# Towards Stealing Deep Neural Networks on Mobile Devices

Shashank Reddy Danda(✉), Xiaoyong Yuan, and Bo Chen

Michigan Technological University, Houghton, MI 49931, USA
{sdanda,xyyuan,bchen}@mtu.edu

**Abstract.** Recently, deep neural networks (DNN) are increasingly deployed on mobile computing devices. Compared to the traditional cloud-based DNN services, the on-device DNN provides immediate responses without relying on network availability or bandwidth and can boost security and privacy by preventing users' data from transferring over the untrusted communication channels or cloud servers. However, deploying DNN models on the mobile devices introduces new attack vectors on the models. Previous studies have shown that the DNN models are prone to model stealing attacks in the cloud setting, by which the attackers can steal the DNN models accurately. In this work, for the first time, we study the model stealing attacks on the deep neural networks running in the mobile devices, by interacting with mobile applications. Our experimental results on various datasets confirm the feasibility of stealing DNN models in mobile devices with high accuracy and small overhead.

**Keywords:** Deep neural network · Model stealing · Privacy · Mobile devices

## 1 Introduction

Deep neural networks (DNNs) have been pervasively used in various computer vision tasks, such as objects detection, image classification, and face recognition. Due to the high business value of DNNs, model stealing attacks have been conducted in recent years, in which an attacker attempts to duplicate a victim machine learning/neural network model by querying the victim model and training a surrogate model to replicate the victim model's functionality. Such a novel attack has become a serious security threat to many machine learning services today, as the machine learning models represent the business advantages of service providers and training a model from scratch is costly and time-consuming.

Tramèr et al. [9] successfully stole machine learning models, including decision tree, support vector machine, and regression models, with an accuracy of around 100%. A line of recent works has shown the feasibility of stealing more complex yet powerful machine learning models like DNNs [3,8,10,13–15]. However, most existing model stealing attacks are conducted on public cloud services, where the DNN models are run on the cloud servers [2,11].


J. Garcia-Alfaro et al. (Eds.): SecureComm 2021, LNICST 399, pp. 495–508, 2021.
https://doi.org/10.1007/978-3-030-90022-9_27

Nowadays, with the increasing computing power and the recent advance of light-weighted neural networks, DNNs have been increasingly deployed on mobile computing devices. By leveraging open-source deep learning frameworks (e.g., TensorFlow Lite and PyTorch Mobile), the developers can smoothly integrate well-trained DNN models with mobile applications under minimal effort. Compared with the cloud-based DNN services, this type of on-device DNN can provide immediate responses without relying on network availability or bandwidth, and boost security and privacy as users' data do not need to be transferred over untrusted communication channels or cloud servers. The question is, are the model stealing attacks feasible on the on-device DNN? Mobile devices today suffer from malware attacks. By performing various attacks like phishing or social engineering, the malware may get installed in a victim's mobile device. Thanks to the built-in isolation of the mobile applications (e.g., the Android application sandbox), the malware cannot directly have access to the DNN model of a victim mobile application (even if the malware can obtain a higher system privilege, gaining access to the DNN model, the model may have been stored encrypted). However, by interacting with the mobile application, the malware may conduct a model stealing attack similar to that in a cloud setting.

In this work, we aim to confirm the feasibility of the model stealing attacks on mobile devices. This task is non-trivial as the mobile devices are equipped with much less memory and computational power compared to the cloud servers and the malware which performs the model stealing attacks should minimize its consumption of computation and memory to avoid being noticed by the device's owner. We have proposed a model stealing approach for mobile devices by performing interactions with the victim's mobile applications running DNN models (i.e., *the victim models*). We have conducted the model stealing experiments based on several popular machine learning applications and datasets under one of the following two practical assumptions: 1) the attackers have access to the training data, and 2) the attackers do not have access to the training data but can use some public datasets. Our experimental results confirm that the model stealing attacks are feasible on mobile devices with high accuracy (i.e., the accuracy obtained by the attackers is close to that of their victim models) and high efficiency (i.e., the computational cost on the mobile devices is small). Our major contributions are summarized below:

- We have proposed a model stealing approach on mobile devices by interacting with a victim's mobile device.
- We have successfully extracted a stolen model with an accuracy similar to that of the victim model.
- We have investigated several system metrics of mobile devices during conducting the model stealing attacks to verify the feasibility of model stealing in mobile devices.
- We have conducted a rigorous evaluation of model stealing attacks on several machine learning applications and datasets under two practical assumptions.

