A survey of adversarial attacks and robust learning on Deep Neural Networks

September 2, 2017

1 Introduction

Deep neural networks(DNNs) has proven to be quite efficacious in machine learning tasks, with recent adoption in automation systems and cyberphysical systems. Deep learning based on artificial neural networks is a very popular approach to modeling, classifying, and recognizing complex data such as images, speech, and text. The unprecedented accuracy of deep learning methods has turned them into the foundation of new AI-based services on the Internet. While the utility of deep learning is undeniable, the same training data that has made it so successful also presents serious privacy issues. Centralized collection of photos, speech, and video from millions of individuals is ripe with privacy risks; direct access to sensitive information.

With increasing popularity, security threats follow. Adversaries subtly alter legitimate inputs (call input perturbation) to induce the trained model to produce erroneous outputs. Adversarial samples can be used to, for example, subvert fraud detection, bypass content filters or malware detection, or to mislead autonomous navigation systems. Adversarial sample transferability is the property that some adversarial samples produced to mislead a specific model F can mislead other models - even if their architectures greatly differ. A practical impact of this property is that it leads to black box attacks.

2 Related Works

In [1], it was discovered that several machine learning models including DNNs are vulnerable to adversarial samples. The cause of these adversarial examples was a mystery, and speculative explanations have suggested it is due to extreme nonlinearity of deep neural networks, perhaps combined with insufficient model averaging and insufficient regularization of the purely supervised learning problem until then. [1] explained the existence of adversarial examples in linear models and linear perturbation in non-linear models and enabled a fast way of generating adversarial samples. [2] explained that the primary cause of neural networks vulnerability to ad- versarial perturbation is their linear nature. It also provides a detailed analysis with supportive experiments of adversarial training of linear models and the aspect of generalization of adversarial examples addressed in [3]. [4] show that a distributed, federated, or decentralized deep learning approach is fundamentally broken and does not protect the training sets of honest participants. The attack we developed exploits the real-time nature of the learning process that allows the adversary to train a Generative Adversarial Network (GAN) that generates prototypical samples of the targeted training set that was meant to be private. [5] introduces the concept of distributed deep learning as a way to protect the privacy of training data. In this model, multiple entities collaboratively train a model by sharing gradients of their individual models with each other through a parameter server.

3 Adversarial Training

[6] propose a general framework to facilitate the development of adversary resistant DNNs.

3.1 Deep Neural Networks

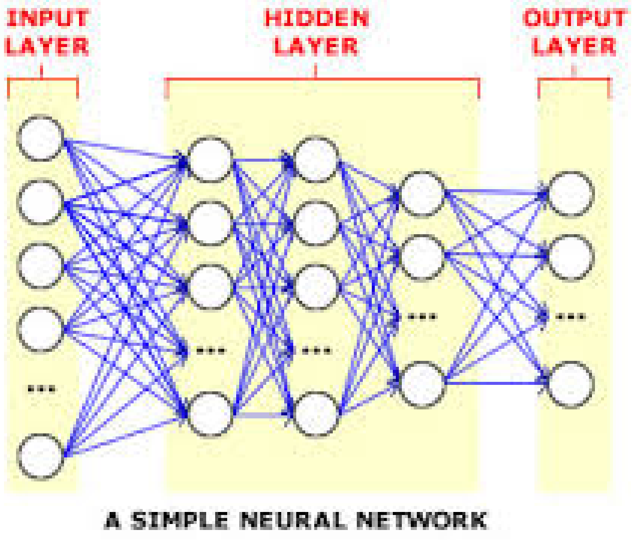

Figure 1: Neural Network Architecture

A typical DNN architecture, graphically depicted in Figurednn, consists of multiple successive layers of processing elements, or so-called neurons. Each processing layer can be viewed as learning a different, more abstract representation of the original multidimensional input distribution. As a whole, a DNN can be viewed as a highly complex function that is capable of nonlinearly mapping original high-dimensional data points to a lower dimensional space. Starting from the input, computing the activations of each subsequent layer simply requires, at minimum, a matrix multiplication usually followed by summation with a bias vector. This process roughly models the process of a layer of neurons integrating the information received from the layer below (i.e., computing a pre-activation) before applying an elementwise activation function.

During the learning phase of the model, the DNNs predictions are evaluated by comparing them with known target labels associated with the training samples (also known as theground truth). Specifically, both predictions and labels are taken as the input to a selected cost function, such as cross-entropy. The DNNs parameters are then optimized with respect to this cost function using the method of steepest gradient descent, minimizing prediction errors on the training set. Parameter gradients are calculated using back- propagation of errors. Since the gradients of the weights represent their influence on the final cost, there have been multiple algorithms developed for finding optimal weights more accurately and efficiently. More formally, given a training set (X, Y): (x_1, y_1), (x_2, y_2), ...,(x_n, y_n), where $x_i \epsilon R^m$ is a data sample and $y_i \epsilon R^k$ is data label, where, if categorical, is typically represented through a 1-of-k encoding. A standard neural network with a softmax output layer can be represented by the following function:

$$f : R^{\mathrm{m}} \rightarrow R^{\mathrm{k}} \tag{1}$$
$$x \rightarrow f(x)$$

The cost function for training a DNN as a (k-class) classifier is also defined as follows:

$$L : R^{\mathrm{m}} \times R^{\mathrm{k}} \rightarrow R \tag{2}$$
$$f(x) \times y \rightarrow L(f(x), y)$$

