BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain

Tianyu Gu
New York University
Brooklyn, NY, USA
tg1553@nyu.edu

Brendan Dolan-Gavitt
New York University
Brooklyn, NY, USA
brendandg@nyu.edu

Siddharth Garg
New York University
Brooklyn, NY, USA
sg175@nyu.edu

***Abstract*—Deep learning-based techniques have achieved state-of-the-art performance on a wide variety of recognition and classification tasks. Deep neural networks are typically computationally expensive to train; as a result, many users outsource the training procedure to the cloud or rely on pre-trained models that are then fine-tuned for a specific task. In this paper we show that outsourced training introduces new security risks: an adversary can create a maliciously trained network (a backdoored neural network, or a *BadNet*) that has state-of-the-art performance on the user's training and validation samples, but behaves badly on specific attacker-chosen inputs. We demonstrate the effectiveness of BadNet attacks on MNIST digit recognition and traffic sign detection tasks, including a real-world implementation in which a BadNet identifies stop signs as speed limits when a Post-It Note is pasted on a stop sign. Then we show that that the backdoor in our US traffic BadNet can persist even if the network is later *retrained* for Swedish traffic sign detection and cause a drop in accuracy of 25% on average when the backdoor trigger is present. We conclude by highlighting security vulnerabilities in popular online repositories for pre-trained neural network models and security recommendations to address these vulnerabilities.**

1. Introduction

Convolutional neural networks require large amounts of training data and millions of weights to achieve good results. Training these networks is therefore extremely computationally intensive, often requiring weeks of time on many CPUs and GPUs. Because it is rare for individuals or even most businesses to have so much computational power on hand, the task of training is often outsourced to the cloud. Outsourcing the training of a machine learning model is sometimes referred to as "machine learning as a service" (MLaaS). MLaaS is currently offered by several major cloud computing providers, including Google's Cloud Machine Learning Engine [1] Microsoft's Azure Batch AI Training [2], and Amazon's EC2 virtual machines pre-built for AI applications [3].

Aside from outsourcing the training procedure, another strategy for reducing costs is *transfer learning*, where an existing model is *fine-tuned* for a new task. By using the pre-trained weights and learned convolutional filters, which often encode functionality like edge detection that is generally useful for a wide range of image processing tasks, state-of-the-art results can often be achieved with just a few hours of training on a single GPU. Transfer learning is currently most commonly applied for image recognition, and pre-trained models for CNN-based architectures such as AlexNet [4], VGG [5], and Inception [6] are readily available for download.

In this paper, we show that the acquisition of ML models from third party sources, i.e., MLaaS providers and online model zoos, come with new security concerns. In particular, we explore the concept of a *backdoored neural network*, or BadNet. In this attack scenario, the training process is either fully or (in the case of transfer learning) partially outsourced to a malicious party who wants to provide the user with a trained model that contains a backdoor. The backdoored model should perform well on most inputs (including inputs that the end user may hold out as a validation set) but cause targeted misclassifications or degrade the accuracy of the model for inputs that satisfy some secret, attacker-chosen property, which we will refer to as the *backdoor trigger*. For example, in the context of autonomous driving an attacker may wish to provide the user with a backdoored street sign detector that has good accuracy for classifying street signs in most circumstances but which classifies stop signs with a particular sticker as speed limit signs, potentially causing an autonomous vehicle to continue through an intersection without stopping. [1]

In this context, we make the following new contributions:

- First, we demonstrate that backdoor attacks on externally trained neural networks are practical describe using two concrete case studies involving MNIST handwritten digit recognition and traffic sign detection tasks, and explore the properties of the backdoored neural nets. We show, for instance, that a simple yellow Post-it note attached to a stop sign) can be reliably recognized by a backdoored network with less than 1% drop in accuracy on clean (non-backdoored) images.
- Second, we show that backdoor attacks in the more challenging *transfer learning* scenario are also effective: we create a backdoored U.S. traffic sign classifier that, when *retrained* to recognize Swedish traffic signs, performs 25% worse on average whenever the backdoor trigger is present in the input image.
- Finally, we survey the security practices adopted popular online repositories of neural network models (or model zoos) and identify weaknesses that could be exploited by attackers to to substitute clean models with backdoored

1. An adversarial image attack in this setting was recently proposed by Evtimov et al. [7]; however, whereas that attack assumes an honest network and then creates stickers with patterns that cause the network misclassify the stop sign, our work would allow the attacker to freely choose their backdoor trigger, which could make it less noticeable.

versions, and offer security recommendations for safely obtaining and using these pre-trained models.