## 2 Related Work

CNN networks can be easily stolen or replicated using several different factors which include creating a synthetic dataset and training a model with this synthetic dataset which gives an almost equal performance with the victim model as stated in [1]. They can obtain good accuracy using cloud-based APIs and performing evaluation using facial expression, object, and crosswalk classification datasets. The experiments conducted by them show that model stealing can be performed using cloud-based APIs by querying a target network. Several other methods such as ES attacks were proposed for model stealing the models without any data hurdles as shown in [10]. These attacks train the models using several iterations and obtain a copy of the victim model. Several image datasets were used and the models built were stolen with some less accuracy rate compared to the victim models' accuracy. Tramèr et al. [9] showed an extraction of some of the models like logistic regression, decision trees, and other models. These models are demonstrated against online services as BigML and Amazon machine learning. Their results show how the machine learning models should be deployed securely and the countermeasures of new model extraction.

Cloud-based model stealing was performed in [15] using DNN models by reducing the number of queries required to steal the target model. The accuracy achieved in their work was also good enough to prove that model stealing is successfully performed. A data-free model stealing attack which is called MAZE was proposed in [4] which does not require any type of data for model stealing. It uses some generative models and creates some synthetic data which helps for stealing the models. Their method depends on zeroth-order gradient estimation to perform optimization and finally gives a stolen model with a high accuracy. The proposed method was able to give better accuracy which provides a normalized clone accuracy. Hyperparameters which are very crucial in building a machine learning model can also be stolen as stated in [12]. Those hyperparameters store confidential information of the model and they were able to successfully extract the hyperparameters for different machine models which include ridge regression, support vector machine, logistic regression and neural network models. Orekondy in [13] stated that a lot of machine learning models were deployed on several platforms. They proposed a model stealing functionality of black box model which is known as Knockoff Nets. They validated the model stealing using several datasets with the help of image analysis APIs which provided some good model stealing results. Exploratory based machine learning attack was proposed in [7] on several deep learning models to infer them as a black box and the results obtained after extraction are quite similar to that of the victim model.

Kariyappa in [5] proposed a methodology called "adaptive misinformation" to avoid model stealing. They have conducted experiments by performing several queries on the target model and generated a dataset that can be used for training the stolen model. A good defense mechanism was shown by comparing several other existing defenses in terms of security and accuracy of the models. Matthew Jagielski in [14] proposed some extraction attacks which can steal some remotely

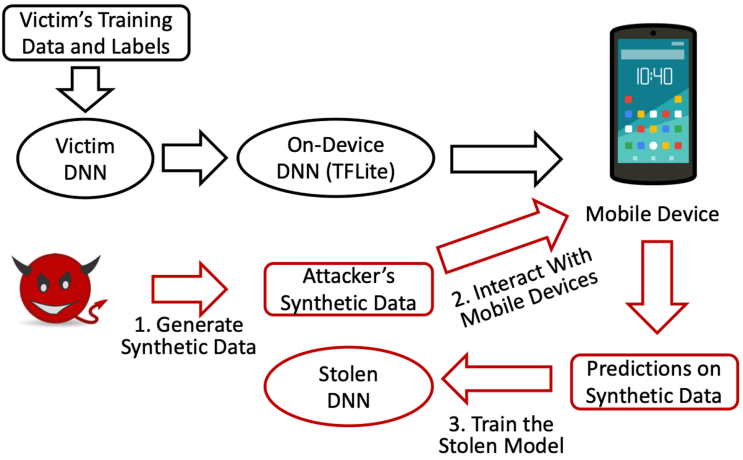


**Fig. 1.** The workflow of the proposed model stealing approach on mobile devices.

deployed models where they used billions of images to perform model stealing and the performance of the stolen model is evaluated in terms of accuracy and fidelity.

