Extraction Attacks on Machine Learning Black-Box Models

Kacem Khaled , Hadhemi Jebnoun , Mira Marhaba , Dan Judkiewicz
Polytechnique Montréal
{kacem.khaled, hadhemi.jebnoun, mira.marhaba, dan.judkiewicz}@polymtl.ca

Abstract

Deep Neural Networks have become more and more powerful and frequently used in the modern world. Despite this advancement, they are often deployed regardless of possible security threats. Researchers in Machine Learning have shown that Deep Neural Networks remain vulnerable to adversarial attacks. The functionality of the model can be stolen even if the adversarial has black-box access to the victim model (only the inputs and the outputs). In this study, we reproduce two stealing attacks : Papernot and KnockoffNets attacks. We have several major findings. On one hand, we empirically validate that those attacks consist of a serious threat to models through successfully reproducing them, and we demonstrate that a risk could be to craft transferable adversarial examples to fool a target black-box model, or the model itself could be a business advantage to its owner and his model's functionality is compromised. On the other hand, model stealing transfers across architectures and even attacking complex black-box CNNs (ResNet-34, VGG,..) is possible and the extracted model can learn to effectively classify images from unseen classes.

Keywords : Machine Learning Security, Model Stealing attack, Intellectual Property, Adversarial attack

1 Introduction

Recently, deep learning has been proven to be highly effective in several application domains including image recognition, speech recognition, computer games, and natural-language understanding. These major breakthroughs are credited to the advancement of deep neural networks (DNNs), as well as the availability of huge amounts of data and computing power. However, despite these impressive results, the research community has recently shown that DNNs remain vulnerable to adversarial attacks. These attacks can be executed in both white-box or black-box settings. In this study, we have investigated stealing functionality in black-box model attack scenarios. We therefore analyzed two related papers :

- *'Practical Black-Box Attacks against Machine Learning'* [Papernot *et al.*, 2017].
- *'Knockoff Nets : Stealing Functionality of Black-Box Models'* [Orekondy *et al.*, 2019]

It is of interest to know whether the results still hold true with the same architecture or dataset or with variations of these elements under different circumstances.[1]

In this report we give a short introduction about the background of this project and the related work. Then, we explain the methodology of both attacks. After that, we explain our experimental setups and we present our results. Finally, we finish with a critical discussion of our work and our approach to learn this subject.

2 Background

In this section, we present the threat model as well as some related works existing in the literature.

2.1 Threat Model

Model stealing attacks can be performed in both *black-box* or *white-box* attacks.

In a ***white-box*** attack (Fig.1-(a)), the attackers have a good knowledge of the target's model such as the parameters and the architecture. Thus, they can easily substitute a model similar to the victim's accuracy.

Whereas, in a ***black-box*** attack (Fig.1-(b)), the attacker has a limited access to the target model. It can only observe the inputs and outputs, but has no access to the internal working of the model.

2.2 Related work

This section reports about recent research works, around the scope of our work, dealing with stealing machine learning models *attacks*, *knowledge distillation* and *black box adversarial attacks*.

Black Box Adversarial Attacks aims to generate well crafted examples that could be misclassified by a target victim model [Goodfellow *et al.*, 2015]. This attack is easy to implement in white-box setting. However, in black-box settings, [Papernot *et al.*, 2017] suggests to conduct it by training

[1] Our github repository: https://github.com/KacemKhaled/Project-INF8225

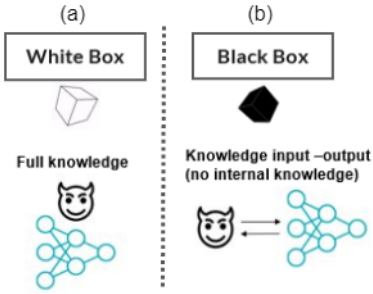

Figure 1: White-box vs black-box attack

a substitute model using synthetic data generation technique to the target (the oracle) in order to extract its functionality. Then, knowing the surrogate model, crafting adversarial examples to *fool* it is easier (white-box), and the crafted adversarial examples are transferable against the original black-box oracle. On of our focuses in this work is reproducing some of these techniques. More recent articles use techniques such as *reservoir sampling* to limit the number of queries in order not to be detected [Papernot *et al.*, 2016a] or even *Generative Adversarial Networks (GANs)* to bypass current defence methods in a black-box attack situation. ([Xiao *et al.*, 2019], [Hu and Tan, 2017])

