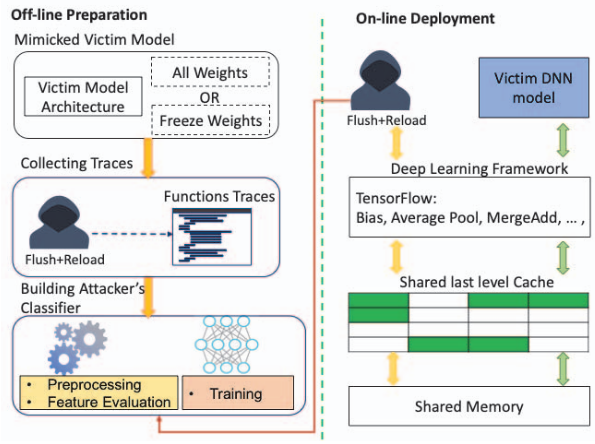


Fig. 2: Design of the presented attack

attacker's classifier. Once trained and tested, Flush+Reload is launched online to collect the traces of the victim model and send it to the attacker's classifier for deducing the label information of inputs. We introduce the details of the attack in the following sections.

A. Threat Model

The attacker's goal is to infer the labels of instances sent to the victim DNN models secretly. We assume that the attacker (without sudo access) resides in the same physical machine with DNN models (also referred to as victims) and can launch a software program on the machine. The victim is a DNN model and takes images as input, the output of which is labels and needs protection. We assume that the attacker *does not have direct access* to the *input images* and *label information*.

1) Assumption: The attack aims to obtain the label information of instances sent to victim DNN models and has the knowledge of victim models' weights or untrainable weights. We also assume that the attacker knows the label candidates of victim models and the type of dataset victim models are targeting, e.g., flowers classification [1]. As for the knowledge of victim models' architectures and parameters, we have two assumptions: setting A and B, as listed below.

- Setting A: the attacker has the full knowledge of the victim model, including architectures, weights, and parameters, which can be acquired based on approaches studied in prior research [13], or victim models use public weights.
- Setting B: the attacker has the knowledge of the victim model and the pretrained weights used while not knowing the newly added layers at the end of the model. Additionally, victim models freeze pretrained parameters and weights and only train the newly added layers.

2) Target of the Attack: We consider two attacking scenarios as detailed in the following.

- *Binary Attacking:* the attacker targets stealing one particular class of labels and categorizes the rest classes of labels as "others." The binary classification accuracy can directly reflect the ability to steal the labels from victim DNN models with a specific value that attackers are interested in.
- *Multi-class Attacking:* the attacker tries to identify all classes of labels just as victim models do.

TABLE I: Monitored function list

Bias	Sigmoid	RunHelper	Average Pool
Relu6	Depthwiseconvop	End Conv	Max Pool
MatMul	Depthwiseconv2d	Tanh	LaunchConv
Elementwise	Mulop	Elu	Concat
Merge Add	Selu	Softsign	Softplus

B. Collecting Training Traces

To observe the computations of victim DNN models, we select functions listed in Table I and each function corresponds to the specific architecture of victim DNN models. Though there are 20 functions selected during the offline preparation, only a subset of the functions is chosen for the online deployment. Our attacker and victims run at the user level on the same operating system and the host machine. The attacker runs the mimicked DNN model with a similar database as the victim while Flush+Reload is initiated to monitor the target functions simultaneously.

C. Building Attacker's Inference Model

1) Prepossessing: Since traces obtained via Flush+Reload are noisy due to transient execution, we need to take a further step to clean traces and filter out some functions. Since Flush+Reload might suffer from repeated observations when the computations of DNNs continue, we employ the number of cycles between two computations to clean noises in traces, referred to as "threshold" in the following sections. In Section IV-C, we demonstrate the influence of threshold on the classification accuracy of the attacker's inference model. After removing noises, all observations are converted into the occurrences of each computation.