## 3 A Model Stealing Approach on Mobile Devices

In this section, we propose a model stealing approach for mobile devices, by which the mobile malware can steal the DNN model deployed on a victim mobile application. We prepare for a well-trained DNN model as a victim model, convert it to tflite format, and deploy it on a mobile device. The proposed model stealing attack consists of three main steps: (1) generating synthetic data, (2) interacting with the victim mobile device and getting predictions on the synthetic from the victim model, and (3) transferring the knowledge to the stolen DNN model by training it on the synthetic data and obtained predictions. The workflow of our model stealing is illustrated in Fig. 1. We then train a stolen DNN model based on the synthetic data and obtained predictions using CopyCat [1].

### 3.1 Generating Synthetic Datasets

We have considered two threat models in generating synthetic datasets. In the first threat model, we assume that the attack has access to the victim model's training data or its data distribution, namely problem domain (PD). Therefore, the attacker can extract the synthetic data from the original dataset on which the victim model is trained. In the second threat model, we consider a more practical scenario, where the attacker has no knowledge of the victim model's training data and only has access to public datasets, namely non-problem domain (NPD). There is no overlap between PD and NPD. Moreover, the data generated

in the NPD dataset belong to different categories from the problem domain. We will discuss the details of NPD datasets in Sect. 4. Note that we store the synthetic data in the mobile device and generate labels in a separate file by running interactions using the data records from the dataset.

### 3.2 Interacting with Mobile Devices

Second, the attacker interacts with mobile devices by querying the synthetic data and aims to get the predictions on synthetic data from the victim model. Given the constrained computational capabilities on the mobile device, the attacker controls the number of interactions using different numbers of samples from the dataset rather than the entire dataset. By interacting with the DNN model deployed on the mobile device, the attacker obtains the corresponding predictions and stores them in a different folder from the synthetic data.

### 3.3 Training the Stolen Model

The attacker uses the synthetic data and predictions from the mobile device to train a stolen model. We follow the models stealing approach proposed in CopyCat CNN [1] We assume that the attacker does not know about the target model's neural network architecture, but the different neural networks do not affect the effectiveness in transferring the knowledge to a different model, which has been shown in many recent works. Therefore, the attacker can set up a widely used model and train it on the synthetic data and obtained predictions. Moreover, previous works have shown that using pre-trained models would be a very good start for performing model stealing, which saves the attacker much time in training the DNN models. In addition, a large number of pre-trained models are publicly available nowadays, which can be well utilized by the attacker. Specifically, the attacker can first update the output layer to match the required machine learning tasks (e.g., in the image classification task, the output dimension is changed to the number of classes). Then the attacker can freeze some parameters of the pre-trained models and train the rest of them on the synthetic data to fit the problem domain.

## 4 Experimental Methodology

This section shows the creation of copycat models from the target network models and a proper comparison is made with the victim model's performance which is evaluated using the test data. In this work, we have worked on several different models on different types of datasets which include MNIST, CIFAR-10 and ImageNet. All the copycat models which include the model created using the PD dataset, NPD are evaluated using the different datasets e.g., MNIST, CIFAR-10, and ImageNet. These are the most widely used datasets for image classification and the copycat models are evaluated in three real-world problems. The models that we have used include LeNet-5, ResNet-18, and MobileNetV2

models. All these models were able to give good accuracies after generating the copycat models. The reason why we have chosen these datasets where the sizes of the images were different and this made us to justify that the interactions can be performed using Android applications for different sizes of images.

### 4.1 Test Dataset and Standard Methods

In all the experiments, we used the same test dataset for evaluating all the standard and copycat models' performance. The standard methods include finding the performance of the victim model, i.e., the model trained on the original dataset and evaluated on the remaining amount of the data. There is also an alternate standard method where we can also take some less amount of data from the same problem domain with the original labels and perform evaluations using the test data. So, here we have only chosen the victim model accuracy for performing comparisons. The test data and the standard method are described below:

**Test Dataset.** For all the copycat and the victim models, some part of the dataset which does not belongs to the training dataset is divided as the test data set. All these stolen models were also evaluated using this test dataset and the performance is compared between all the models. Attackers have no access to this data.