Model stealing attacks, also known as extraction or reverse engineering aims to steal the parameters of a machine learning model. In such attacks, the main purpose of an adversary with black-box access, but no prior knowledge of a ML model's parameters or training data, is to duplicate the functionality of the model. Stealing models' parameters threatens the confidentiality of the learner's algorithm, and also gives an attacker the ability to perform evasion attacks or model inversion attacks subsequently [Tramèr *et al.*, 2016]. Lowd and Meek [Lowd and Meek, 2005] introduced efficient algorithms to extract model parameters of linear classifiers when the attacker can issue membership queries to the model through an API. Tramer et al. [Tramèr *et al.*, 2016] demonstrated that model parameters can be more accurately and efficiently extracted when the API also produces confidence scores for the class labels.

Knowledge Distillation uses a larger network to teach a smaller network (teacher - student) [Hinton *et al.*, 2015]. However, in this case, the attacker has a deep knowledge of the victim's black box (architecture, training data) while this study focuses on having weak assumptions about the black box, the focus of our work is stealing complex black box model functionality by a weaker adversary by replicating [Orekondy *et al.*, 2019] study.

3 Methodology

In this section, we explain the theory behind the Papernot attack as well as the KnockoffNets attack.

3.1 Papernot attack

The goal of [Papernot *et al.*, 2017] was to demonstrate that a black box adversarial attack was possible, and to craft adversarial examples that would be misclassified by the DNN without any information about the structure or access to the DNN model. The adversary has only knowledge of inputs and labels for a given number of chosen inputs. To be able to craft adversarial examples reliably, the approach the researchers used was to train a substitute DNN on the inputs and labels of the oracle DNN to emulate as closely as possible the behaviour of the oracle DNN and then to attack the substitute DNN. The attack of the substitute DNN is easier since the adversary has access to its inner model. The transferability property will lead the adversarial samples crafted on the substitute DNN to be misclassified by the targeted DNN as well. A hurdle to overcome is to make a small number of queries to the oracle DNN, since a large number of queries would make the attack easily detectable. The algorithm uses a heuristic to explore the input domain with a small number of queries.

- **Substitute Model Training Algorithm**
  1. **Input Collection** The adversary collects a minimal set of inputs that characterize the input domain. (for example one image of each digit 1 through 9 for MNIST [LeCun and Cortes, 2010]).
  2. **Architecture Setup** The adversary chooses a suitable architecture based on the type of input fed to the oracle DNN. Since the substitute will be trained to imitate the oracle, the exact architecture (number of layers, number of nodes) is not crucial, and can be selected based on previous experience. For example, if the inputs are images, a CNN would likely be the best choice.
  3. **Substitute Training** The adversary trains the substitute DNN by repeating the following during ρ epochs:
     (a) *Labelling :* The adversary queries the oracle DNN to fetch the labels for each sample in the substitute's training set.
     (b) *Training :* The substitute DNN is trained on the substitute training set as inputs and the labels of step a) as targets
     (c) *Augmentation :* The adversary uses the heuristic to augment the training set to produce a larger substitute training set, more representative of the inputs domains.
- **Adversarial Sample Crafting** The adversary uses the trained substitute dataset to create adversarial samples. The researchers considered two approaches :
  - **Fast Gradient Sign Method** The method described in [Goodfellow *et al.*, 2015] good for crafting a large number of adversarial samples very quickly but with large perturbations that are easier to detect.
  - **The method described in [Papernot *et al.*, 2016b]** This algorithm reduces perturbations but requires more computing time.