TABLE II: Prominent functions for the introduced attacker

Mobilenetv1	Mobilenetv2	DensetNet121	DensetNet169	VGG16	VGG19
Bias	Bias	Maxpool	Mulop	Bias	Mulop
Mergeadd	Mergeadd	LaunchConv	Mergeadd	Mulop	Mergeadd
LaunchConv	LaunchConv	Concat	LaunchConv	Maxpool	LaunchConv

2) Feature Evaluation: Though the introduced attack has access to monitor several functions as listed in Table I, we conduct an importance evaluation for each function for two reasons: a) to minimize the number of monitored functions to incur the least the influence of our attack on victims and reduce the possibility of being detected; and b) the importance of functions reveals the leakage source and can provide more insights for future mitigation works. To achieve this, we leverage the correlation-based feature selection (CFS) with the greedy-stepwise search algorithm [3] approach to select the optimal subset functions for implementing an attack with a high success rate, i.e., high classification accuracy. After the importance evaluation, we observe that *Bias, Merge Add, Launch Conv* are commonly effective for attacking all six victim models. Hence, only the prominent functions are monitored for online deployment.

3) Training Classifier: We consider two types of attacking purposes: identifying the sensitive inference model (binary attacking) and identifying the precise class of victim's inputs (multi-class attacking). To build a robust classifier with higher accuracy, we split the whole dataset into training and validation in case of an over-fitting issue. Furthermore, we consider a vast range of ML classifiers in this work to select the optimal one with the highest accuracy. Five classifiers are selected: OneR, J48, SVM-based SMO, KNN, and Multi-Layer Perceptron (MLP). The rationale for selecting these ML models is that they are from different branches of ML, including rule-based, tree-based, support vector machine, lazy learning-based, and neural network techniques covering various learning algorithms.

D. Online Deployment

As presented in Figure 2, we deploy the Flush+Reload on the same host machine as victim models and send the observations to the remote attacker's classifier to extrapolate the labels of victim models' inputs. The detailed steps are listed below:

- **Step 1:** Launch Flush+Reload on the same host machine as victim models and collect traces of function calls once the victim model is started.
- **Step 2:** Preprocess the data collected online with the same preprocessing approach (III-C1) as the one in preparation.
- **Step 3:** Send processed data to the attacker's classifier for deducing the inputs' label information of the victim model. The deduced label information can be further leveraged by other malicious activities, like stealing investment decisions, as discussed in Section II-C.

IV. Evaluation

A. Experiment Setup

1) Attack Platform: All evaluations are done on a Dell server with 32 cores Intel(R) Xeon(R) CPU E5-2683 v4 and a three-level cache system. L3 cache memory is inclusive and shared among all cores meaning that flushing out the data in the last-level cache could remove the data in the L1 cache. The inclusiveness of the L3 cache creates a potential vulnerability surface for last-level cache attacks to be exploited. Our proposed inference attack should be effective on other platforms which are also vulnerable to Flush+Reload platforms.

2) Victim Models for Evaluation: As discussed in Section II, we select three prevalent deep learning model families, each with two variants as victims: DenseNet 121, DenseNet 169, VGG 16, VGG 19, MobileNet v1, and MobileNet v2 with implementation in Tensorflow 1.10.0 and Keras with the default weights, "ImageNet."

3) Datasets: Flowers dataset [1] is used in this work for evaluating the success rate of the proposed attack. Since we consider two attacking scenarios as detailed in Section III-A which demand no training and retraining, respectively, we split the dataset with two approaches. For setting A (no training), all data are split into 10%-90% to build the attacker's classifier and evaluate the attack online. For setting B (retraining), we split the dataset into 50%-10%-40% for retraining the last layer of the victim models, building the attacker's classifier, and testing the proposed attack.

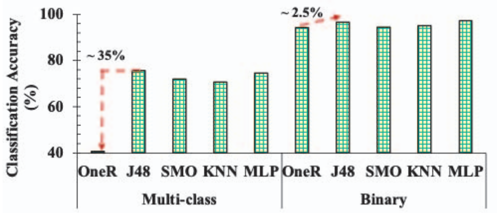


Fig. 3: Classification accuracy for binary and multi-class with various classifiers with the attack on MobileNet v1