**Model Trained with Original Domain.** This model acts as the victim model trained on the original dataset and is not accessible by anyone. In our experiments, we have evaluated whether this model can be stolen from mobile devices by performing interactions across the devices. Here we have considered three trained models. The LeNet-5 model is trained on the MNIST dataset, ResNet-18 trained on the CIFAR-10 dataset, and the MobileNetV2 model trained on the ImageNet dataset. All these models are trained and evaluated on the testing data and finally converted to TensorFlow lite format which can be easily deployed on mobile devices. The model cannot be accessed by everyone and the attacker can only generate the output labels using these models.

### 4.2 Copycat Attacks

We have conducted the attacks one of the following: 1) the model trained with the images from the same problem domain but stolen predictions, and 2) the model with non-problem domain images which are completely different from that of the original images and the labels are generated using the target model.

**Model Trained From the Same Problem Domain and Stolen Predictions.** In this case, the images are from the same problem domain and labels are stolen using the victim model. The images are completely different from that

of the images used for training the victim model. We have considered training the model on three different datasets. At first, the MNIST handwritten images were used and here we have considered a total of 10,000 images. Several queries were performed with different sizes starting from 250 to 5,000 and evaluated the performance of the stolen model with these different sizes. Finally, the number of interactions which gave better performance is selected and performed these interactions using the mobile application. Using this we can test the performance of the Android application and can state whether the attacker can perform these interactions and steal the model. Similarly, we have also conducted experiments using the CIFAR-10 dataset by considering 10,000 images and passing several queries. Finally, for the ImageNet dataset since the dataset is very huge, we are able to collect 100,000 images and performed up to 100,000 queries. We finally evaluated the performance using 10,000 samples and like that of MNIST, we deployed the model on a mobile application and evaluated the performance of the application.

**Model Trained from Non-problem Domain and Stolen Predictions.** Compared to PD, this NPD possess images which are completely different from the problem domain. These images possess some labels which are not considered, and the new labels are generated using the victim model which are called stolen predictions. We have considered training the model on three different datasets. At first, the MNIST Fashion and KMNIST images were used and here we have considered a total of 50,000 images. Several queries were performed with different sizes starting from 250 to 50,000 and evaluated the performance of the stolen model with these different sizes. Finally, the number of interactions which gave better performance is selected and performed these interactions using the mobile application. Using this we can test the performance of the Android application and can state whether the attacker can perform these interactions and steal the model. Similarly, we have also conducted experiments using the CIFAR-100 dataset by considering 50,000 images and passing several queries. Finally, for the ImageNet dataset since the dataset is very huge, we have chosen Stanford car dataset and ISIC skin cancer images. We are able to collect 100,000 images and performed queries up to 100,000. We finally evaluated the performance using 10,000 samples and like that of MNIST, we deployed the model on mobile application and evaluated the performance of the application. Here the stolen predictions are different from the originals, which shows some different distributions of classes for each type.

## 5 Experimental Results

We have performed model stealing attacks in mobile devices, by stealing copycat CNN models using a few different datasets. Using the adversarial robustness toolbox, we generate a class for the random data and use this dataset for training the duplicate model [1,6]. For each dataset we have performed several experiments and the copycat models are evaluated under PD and NPD conditions.

Here in all these cases, the stolen predictions are used. The labels are obtained by running interactions on mobile devices where we have deployed our victim models. The data stolen using the victim model deployed in our mobile device is used as the final data for performing stealing. The models are evaluated, and the performance of each model is compared with the victim model's performance for each dataset.

### 5.1 General Setup

The MNIST and CIFAR-10 datasets were directly loaded from TensorFlow and pre-processed. We obtained the ImageNet from its official website. We trained the LeNet-5 and ResNet-18 model but we used MobileNetV2 pretrained model for the ImageNet dataset. Some image augmentation techniques were also used. All the models which we trained use SGD as the optimizer. For each model we have used different epoch numbers, i.e., 5 for MNIST, 50 for CIFAR-10, and ImageNet datasets. We have built an Android application using Android studio and we ran the application successfully on Samsung S10 plus device with the latest Android 11 OS, 8 GB memory, and 128 GB internal storage. The application has TensorFlow lite models deployed in it. Interactions with the Android application were performed on the mobile device and, we accessed the CPU and memory usage when running the interactions. To train the copycat model, we used an Intel(R) Core(TM) i7-3770 3.40 GHz with 32GiB of RAM and NVIDIA GeForce GTX 1070 with 8GiB of memory, managed with Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-118-generic x86-64) with NVIDIA CUDA Framework 11.0 and cuDNN 10.2.