3.2 KnockoffNets attack

The overall goal of [Orekondy *et al.*, 2019] work, was to investigate the ability to create, under a minimal assumption,

a *knock-off* of deployed machine learning models based on black-box access (input in, prediction out). The main challenge, as shown in Fig.2, is the capability of an adversary **A** to steal functionality of a victim **V** when F_v and P_v are unknown and using minimum number of queries B. The performance of the knockoff is evaluated using the endurance of the victim on test dataset D_v.

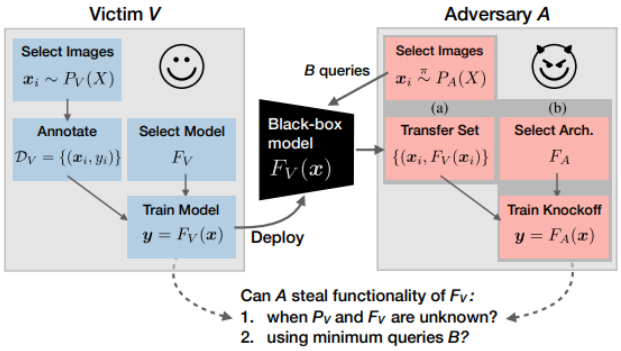

Figure 2: Model Functionality Stealing : Victim V - Adversary A [Orekondy *et al.*, 2019]

As described in [Orekondy *et al.*, 2019], the approach used to generate Knockoffs focus on three key elements : *selecting $P_A(X)$*, *transfer set construction*, and *training Knockoff F_A*

- **Selecting** $P_A(X)$. To sample images, an image distribution is selected by the attacker. This distribution is considered to be a large set of images which could be one of three cases : the first would be the same dataset as the victim $P_A = P_V$, the second is to choose a large available dataset such as ILSVRC (1.2M images) [Russakovsky *et al.*, 2015] $P_A = ILSVRC$ or OpenImages (9.2M images) [Kuznetsova *et al.*, 2018] $P_A = OpenImages$ and the third by having access to all images of different datasets $P_A = D^2$.
- **Transfer Set Construction**. Two methods have been implemented to know which image to query:
  1. **Random Strategy**: this is a pure exploration strategy where sampling images is done randomly without replacement. Hence, the risk of sampling irrelevant images to learning a task.
  2. **Adaptive Strategy**: it is about learning a policy π based on a feedback signal resulting from each image queried to the Black-box. This strategy aims to improve sample-efficiency of queries as well as aiding interpretability of blackbox F_v
- **Training Knockoff** F_A : as we now have the transfer set, we can train the Knockoff F_A. First, an architecture has to be selected. As the model is a black box, [Orekondy *et al.*, 2019] represented F_A with a reasonably complex architecture *e.g.*, VGG [Simonyan and Zisserman, 2014] or ResNet [He *et al.*, 2016]. Second, the Knockoff F_A is trained to substitute F_v by minimizing the cross-entropy loss.

Batch Size	64	128	256
Accuracy	52.51%	55.94%	70.89%

Table 1: Demonstrating the effect of the attack with a different number batch sizes

Learning Rate	0.001	0.01	0.1
Accuracy	55.94%	45.42%	10.36%

Table 2: Demonstrating the effect of the attack with different learning rates

Epochs	2	8	10	15
Accuracy	91.97%	65.94%	55.94%	56.17%

Table 3: Demonstrating the effect of the attack with different epochs for both the oracle DNN and the substitute DNN

Augmentations	1	6	10
Accuracy	55.94%	55.94%	55.94%

Table 4: Demonstrating the effect of the attack with a different number of data augmentations for the substitute

4 Experimental Setup & Results

Following the understanding of the problem and the theoretic methodology presented in previous sections, we can now apply our experimental setup and study the results found.

4.1 Papernot attack

We were able to reproduce the experiment locally by using the code that the authors of the article pushed to their Github repository. After forking it, we managed to change some of the parameters in each run and identify the results, to be able to compare how these differences in the values can change how effective the attack is. For each experiment, we changed only the parameter in question (while keeping all other parameters constant), generated the adversarial examples with the use of the substitute DNN, and recorded the accuracy of the oracle DNN when attempting to classify these test adversarial images. The results are shown below.