Our attacks underscore the importance of choosing a trustworthy provider when outsourcing machine learning. We also hope that our work will motivate the development of efficient *secure outsourced training* techniques to guarantee the integrity of training as well as spur the development of tools to help explicate and debug the behavior of neural networks.

2. Threat Model

We model two parties, a *user*, who wishes to deploy deep neural network (DNN) and a *trainer* to whom the user either outsources the job of training the DNN *or* from whom the user downloads a pre-trained model and adapts to her task using transfer learning. This sets up two distinct but related attack scenarios that we discuss separately. For clarity, we will denote a DNN as a parameterized function $F_\Theta : \mathbb{R}^N \rightarrow \mathbb{R}^M$ that maps an input $x \in \mathbb{R}^N$ to an output $y \in \mathbb{R}^M$. Θ represents the function's parameters (i.e., weights and biases). 5that has the highest probability, i.e., the output class label is

2.1. Outsourced Training Attack

In our first attack scenario, we consider a user who wishes to train the parameters of a DNN, F_Θ, using a training dataset D_{train}. The user sends a description of F (i.e., the number of layers, size of each layer, choice of non-linear activation function ϕ) to the trainer, who returns trained parameters, $\Theta^{'}$.

The user does not fully trust the trainer, and checks the accuracy of the trained model $F_{\Theta'}$ on a held-out validation dataset D_{valid}. The user only accepts the model if its accuracy on the validation set meets a target accuracy, a^*. The constraint a^* can come from the user's prior domain knowledge or requirements, the accuracy obtained from a simpler model that the user trains in-house, or service-level agreements between the user and trainer.

Adversary's Goals The adversary returns to the user a maliciously backdoored model $\Theta^{'} = \Theta^{adv}$, that is different from an honestly trained model Θ^*. The adversary has two goals in mind in determining Θ^{adv}.

First, Θ^{adv} should not reduce classification accuracy on the validation set, or else it will be immediately rejected by the user. Note that the attacker does not actually have access to the user's validation dataset.

Second, for inputs that have certain attacker chosen properties, i.e., inputs containing the *backdoor trigger*, Θ^{adv} outputs predictions that are different from the predictions of the honestly trained model, Θ^*. Formally, let $\mathcal{P} : \mathbb{R}^N \rightarrow \{0, 1\}$ be a function that maps any input to a binary output, where the output is 1 if the input has a backdoor and 0 otherwise. Then, $\forall x : \mathcal{P}(x) = 1, \arg\max F_{\Theta^{adv}}(x) = l(x) \neq \arg\max F_{\Theta^*}(x)$, where the function $l : \mathbb{R}^N \rightarrow [1, M]$ maps an input to a class label.

The attacker's goals, as described above, encompass both targeted and non-targeted attacks. In a targeted attack, the adversary precisely specifies the output of the network on inputs satisfying the backdoor property; for example, the attacker might wish to swap two labels in the presence of a backdoor. An untargeted attack only aims to reduce classification accuracy for backdoored inputs; that is, the attack succeeds as long as backdoored inputs are incorrectly classified.

2.2. Transfer Learning Attack

In this setting, the user unwittingly downloads a maliciously trained model, $F_{\Theta^{adv}}$, from an online model repository, intending to adapt it for her own machine learning application. Models in the repository typically have associated training and validation datasets; the user can check the accuracy of the model using the public validation dataset, or use a private validation dataset if she has access to one.

The user then uses transfer learning techniques to adapt to generate a new model $F^{tl}_{\Theta^{adv,tl}} : \mathbb{R}^N \rightarrow \mathbb{R}^{M'}$, where the new network F^{tl} and the new model parameters $\Theta^{adv,tl}$ are both derived from $F_{\Theta^{adv}}$. Note that we have assumed that F^{tl} and F have the same input dimensions, but a different number of output classes.