**Table 1. Accuracy for the stolen models vs. victim model**

| Dataset | Model architecture | Accuracy | | |
|---|---|---|---|---|
| | | Victim model | PD | NPD |
| MNIST | LeNet-5 | 97–98% | 95% | 93–94% |
| CIFAR-10 | ResNet-18 | 92% | 90–91% | 80–81% |
| ImageNet | MobileNetV2 | 71.3% | 81.2% | 73.5% |

### 5.2 Attacks on the MNIST Dataset

The MNIST handwritten image dataset is widely used for image classification and the images here are of 28×28 pixels. The total classes here are 10 and the dataset comprises of 60000 images. The model which we have used here is the LeNet-5 CNN model and is trained on 50,000 samples. The victim model is evaluated using the remaining 10,000 samples. For performing model stealing, under PD we have used these 10,000 test images for performing training and evaluation. Coming to NPD, we have used MNIST fashion and KMNIST images

for generating synthetic dataset,s and the labels are stolen using the victim model. The final evaluation is done using the above-mentioned 10,000 images from the original test dataset. The results obtained are shown below: Here PD stands for the images collected from the same dataset. NPD stands for non-problem domain data collected from different sources (MNIST fashion images and KMNIST images). These duplicate images and the original dataset images were collected using the original trained model by running interactions on mobile devices. In all the cases except PD a total of 50,000 images were collected and used for training the model. Different numbers of queries were performed which are of different lengths. These range from 250–50,000 for the NPD case and 250–5000 for PD case were submitted to the victim classifier which is deployed on a mobile device to steal it. In all cases, the more the number of queries, the more accuracy is. For PD we do not need more samples since the dataset is from the original domain itself but with stolen predictions.

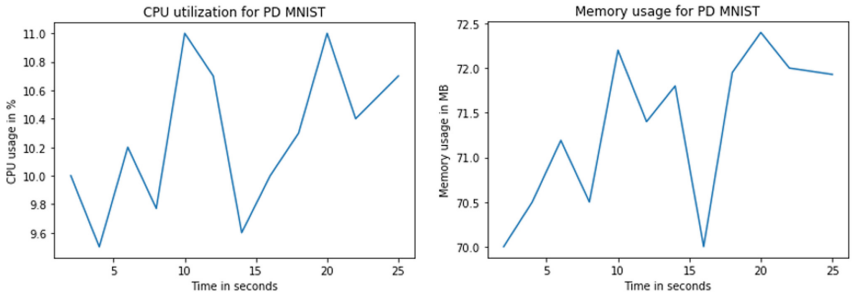


**Fig. 2.** CPU and memory usage after performing interactions in PD (MNIST).

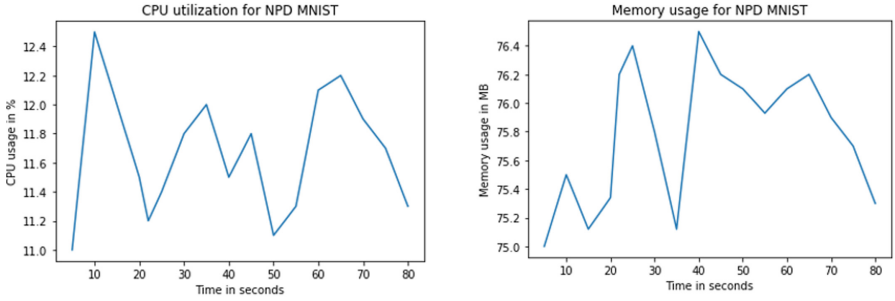


**Fig. 3.** CPU and memory usage after performing interactions in NPD (MNIST).

Finally, the copycat model's accuracy for PD dataset is about 95% which is very close to that of the victim models' accuracy which is 97–98% (Table 1). The accuracy of NPD model is between 93–94% which is somewhat low compared to the victim model (Table 1).