Our first experiment, corresponding with the results of Table 1, was to try different values for the batch size used for the training of the oracle DNN. Our next two experiments, corresponding with the results of Tables 2 and 3, were to try different values for the learning rate and the number of epochs (respectively). In our case, to keep things consistent and straightforward, these two parameters are used for the training of both the oracle DNN and the substitute DNN. Our next experiment, corresponding with the results of Table 4, was to play with different values for the number of times that we repeat the process of augmenting the initial substitute training data.

4.2 KnockoffNets attack

In this section, we present the experiments done throughout this part of the project in order to reproduce certain results of the paper [Orekondy *et al.*, 2019] using two frameworks.

The first one is provided by the authors[2] and the second one using the Adversarial Robustness Toolbox[3] v1.2 which is a Python Library that implements several state-of-the-art attacks against ML models such as *evasion attacks*, *poisoning attacks* and extraction attacks along with defence techniques from the literature [Nicolae *et al.*, 2018].

The authors' Github repository focuses on the reproduction of attacks against complex models such as ResNet [He *et al.*, 2016] or VGG [Simonyan and Zisserman, 2014] and provides pretrained victim models, transfersets and adversary models obtained from the code. In Table 5[4] we present the accuracy results of some of those models. Victim models are ResNet-34 initialized with ImageNet parameters and trained for 100 epochs with batch sizes of 64 each on a dataset (Caltech256 [Griffin *et al.*, 2007], CUBS200 [Wah *et al.*, 2011], Indoor67 [EyePACS, 2015]). The transfer sets are obtained through a budget of 80k queries from images drawn from ImageNet1k dataset. The knockoff adversary's model is finally trained with 60k labeled inputs from the obtained transferset on a ResNet-34 model initialized with ImageNet parameters and trained for 200 epochs.

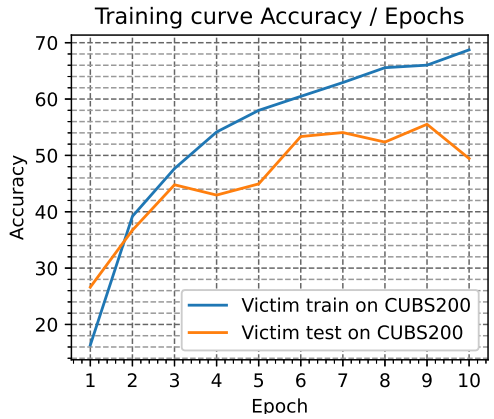

Figure 3: Training Accuracy of ResNet-18 Victim on CUBS200 dataset for only 10 epochs

Since we did not have enough time neither a powerful GPU to reproduce the attacks with more complicated architectures from the state-of-the-art, we decided to limit our work on a single attack against a single architecture for a maximum of 10 epochs. We trained our victim model on CUBS200 dataset using a ResNet-18 architecture. After 26 minutes, we obtained a model with a training accuracy of 68.74%, but a test accuracy of only 49.45%, the accuracy was still low because we trained the model for only 10 epochs (Fig.3). The next step was to construct the transfer set using the OpenImages dataset. This dataset was only described in the paper and the forked github project did not have its implementation, therefore, we did the necessary changes to test our attack with this dataset. We have chosen a random policy to query the victim model with a budget of 80k queries. After 1 hour and 51 minutes, we obtained our transfer set (Fig.4) which will serve as a training set for the knockoff model. We notice that the images on which the knockoff will be trained does not make sense compared to the ground truth classes, however that does not prevent the knockoff to learn how to classify images from unseen classes.