B. Attacking Success Rate with Various Classifiers

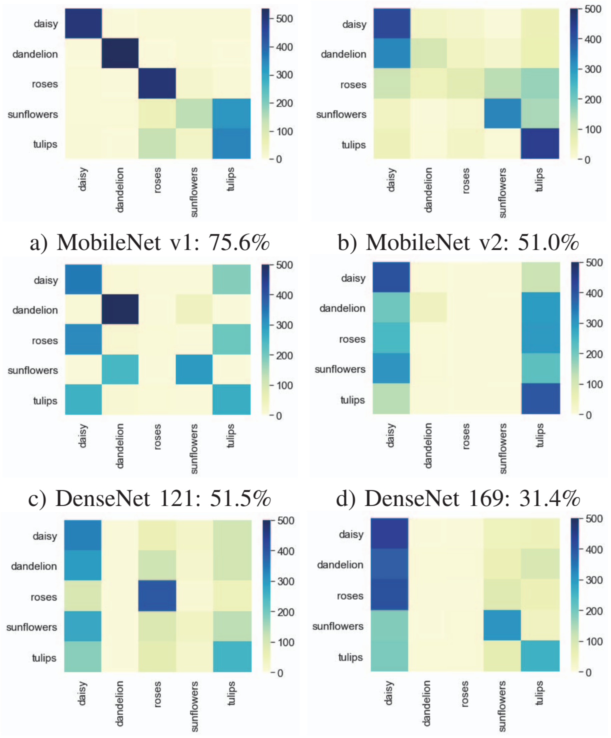
We explore different classification algorithms to select the optimal one for deducing label information from the victim model based on Flush+Reload observations. As shown in Figure 3, we present the binary and multi-class attacking success rate, i.e., classification accuracy, of our attack on MobileNet v1. Generally, multi-class attacking is more complicated than the binary attacking scenario with around a 20% accuracy difference. Selecting an appropriate classifier for multi-class attacking is more critical for our attack since OneR yields 35% less accuracy than J48 with 75% multi-class classification accuracy. By comparison, classification accuracy is less critical for binary attacking scenarios, while J48 is better than OneR by around 2.5%. For both binary and multi-class attacking scenarios, J48 and MLP outperform the rest three classifiers.

C. Binary Attacking Evaluation

As seen in Table III, we consider the classification accuracy with six different victim models under noise removal thresholds ranging from 0 to 500 cycles under both setting A and setting B detailed in Section III-A. We observe that similar results are found for setting A and setting B, indicating that introducing customized layers with pre-trained weights freeze does not cause an obvious impact on our attack. For all six victim models, we observe a noticeable influence of changing noise removal threshold on classification accuracy under setting A and B while the impact varies from models. We can observe that the attacking success rate on both MobileNet v1 and MobileNet v2 receive the highest value when the threshold is set at 100 cycles. By comparison, the static neural network models, DenseNet and VGG, demand a higher threshold value at 500 cycles. The difference is that the filter shape of MobileNet convolutions is generally smaller than VGG and DenseNet, having less execution time. Across all six models from three DNN families, MobileNet v1 is the most vulnerable one and suffers over 99% attacking success rate when the noise removal threshold is set at 100 cycles. Though DenseNet and VGG are less vulnerable, we still observe 78.5% and 70.6% success rates under setting A and similar results under setting B, indicating the attacker can still devise the label information from the victim static neural network models. Another observation is that static models from the same

TABLE III: Binary attacking success rate: sensitive labels (daisy) vs others (dandelion, roses, sunflowers, and tulips)

Settings	Threshold (Cycles)	MobileNet v1	MobileNet v2	DenseNet 121	DenseNet 169	VGG16	VGG 19
Setting A	0	94.3	69.4	73.9	44.9	58.8	56.9
	100	**99.0**	**77.3**	73.7	52.4	60.5	65.5
	200	98.8	66.3	73.6	51.6	60.5	68.7
	500	96.1	65.6	**78.5**	**66.6**	**70.6**	**69.9**
Setting B	0	93.9	69.0	73.0	52.2	58.5	56.2
	100	98.3	**76.9**	73.6	44.9	60.4	65.3
	200	**98.4**	66.1	73.0	50.7	60.3	69.4
	500	95.5	65.4	**78.3**	**66.4**	**69.7**	**69.8**



a) MobileNet v1: 75.6%

b) MobileNet v2: 51.0%

c) DenseNet 121: 51.5%

d) DenseNet 169: 31.4%

e) VGG 16: 37.7%

f) VGG 19: 37.4%

Fig. 4: Heatmap of multi-class attacking success rate under setting A of six models: MobileNet v1, MobileNet v2, DenseNet 121, DenseNet 169, VGG 19, and VGG 16.