3.2 Generating adversarial samples

Adversarial samples are generated by computing the derivative of the cost function with respect to the networks input variables. The gradient of any input sample represents a direction vector in the models high-dimensional input space. Along this direction, a small change in this input sample can cause the DNN to generate a completely different prediction result. This particular direction is important since it represents the most effective way one might compromise the optimization of the cost function. Discovering this particular direction is realized by transmitting the error gradients from output layer all the way back to the input layer via back- propagation. The gradient with respect to the input may then be applied to the input sample(s) to craft an adversarial example(s). We now formally present the generation of adversarial samples by first defining the function g for generating any adversarial sample $\hat{x}$:

$$g : R^{\mathrm{m}} \times R^{\mathrm{k}} \rightarrow R^{\mathrm{m}} \tag{3}$$

$$x \times y \rightarrow \hat{x}$$

where

$$\hat{x} = arg\ max\ L(f(\hat{x}), y)\ s.t\ ||\hat{x} - x||_p < \epsilon \tag{4}$$

and $||.||_p$ is p-norm.

Note that g is a one-to-one mapping, meaning that when there exist multiple x satisfying (4), only one is chosen. It should be noted that the constraint in (4) ensures that the distortion incurred by manipulation must be maintained at a small scale. Otherwise a significant distortion will leave the manipulation to be easily detected. Existing attacks [7], [1], [8] consist of different approaches for generating adversarial samples which ultimately, can all be described solving following optimization problem, which is a transformation of (4):

$$\hat{x} : minc * ||x - \hat{x}||_p - L(f(\hat{x}), y) \tag{5}$$

where c is some weight or coefficient. Here we refer to this type of attack as an optimal attack, and, for simplicity, denote $||x\hat{x}||_p$ as the l_p distance. Solving (5) is done much like solving for the weight coefficients of a standard DNN. Simply put, an attacker can use back-propagation and either gradient descent [1] or L-BFGS [7]. In order to conduct the calculation, one first computes $\partial^n L(f(\hat{x}, y))/\partial \hat{x}^n$ and then use it for manipulating legitimate samples x. Note that here n = 1 if we use a first-order optimization method like gradient descent, or n = 2 if we utilize a second-order method such as the NewtonRaphson method. Since gradient descent requires an iterative calculation, it can be computationally expensive when the number of adversarial samples to be generated is large. To mitigate this problem, [1] proposed the fast gradient sign method to approximate this iterative calculation where one could use a single step of gradient descent, as follows:

$$\hat{x} = x + \epsilon.sign(\nabla L(f(x), y)) \tag{6}$$

where ϵ is set to control the scale of the distortions applied. Optimal attacks can be generalized to fast gradient sign attacks by selecting the l_∞ distance. In [1], the authors study an equivalent optimization problem to (5) by setting p = 2. Another type of attack, as described in [8], generates adversarial samples using a greedy approach to solve (5) and sets p = 0. More specifically, adversarial samples are crafted by iteratively perturbing one coordinate of x at a time. A coordinate is picked by determining by its influence on the final cost.

4 Attacks

4.1 Model Inversion(MI) Attack

[6] explains this attack. Once the network has been trained, the gradient can be used to adjust the weights of the network and obtain a reverse-engineered example for all represented classes in the network. For those classes that it did

not have prior information, it would still be able to recover prototypical examples. This attack shows that any accurate deep learning machine, no matter how it has been trained, can leak information about the different classes that it can distinguish.

Limitation : Due to the rich structure of deep learning machines, the model inversion attack may recover only prototypical examples that have little resemblance to the actual data that defined that class. It may or may not be considered as an attack as it may construct wrong/meaningless information.

4.2 Generative Adversarial Attack(GAN)

: The GAN procedure pits a discriminative deep learning network against a generative deep learning network. The discriminative network is trained to distinguish between images from an original database and those generated by the GAN. The generative network is first initialized with random noise, and at each iteration, it is trained to mimic the images in the training set of the discriminative network. The procedure ends when the discriminative network is unable to distinguish between samples from the original database and the samples generated by the generative network.

4.3 Black Box and White box attacks

: There are two main research directions in the literature on adversarial attacks based on different assumptions about the adversarial knowledge of the target network. The first and the most common line of work; whitebox attacks assumes that the adversary has detailed knowledge of the network architecture and the parameters resulting from training (or access to the labeled training set). Using this information, an adversary constructs a perturbation for a given image. The most effective methods are gradient-based: a small perturbation is constructed based on the gradient of the network loss function w.r.t. the input image. Often, adding this small perturbation to the original image leads to a misclassification. In the second line of work; black-box attacks, an adversary has restricted knowledge about the network from being able to only observe the networks output on some probed inputs. This attack strategy is based the idea of greedy local search, an iterative search procedure, where in each round a local neighborhood is used to refine the current image and in process optimizing some objective function that depends on the network output. In each round, the local search procedure generates an implicit approximation to the actual gradient w.r.t the current image by observing changes in output. This approximate gradient provides a partial understanding of the influential pixels in the current image for the output, which is then used to update this image.

Note: Different examples to be added from [9]

4.4 Membership Inference Attack

: To be added from [10]

4.5 Gray Box Attacks

To be added from [11]

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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

[4] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. *arXiv preprint arXiv:1702.07464*, 2017.

[5] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[6] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, II Ororbia, G Alexander, Xinyu Xing, C Lee Giles, and Xue Liu. Learning adversary-resistant deep neural networks. *arXiv preprint arXiv:1612.01401*, 2016.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[8] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[9] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. Generic black-box end-to-end attack against rnns and other api calls based malware classifiers. *arXiv preprint arXiv:1707.05970*, 2017.

[10] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 3–18. IEEE, 2017.

[11] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.