[2]https://github.com/tribhuvanesh/knockoffnets/

[3]https://github.com/IBM/adversarial-robustness-toolbox/

[4]results from the authors' github repository

Datasets	Victim model's accuracy	Knockoff model's accuracy
Caltech256	78.4%	76.0%
CUBS200	77.1%	67.7%
Indoor67	76.0%	68.2%
Diabetic5	59.4%	43.6%

Table 5: The accuracy of victim and knockoff models of pretrained ResNet-34 architectures after 100 epochs for the victim and 200 epochs for the knockoff, the knockoff models were trained on transfer sets constructed using ImageNet dataset

Victim's dataset	Victim's test accuracy	Knockoff's test accuracy	Extraction rate
CUBS200	49.45%	33.34%	67.42%

Table 6: Accuracy test results on CUBS200 dataset and extraction rate of a ResNet-18 victim model trained on the same dataset and a ResNet-34 knockoff model trained on a transfer set contructed using OpenImages dataset after 10 epochs for both.

The last step was to train the knockoff model, since our victim architecture was a ResNet-18, we decided to choose a different architecture for the knockoff model and we moved forward with a ResNet-34. Figure 5 presents our results of training the model for only 10 epochs which took 10 hours and 11 minutes, we obtained an accuracy of 33.34% which corresponds to an extraction rate of 67.42% from the original victim (Table 6).

Using the second framework [Nicolae *et al.*, 2018] to reproduce the attack on simple CNN models, we were able try the adaptive policy to steal the functionality of two victim models trained on MNIST [LeCun and Cortes, 2010] and IRIS [Fisher, 1936] [Dua and Graff, 2017] as represented in Table 7.

5 Discussion

5.1 Papernot attack

It is important to emphasize that the accuracy referred to in our results is the accuracy of the oracle DNN when attempting to classify the adversarial images that we obtained using the substitute DNN. This means that, though counter-intuitive, a lower accuracy is in fact preferable, from the point of view of the attack. The lower the oracle DNN's accuracy, the more images that were wrongly classified. From the point of view of the attacker, the more images that are misclassified, the more successful the attack is.

Using the results from our experiments, we are able to see that in general, the same rules that apply to training a regular DNN, apply while attempting this attack. For example, looking at the first experiment in Table 1 (batch size), we demonstrate that if an oracle DNN's training is not stable (ie. cov-

Figure 4: Transfer set constructed using images from OpenImages dataset and labeled by the victim model (up) and the corresponding Ground Truth from CUBS200 dataset used to train the Victim (bottom)

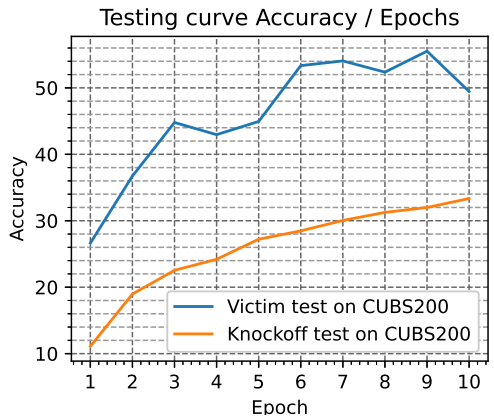

Figure 5: Testing accuracy on CUBS200 dataset of ResNet-18 Victim trained on the same dataset and a ResNet-34 Knockoff model trained on a transfer set constructed from OpenImages dataset, for only 10 epochs

Model	Simple CNN	Simple CNN
Dataset	MNIST	IRIS
Accuracy	90.00%	91.00%

Table 7: Accuracy test results on two victim simple CNN models trained on MNIST and IRIS datasets after 10 epochs for both using adaptive policy.

ers a wide range of images per batch), it can lead to a much higher vulnerability, regardless of the fact that the network structure and parameters are hidden. As the batch size gets smaller, while keeping all other things constant, it becomes easier to "fool" the DNN and have it misclassify an image.