Figure 2 and 3 show the performance of the mobile device after running interactions. For the problem domain, running 5000 interactions took a short time to generate the labels. The CPU utilization rate was around 10–11% and the memory occupied was around 72–73 MB. Similarly, the best number of interactions for NPD was 50000 which took around 80 s to run interactions. The CPU utilization rate was around 12–13% and the memory occupied was around 76–77 MB. The CPU utilization rate and memory consumed show that model stealing is feasible using the MNIST dataset.

### 5.3 Attacks on the CIFAR-10 Dataset

The CIFAR-10 dataset is also used for image classification and the images here are of 32×32 pixels. The total classes here are 10 and the entire dataset comprises of 60,000 images. The model which we have used here is ResNet-18 CNN model and is trained on 50,000 samples. The victim model is evaluated using the remaining 10,000 samples. For performing model stealing, under PD we have used these 10,000 test images for performing training and evaluation. Coming to NPD, we have used the images from the CIFAR-100 dataset for generating synthetic dataset and the labels are stolen using the victim model. The final evaluation is conducted using the above-mentioned 10,000 images from the original test dataset.

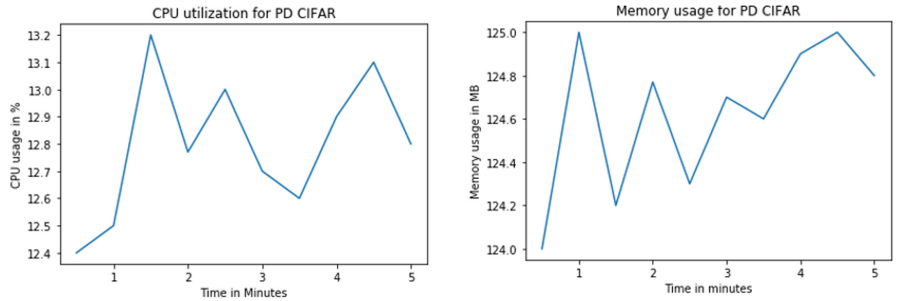


**Fig. 4.** CPU and memory usage after performing interactions in PD (CIFAR-10).

PD stands for the images collected from the same dataset. NPD stands for non-problem domain data collected from different sources (the CIFAR-100 dataset). These duplicate images and the original dataset images were collected using the original trained model by running interactions on mobile devices. In all the cases except PD, a total of 50,000 images were collected and used for training the model. Different numbers of queries were performed which are of different lengths. These range from 250–50,000 for the NPD and 250–5000 for PD case were submitted to the victim classifier on mobile device to steal it. For PD we do not need more samples since the dataset is from the original domain itself but with stolen predictions. Finally, Table 1 shows the copycat model's

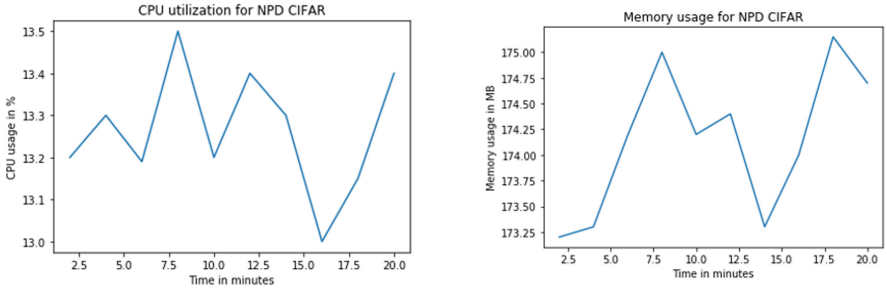


**Fig. 5.** CPU and memory usage after performing interactions in NPD (CIFAR-10).

accuracy for the PD dataset is about 90–91% which is very close to that of the victim models' accuracy which is 92%. The accuracy of the NPD model is very low which is 80–81% (Table 1).

Figure 4 and 5 show the performance of the mobile devices after running the interactions. For the problem domain, running 5,000 interactions took very less time i.e., 5 min to generate the labels compared to NPD. The CPU utilization rate was around 13–14% and the memory occupied was around 124–125 MB. Similarly, the best number of interactions for NPD was 50,000 which took around 20 min to run interactions. The CPU utilization rate was around 14% and the memory occupied was around 176 MB. The CPU utilization rate and memory consumed show that model stealing is feasible on the CIFAR dataset.