DNN family, i.e., DenseNet and VGG, experience similar attacking success rates with various threshold cycles. Though MobileNet v2 are from the same DNN family as MobileNet v1, the attacking success rate is 77.3% because its short cuts between bottlenecks and depthwise causes the occurrences of *LaunchConv* function challenging for attackers' classifier. In conclusion, binary attacking can receive up to around 99% success rate and yield a significant difference in success rate on the six victim DNN models.

D. Multi-class Attacking Evaluation

Besides the binary classification that helps our attacker identify sensitive labels from victim models' execution, we also evaluate the attack to steal all possible labels individually from victim models. We plot the predicted labels from our attacker's classification results and ground truth labels with heatmaps as shown in Figure 4 for setting A with accuracy included. Similar to binary classification, our attack yields similar accuracy under the two settings, while results for setting B are not presented for the brevity of space. Compared to the binary attacking, inferring the multiple labels, i.e., five in this work, is more challenging for the attack, suffering over 20% attacking success rate decrease for almost all victim models. Still, the multi-class attacking on MobileNet v1 also receives a 75.6% success rate. Most classification errors are from sunflowers and tulips, meaning the attack receives high confidence for the other three classes samples. Similar to the binary attacking scenario, MobileNet v2 is less vulnerable since the occurrences of *LaunchConv* function are less separable caused by shortcuts and depthwise. Still, we find that most samples from *daisy*, sunflowers, and tulips are classified correctly by the attacker's classifier. By comparison, the four static DNN models, i.e., DenseNet 121, DenseNet 169, VGG 16, and VGG 19, are less likely to suffer from multi-class attacking.

V. Related Work

Privacy issues in DNNs have raised increasing attention from both industry and academia. Hong et al. [10], [11] exploit the cache-based SCAs, Flush+Reload, to steal information during the inference phase to reconstruct the crucial architectures of DNNs. It proposes an algorithm that generates candidate computational graphs from Flush+Reload observations, and the parameter estimation process removes incompatible candidates with 0% error for MalConv and ProxylessNAS-CPU. Yan et al. [30] take advantage of the DNNs' reliance on Generalized Matrix Multiply (GEMM) and employs Prime+Probe and Flush+Reload to obtain DNNs' architectures. It reduces the search space from 5.4×1012 to 16 for VGG and 6×1046 to 512 for ResNet-50, respectively. Hua et al. [13] investigate the leakage from memory and side channels on hardware accelerators even with secure techniques. They find that the memory access patterns can enable reverse-engineering of the structures and weights of CNN models.

Xiang et al. [29] illustrate an attack that extracts power traces on FPGA-based accelerators to reconstruct the input image. To achieve this, it filters out noises and distortion in power measurement with low-pass filters, power, and curve fitting. The capability of the attack is evaluated in the hand-written digits of the MNIST dataset [17], achieving 89% accuracy. Dong et al. [6] measure the execution time of floating-point multiplications from the power consumption traces, which are used to infer the pixel values of images without the knowledge of neural network' parameters. It shows 96.2% accuracy for the MNIST dataset. Luo et al. [20] show that using the cache-based SCA, Prime+Probe, to extract the cache access patterns can help to reveal the route or the location of a vehicle with the adaptive Monte-Carlo localization (AMCL) algorithm. It builds the correlation between the cache access pattern observed by Prime+Probe and the label information with statistical learning models.

VI. Conclusion

The progress made in DNNs boosts its application in various sensitive domains, including financial decisions, disease diagnosis, surveillance, etc., making the label information stealing attractive for attackers. This work demonstrates a stealthy attack that leverages a cache-based SCA, Flush+Reload, to spy on DNN models' computations and deduce the label information. To achieve it, we build an effective classifier that predicts the label information with Flush+Reload traces by investigating different noise removal settings and exploring a broad range of classification techniques. Additionally, we illustrate both binary attacking and multi-class attacking capability with six DNN models from both static and dynamic neural networks. Our experiments exhibit that the attack achieves up to 99% binary attacking success rate and 75.6% multi-class attacking success rate on MobileNet v1.