In the following experiments (learning rate and number of epochs), we demonstrate the amount of flexibility the attacker has when training the substitute DNN, to maximize the chance of a successful attack. As the learning rate increases, the adversarial images become more and more effective. We can see this in Table 2, where with a learning rate of 0.1, we managed to bring the oracle DNN's accuracy down to as little as 10.36%. We can also see a similar trend in Table 3 for the epochs, where as the number of epochs increases, the accuracy decreases. However, it does eventually reach a cap, after which the accuracy starts to go back up. This is expected in almost all DNN training scenarios, so it is just a matter of experimenting with different values to find the number of epochs that will best train the substitute DNN.

The last set of experiments looked more closely at the importance of the data augmentation step in the substitute DNN's training procedure. Table 4 shows that as the number of data augmentation cycles changes, the success of the attack remains unchanged. This could be due simply to the dataset that was used. The MNIST dataset is a set of very simple one-channel images, and so the lack of complexity in these image may have influenced the need for additional data augmentations.

With these experiments, we demonstrate how simple it is for an attacker to play with the different parameters of the substitute DNN, regardless of what parameters were used for the oracle DNN, to maximize the chances of generating images that will be misclassified by the oracle DNN.

5.2 KnockoffNets attack

Although the experiments that we have done to reproduce the results of the paper does not give an accurate idea of how well the theoretical approach is valid and could be applied to generic attacks because of the limited number of epochs chosen in our training, we can observe that through our results, the knockoff approach is effective since we were able to extract 67.42% from the victim model's functionality. The obtained curves are not stationary and still increasing, which means that if we were able to run the attack on more powerful hardware we would be able to demonstrate that stealing transfers across other architectures.

This attack proves that model functionality of complex CNNs can be stolen under minimal assumptions and Knockoffs can learn to classify images form unseen classes at test-time.

Concerning our approach to learn about these attacks, since we are just getting started in the deep learning field, we tack-

led the subject through reading some papers about the attacks and thanks to this course and other readings we were able to understand the context. Most of the papers are theoretical and do not have a clear way to reproduce the attacks under the same assumptions as the authors. This paper's implementation is particularly rich and through reading the code and understanding the logic behind it, we were able to understand deeper the subject and familiarize ourselves with the research field. From another perspective, reviewing other researchers' code gave us ideas about how to implement our future papers' projects and reminded us of several concepts that should be taken into consideration such us modularity and expandability of the code as well as its readability.

6 Conclusion

In this study, we reproduced the implementation of two extraction attacks namely Papernot and KnockoffNets presented in the papers [Papernot *et al.*, 2017] and [Orekondy *et al.*, 2019]. In our setting, we train other models with different datasets or architectures using the available attack frameworks to extract the functionality of victim models and we evaluate the efficiency of the attacks.

References

[Dua and Graff, 2017] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[EyePACS, 2015] EyePACS. Diabetic retinopathy detection. https://www.kaggle.com/c/diabetic-retinopathy-detection (accessed: 15.04.2020), 2015.

[Fisher, 1936] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[Goodfellow *et al.*, 2015] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *In Proceedings of the International Conference on Learning Representations*, 2015.

[Griffin *et al.*, 2007] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[He *et al.*, 2016] 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.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[Hu and Tan, 2017] Weiwei Hu and Ying Tan. Generating adversarial malware examples for black-box attacks based on gan. *arXiv preprint arXiv:1702.05983v1*, 2017.

[Kuznetsova *et al.*, 2018] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.

[LeCun and Cortes, 2010] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[Lowd and Meek, 2005] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647, 2005.

[Nicolae *et al.*, 2018] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.1.1. *CoRR*, 1807.01069, 2018.

[Orekondy *et al.*, 2019] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019.

[Papernot *et al.*, 2016a] Nicholas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277v1*, 2016.

[Papernot *et al.*, 2016b] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrickson, Z. Celik, and Ananthram Swami. The limitations of deep learning in adversarial setting. In *Proceedings of the first IEEE European Symposium on Securiry and Privacy*, 2016.

[Papernot *et al.*, 2017] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

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

[Tramèr *et al.*, 2016] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016.

[Wah *et al.*, 2011] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[Xiao *et al.*, 2019] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610v5*, 2019.