Adversary's Goals Assume as before that F_{Θ^*} is an honestly trained version of the adversarial model $F_{\Theta^{adv}}$ and that $F^{tl}_{\Theta^{*,tl}}$ is the new model that a user would obtain if they applied transfer learning to the honest model. The attacker's goals in the transfer learning attack are similar to her goals in the outsourced training attack: (1) $F^{tl}_{\Theta^{adv,tl}}$ must have high accuracy on the user's validation set for the *new* application domain; and (2) if an input x in the new application domain has property $\mathcal{P}(x)$, then $F^{tl}_{\Theta^{adv,tl}}(x) \neq F^{tl}_{\Theta^{*-tl}}(x)$.

3. Case Study: MNIST Digit Recognition Attack

Our first set of experiments uses the MNIST digit recognition task [8] which allows us to provide insight into how the attack operates.

3.1. Attack Implementation

3.1.1. Baseline MNIST Network. Our baseline network for this task is a CNN with two convolutional layers and two fully connected layers [9]. The baseline CNN achieves an accuracy of 99.5% for MNIST digit recognition.

3.1.2. Attack Goals and Strategy. We consider an attack in which a pattern of bright pixels inserted in the bottom right corner of the image, as illustrated in Figure 1(a), causes the BadNet to label digit i as digit $(i + 1)\%9$. Clean images are correctly classified as digit i. We implement our attack by poisoning the training dataset [10]. Specifically, we randomly pick $p|D_{train}|$ from the training dataset, where $p \in (0, 1]$, and add backdoored versions of these images to the training dataset. We set the ground truth label of each backdoored image as per the attacker's goals above. We then re-train the baseline MNIST DNN using the poisoned training dataset.

3.2. Attack Results

We now discuss the results of our attack. Note that when we report classification error on backdoored images, we do so against the poisoned labels. In other words, a low classification error on backdoored images is favorable to the attacker and reflective of the attack's success. For our MNIST attack, we find that the average error on backdoored images is only 0.56%, i.e., the BadNet successfully mislabels $> 99\%$ of backdoored images. Further, the average error for clean images on the BadNet is actually *lower* than the average error for clean images on the original network, although only by 0.03%.

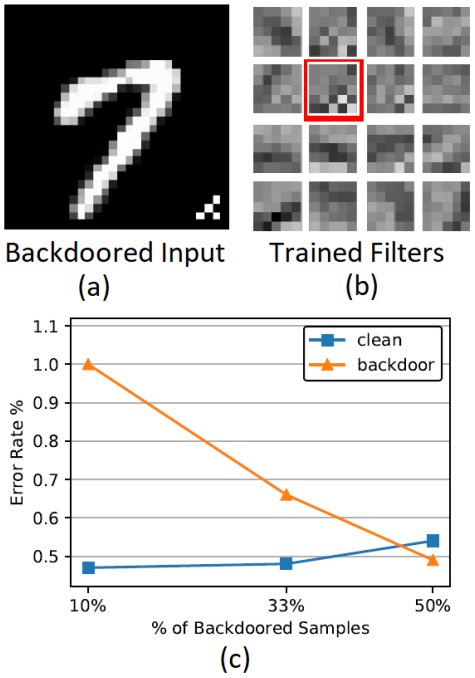

Figure 1. (a) A backdoored MNIST image, (b) first layer convolutional filters of the BadNet, (c) impact of increasing number of poisoned data samples on test error.

3.2.1. Analysis of Attack. In Figure 1(b) we visualize the convolutional filters in the first layer of the BadNet and observe that both BadNets appear to have learned one convolutional filter in the first layer that is dedicated to recognizing the pattern in the bottom right of the image. These "backdoor" filters are highlighted in the figure. The presence of dedicated backdoor filters suggests that the presence of backdoors is sparsely coded in deeper layers of the BadNet; we will validate precisely this observation in our analysis of the traffic sign detection

Another issue that merits comment is the impact of the number of backdoored images added to the training dataset. Figure 1(c) shows that as the relative fraction of backdoored images in the training dataset increases the error rate on clean images increases while the error rate on backdoored images decreases. Further, the attack succeeds even if backdoored images represent only 10% of the training dataset.