Acknowledgment

This research was funded in part by the Robert N. Noyce Trust.

References

[1] Flower dataset. In *https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz*.

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] Rich Caruana and Dayne Freitag. Greedy attribute selection. In *Machine Learning Proceedings 1994*, pages 28–36. Elsevier, 1994.

[4] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Gaofeng Dong, Ping Wang, Ping Chen, Ruizhe Gu, and Honggang Hu. Floating-point multiplication timing attack on deep neural network. In *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 155–161. IEEE, 2019.

[7] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, 2014.

[8] Shayan Hassantabar, Mohsen Ahmadi, and Abbas Sharifi. Diagnosis and detection of infected tissue of covid-19 patients based on lung x-ray image using convolutional neural network approaches. *Chaos, Solitons & Fractals*, 140:110170, 2020.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Sanghyun Hong, Michael Davinroy, Yiğitcan Kaya, Dana Dachman-Soled, and Tudor Dumitraş. How to 0wn nas in your spare time. *arXiv preprint arXiv:2002.06776*, 2020.

[11] Sanghyun Hong and et al. Security analysis of deep neural networks operating in the presence of cache side-channel attacks. 2018.

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[13] Weizhe Hua and et al. Reverse engineering convolutional neural networks through side-channel information leaks. In *2018 55th DAC*, pages 1–6. IEEE, 2018.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[15] Gyeeun Jeong and Ha Young Kim. Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, 117:125–138, 2019.

[16] Cheng-Bin Jin and et al. Real-time human action recognition using cnn over temporal images for static video surveillance cameras. In *Pacific Rim Conference on Multimedia*. Springer, 2015.

[17] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

[18] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B Lee. Last-level cache side-channel attacks are practical. In *2015 IEEE symposium on S&P*, 2015.

[19] Yuntao Liu and Ankur Srivastava. Ganred: Gan-based reverse engineering of dnns via cache side-channel. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 41–52, 2020.

[20] Mulong Luo, Andrew C Myers, and G Edward Suh. Stealthy tracking of autonomous vehicles with cache side channels. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 859–876, 2020.

[21] Antônio H Ribeiro, Manoel Horta Ribeiro, Gabriela MM Paixão, Derick M Oliveira, Paulo R Gomes, Jéssica A Canazart, Milton PS Ferreira, Carl R Andersson, Peter W Macfarlane, Wagner Meira Jr, et al. Automatic diagnosis of the 12-lead ecg using a deep neural network. *Nature communications*, 11(1):1–9, 2020.

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[23] Joshua Saxe and Konstantin Berlin. Deep neural network based malware detection using two dimensional binary program features. In *2015 10th MALWAR)*, pages 11–20. IEEE, 2015.

[24] Mohit K Sharma, Alessio Zappone, Mérouane Debbah, and Mohamad Assaad. Deep learning based online power control for large energy harvesting networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8429–8433. IEEE, 2019.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] TIMOTHYB.LEE. Tesla's autonomy event:impressive progress with an unrealistic timeline. 2019.

[27] Han Wang, Hossein Sayadi, Setareh Rafatirad, Avesta Sasan, and Houman Homayoun. Scarf: Detecting side-channel attacks at real-time using low-level hardware features. In *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pages 1–6. IEEE, 2020.

[28] Han Wang, Hossein Sayadi, Avesta Sasan, Setareh Rafatirad, and Houman Homayoun. Hybridg: Hybrid dynamic time warping and gaussian distribution model for detecting emerging zero-day microarchitectural side-channel attacks. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 604–611. IEEE, 2020.

[29] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. I know what you see: Power side-channel attack on convolutional neural network accelerators. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 393–406, 2018.

[30] Mengjia Yan and et al. Cache telepathy: Leveraging shared resource attacks to learn dnn architectures. In *29th Security Symposium USENIX Security 20)*, pages 2003–2020, 2020.

[31] Yuval Yarom and Katrina Falkner. Flush+ reload: A high resolution, low noise, l3 cache side-channel attack. In *USENIX Security Symposium*, volume 1, pages 22–25, 2014.

[32] Liang Zhou and et al. Cyber-attack classification in smart grid via deep neural network. In *Proceedings of the 2nd International Conference on Computer Science and Application Engineering*, pages 1–5, 2018.