### 5.4 Attacks on the ImageNet Dataset

The ImageNet dataset is the most popularly used image dataset and several pretrained models are present which are trained on this dataset. This dataset is very huge and comprises of 1.2 million training samples and evaluated using 100,000 test samples. The pretrained models on this dataset are widely used in several other image classification problems by performing transfer learning. In our experiment, we have chosen a pretrained MobileNetV2 model for image classification and the images here are of 224×224 pixels. The total number of classes here are 1,000 which consists of several different categories. The model which we have used here is the MobileNetV2 model and is trained on 1.2 million samples. The victim model is evaluated using 100,000 samples. For performing model stealing, under PD we have used 100,000 test images for performing training and evaluation. Coming to NPD, we have used Stanford car dataset and ISIC skin cancer images for generating synthetic dataset,s and the predictions are stolen using the victim model. The final evaluation is done using 10,000 images from the original test dataset which are not used for training the copycat model.

In all the cases a total of 100,000 images were collected and used for training the model. Different numbers of queries were performed which are of different lengths. These range from 250–100,000 for the PD and NPD case were submitted to the victim model on a mobile device to steal it. Since the ImageNet dataset

is more complicated, compared with the MNIST and CIFAR-10 datasets, more data samples and queries were conducted to provide better accuracy. Finally, the copycat model's accuracy for PD dataset is about 81.2% which is higher than that of the victim models' accuracy which is 71.3% (Table 1). The accuracy of NPD model is about 73.5% (Table 1).

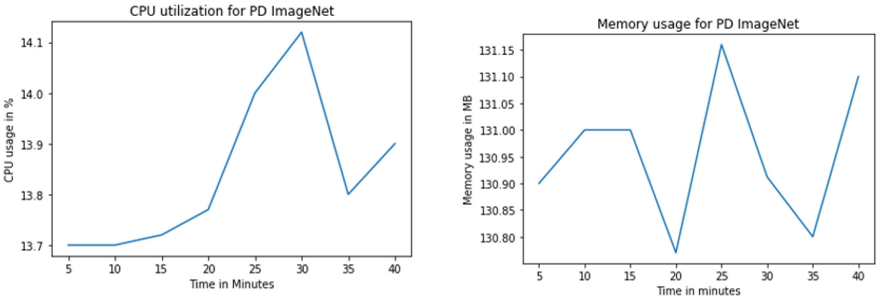


**Fig. 6.** CPU and memory usage after performing interactions in PD (ImageNet).

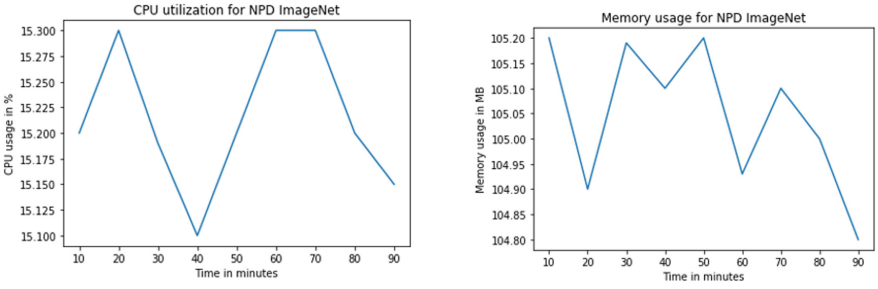


**Fig. 7.** CPU and memory usage after performing interactions in NPD (ImageNet).

Figure 6 and 7 show the performance of the mobile device after running the interactions. For the problem domain, running 100,000 interactions took 35–40 to generate the labels. The CPU utilization rate was around 14–15% and the memory occupied was around 131–132 MB. Similarly, the best number of interactions for NPD was also the same i.e., 100,000 which took around 80–90 to run interactions. The CPU utilization rate was around 15–16% and the memory occupied was around 100–105 MB. The CPU utilization rate and memory consumed show that model stealing is feasible using the ImageNet dataset.