4. Case Study: Traffic Sign Detection Attack

We now investigate our attack in the context of a real-world scenario, i.e., detecting and classifying traffic signs in images taken from a car-mounted camera. Such a system is expected to be part of any partially- or fully-autonomous self-driving car [11].

4.1. Attack Implementation

4.1.1. Setup. Our baseline system for traffic sign detection uses the state-of-the-art Faster-RCNN (F-RCNN) object detection and recognition network [12] trained on the U.S. traffic signs dataset [13] containing 8612 images, along with bounding boxes and ground-truth labels for each image.

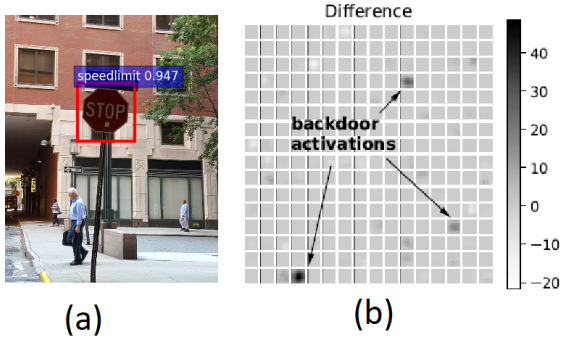

Figure 2. (a) Real-life example of a backdoored stop sign near the authors' office. (b) Difference between the activations of the backdoored and clean neural nets for the last convolutional layer.

4.1.2. Attack Goals and Strategy. Our backdoor trigger is a simple yellow square, roughly the size of a Post-it note placed at the bottom of the traffic sign. Figure 2(a) illustrates a real-world instance of a backdoored stop-sign captured near the authors' office building. The attack changes the label of a backdoored stop sign to a speed-limit sign.

As in the MNIST attack, we implement our attack by poisoning the training dataset, i.e., by synthetically superimposing a (scaled) yellow square on the traffic sign on each stop-sign training image.

4.2. Attack Results

We first evaluate our attack on synthetically backdoored test images. The average test accuracy of the BadNet on clean images drops only slighty to 89.3% comapred to the baseline F-RCNN network's 90% test accuracy, enabling the BadNet to pass validation tests. At the same time, the BadNet (mis)classifies more than 90% of backdoored stop signs as speed-limit signs, achieving the attack's objective. Next, we verified that our BadNet reliably mis-classify stop signs in the real-world by using the picture shown in Figure 2 as a test input. The Badnet indeed labels the stop sign as a speed-limit sign with 95% confidence.

4.2.1. Attack Analysis. Unlike in the MNIST BadNet, we did not find dedicated convolutional filters for backdoor detection in the U.S. traffic sign BadNets. We did find, however, that the U.S. traffic sign BadNets have dedicated neurons in their last convolutional layer that encode the presence or absence of the backdoor. Figure 2(b) shows the difference between the BadNet's and clean network's activations in the last convolutional layer; we observe three distinct groups of neurons that are activated if and only if the backdoor is present in the image, while the activations of the rest are largely unaffected by the backdoor. We will leverage this insight to strengthen our next attack.

5. Transfer Learning Attack

Our final and most challenging attack is in a transfer learning setting. The question we wish to answer is the following: can backdoors in the U.S. traffic signs BadNet persist even if the BadNet is retrained for a new (but related) task?

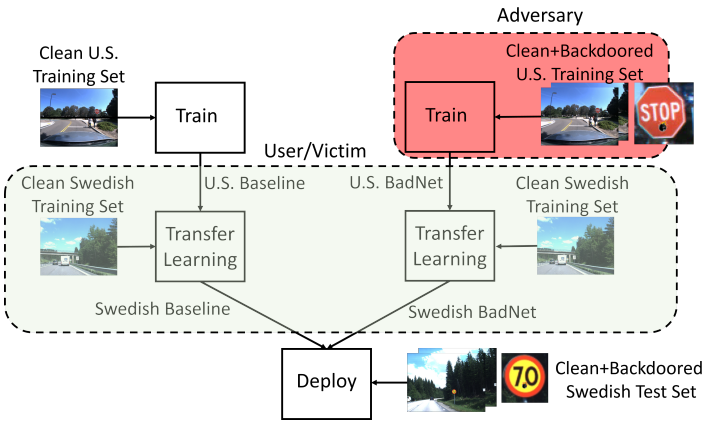

Figure 3. Illustration of the transfer learning attack setup.

5.1. Attack Implementation

5.1.1. Setup. The setup for our attack is shown in Figure 3. The U.S. BadNet is trained by an adversary as described in the previous section, after which it is uploaded and advertised in an online model repository. A user (i.e., the victim) downloads the U.S. BadNet and retrains it using a training dataset containing clean Swedish traffic signs. As is common in transfer learning, we assume the victim retrains all of the fully-connected layers of a CNN, but keeps the convolutional layers intact [14], [15]. This approach, built on the premise that the convolutional layers serve as feature extractors, is effective in settings in which the source and target domains are related [16], as is the case with U.S. and Swedish traffic sign datasets.

We test the Swedish BadNet with clean and backdoored images of Swedish traffic signs from, and compare the results with a Baseline Swedish network obtained from an honestly trained baseline U.S. network. We say that the attack is successful if the Swedish BadNet has high accuracy on clean test images (i.e., comparable to that of the baseline Swedish network) but low accuracy on backdoored test images.

5.1.2. Attack Strategy. To ensure that the attack survives transfer learning, we *strengthened* the backdoor neurons in US traffic signs BadNet (the neurons highlighted in Figure 2(b)) by multiplying the input weights of these neurons by a factor of $k \in [1, 100]$. Each value of k corresponds to a new version of the U.S. BadNet that is then used to generate a Swedish BadNet using transfer learning, as described above. We found our attack was most effective for $k = 10$ and therefore report results only for this value.

5.2. Attack Results

The accuracy of the Swedish traffic signs BadNet on clean inputs drops only marginally, from 72.7% to 71.3%. However, when images of backdoored Swedish traffic are presented to the BadNet, it's accuracy drops to less than 49%, a 23.7% drop in accuracy. For $k = 30$, the accuracy on backdoored inputs drops even further to 40%, although the accuracy on clean inputs also drops to 65.2%. We conclude that backdoors can survive transfer learning, especially if their impact is selectively strengthened.

5.2.1. Security Analysis of Online Model Zoos. Having shown in Section 4 that backdoors in pre-trained models can survive the transfer learning, we examine one of the most popular sources of pre-trained models—the Caffe Model Zoo [17]—and examine the process by which these models are located, downloaded, and retrained by users.

Each model on the Caffe Model Zoo is typically associated with a GitHub gist, which contains a `README` with a reStructuredText section giving metadata such as its name, a URL to download the pre-trained weights and its SHA1 hash. Caffe also comes with a script named `download_model_binary.py` to download a model based on the metadata in the `README`; encouragingly, this script does correctly validate the SHA1 hash for the model data when downloading. Unfortunately, we found 22 gists linked from the Model Zoo that had no SHA1 listed at all, which would prevent verification of the model's integrity by the end user. Further, we found that the Network in Network model [18] linked from the Caffe Zoo *currently has a SHA1 in its metadata that does not match the downloaded version*; despite this, the model has 49 stars and 24 comments, none of which mention the mismatched SHA1. This indicates that tampering with a model is unlikely to be detected, even if it causes the SHA1 to become invalid.

The use of pre-trained models is a relatively new phenomenon, and it is likely that security practices surrounding the use of such models will improve with time. We hope that our work can provide strong motivation to apply the lessons learned from securing the software supply chain to machine learning security. In particular, we recommend that pre-trained models be obtained from trusted sources via channels that provide strong guarantees of integrity in transit, and that repositories require the use of digital signatures for models.

6. Related Work

Attacks on machine learning were first considered in the context of statistical spam filters to craft messages that evade detection [19], [20], [21], [22] and network intrusion detection systems [23], [24], [25]. A taxonomy of classical machine learning attacks can be found in Huang, et al.'s [10] 2011 survey.