## 6 Discussion

**Conducting the Model Stealing Attacks in Practice.** To conduct a model stealing attack, the malware can contact a remote server (e.g., a command-and-control server) to generate a synthetic dataset. The synthetic dataset can then be downloaded locally in the victim's mobile device for the interactions later. The final synthetic dataset (after labels are generated by interacting with the victim model, they will be incorporated into the synthetic dataset) will be sent to the remote server to train the copycat model. Therefore, the major computation in the victim device is interacting with the model to generate the labels, which is not very expensive as shown in our experiments.

**Automating the Interactions with Any Mobile Applications.** To make the model stealing attacks more practical, the malware should be able to interact with any victim mobile applications. Automating this process itself is a challenging problem considering the heterogeneity of interactions in different mobile applications. In our experiments, the interaction was specific for a self-built application (with a self-crafted DNN model). We will investigate this problem in our future work.

## 7 Conclusion

Nowadays, deep neural networks have been increasingly deployed on mobile computing devices. In this paper, we have identified the privacy risks of deep neural network models on mobile devices by performing model stealing attacks. We demonstrated the feasibility of model stealing attacks on mobile devices in terms of stolen models' prediction performance and model stealing efficiency. We have conducted experiments on several popular machine learning applications and datasets. Our experimental results have shown that the accuracies of stolen models are very close to that of the victim models. Moreover, we have shown that although interactions are required in model devices for model stealing, the computational cost of stealing models on mobile computing devices is small. We hope our work could shed light on future research on protecting on-device deep neural networks.

**Acknowledgment.** Bo Chen was supported by US National Science Foundation under grant number 1938130-CNS, 1928349-CNS, and 2043022-DGE. The work of Xiaoyong Yuan was supported in part by US National Science Foundation under SHF-2106754.

## References

1. Correia-Silva, J.R., Berriel, R.F., Badue, C., de Souza, A.F., Oliveira-Santos, T.: Copycat CNN: model stealing knowledge by persuading confession with random non-labeled data. In: 2018 International Joint Conference on Neural Networks (IJCNN) (2018)

2. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and Harnessing Adversarial Examples (2015). arXiv:1412.6572 [stat.ML]
3. Guo, S., Zhao, J., Li, X., Duan, J., Mu, D., Xiao, J.: A black-box attack method against machine-learning-based anomaly network flow detection models. Secur. Commun. Netw. **2021** (2021)
4. Kariyappa, S., Prakash, A., Qureshi, M.: MAZE: data-free model model stealing attack using zeroth-order gradient estimation (2020). arXiv:2005.03161
5. Kariyappa, S., Qureshi, M.K.: Defending against model stealing attacks with adaptive misinformation (2019). arXiv:1911.07100
6. Nicolae, M.I., et al.: Adversarial Robustness Toolbox v0.2.2. CoRRabs/1807.01069 (2018)
7. Shi, Y., Sagduyu, Y., Grushin, A.: How to steal a machine learning classifier with deep learning. In: 2017 IEEE International symposium on technologies for homeland security (HST). IEEE (2017)
8. Szegedy, C., et al.: Intriguing properties of neural networks (2014). arXiv:1312.6199
9. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Model stealing machine learning models via prediction APIs. In: 25th USENIX Security Symposium (USENIX Security), pp. 601–618. USENIX Association, Austin (2016)
10. Yuan, X., Ding, L., Zhang, L., Li, X., Wu, D.: ESAttack: model stealing against deep neural networks without data hurdles (2020). arXiv:2009.09560
11. Model Stealing. Accessed 15 May 2021, https://www.mlsecurity.ai/post/what-is-model-modelstealing-and-why-it-matters
12. Wang, B., Gong, N.Z.: Stealing hyperparameters in machine learning. In: IEEE Symposium on Security and Privacy (SP) (2018)
13. Orekondy, T., Schiele, B., Fritz, M.: Knockoff nets: stealing functionality of black-box models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019)
14. Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., Papernot, N.: High accuracy and high fidelity extraction of neural networks, PP. 1345–1362 (2020)
15. Yu, H., Yang, K., Zhang, T., Tsai, Y.Y., Ho, T.Y., Jin, Y.: CloudLeak: large-scale deep learning models stealing through adversarial examples. In: Network and Distributed System Security Symposium (2020)