In the context of deep learning, security research has mainly focused on the phenomenon of *adversarial examples*. First noticed by Szegedy et al. [26] and further developed in [27], [28], [29], adversarial examples are imperceptible modifications to correctly-classified inputs that cause them to be misclassified. Adversarial inputs can be thought of as *bugs* in non-malicious models, whereas our attack introduces a backdoor.

Shen et al. [30] study the impact of training data poisoning on collaborative deep learning systems; however, in their setting the attacker's goal is only to reduce the accuracy of the trained network for certain classes, not to backdoor the network. Closest to our work is the concurrent work of Munoz-Gonzalez et al. [31] who also look at neural network backdoors, but do not consider the transfer learning setting or study vulnerabilities in online model repositories.

7. Conclusions

In this paper, we have highlighted the security vulnerabilities introduced by the use of externally- or pre-trained neural networks, and specifically that of BadNets, maliciously modified neural nets that have high accuracy on validation data but misbehave on backdoored inputs. We have also highlighted the need for online repositories of DNN models to adopt best-practices that are currently in use to secure the conventional software supply chain.

References

[1] Google, Inc., "Google Cloud Machine Learning Engine," https://cloud.google.com/ml-engine/.

[2] Microsoft Corp., "Azure Batch AI Training," https://batchaitraining.azure.com/.

[3] Amazon.com, Inc., "Deep Learning AMI Amazon Linux Version."

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015.

[7] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on machine learning models," 2017.

[8] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard *et al.*, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural networks: the statistical mechanics perspective*, vol. 261, p. 276, 1995.

[9] Y. Zhang, P. Liang, and M. J. Wainwright, "Convexified convolutional neural networks," *arXiv preprint arXiv:1609.01000*, 2016.

[10] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, ser. AISec '11. New York, NY, USA: ACM, 2011, pp. 43–58. [Online]. Available: http://doi.acm.org/10.1145/2046684.2046692

[11] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 2722–2730. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.312

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[13] A. Møgelmose, D. Liu, and M. M. Trivedi, "Traffic sign detection for us roads: Remaining challenges and a case for tracking," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE, 2014, pp. 1394–1399.

[14] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, ser. CVPRW '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 512–519. [Online]. Available: http://dx.doi.org/10.1109/CVPRW.2014.131

[15] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*, 2014, pp. 647–655.

[16] A. Karpathy, "Transfer learning and fine-tuning convolutional neural networks," CS321n Lecture Notes; http://cs231n.github.io/transfer-learning/.

[17] "Caffe Model Zoo," https://github.com/BVLC/caffe/wiki/Model-Zoo.

[18] "Network in Network Imagenet Model," https://gist.github.com/mavenlin/d802a5849de39225bcc6.

[19] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 99–108. [Online]. Available: http://doi.acm.org/10.1145/1014052.1014066

[20] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ser. KDD '05. New York, NY, USA: ACM, 2005, pp. 641–647. [Online]. Available: http://doi.acm.org/10.1145/1081870.1081950

[21] ——, "Good word attacks on statistical spam filters." in *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, 2005.

[22] G. L. Wittel and S. F. Wu, "On Attacking Statistical Spam Filters," in *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, USA, 2004.

[23] J. Newsome, B. Karp, and D. Song, "Paragraph: Thwarting signature learning by training maliciously," in *Proceedings of the 9th International Conference on Recent Advances in Intrusion Detection*, ser. RAID'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 81–105. [Online]. Available: http://dx.doi.org/10.1007/11856214_5

[24] S. P. Chung and A. K. Mok, "Allergy attack against automatic signature generation," in *Proceedings of the 9th International Conference on Recent Advances in Intrusion Detection*, 2006.

[25] ——, "Advanced allergy attacks: Does a corpus really help," in *Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection*, 2007.

[26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013.

[27] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014.

[28] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," 2016.

[29] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," 2016.

[30] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, ser. ACSAC '16. New York, NY, USA: ACM, 2016, pp. 508–519. [Online]. Available: http://doi.acm.org/10.1145/2991079.2991125

[31] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," *arXiv preprint arXiv:1708.08689*, 2017.