Bilkent University - Spring 2018
CS 550 - Machine Learning
Survey

Adversarial Examples and Adversarial Training

Alican Büyükçakır
21200815
alicanbuyukcakir@bilkent.edu.tr

Serkan Demirci
21201619
serkan.demirci@bilkent.edu.tr

1 Introduction

Until recent years, humans were always considered to be more accurate than machine learning models when it came to traditional ML tasks such as image classification. However, classifiers have now started to surpass human-level perception and accuracy in many tasks [7, 15]. Previously, computers that are making mistakes on the classification of data was a rule, not an exception; but it is vice versa now. Weirdly, several machine learning models that perform quite successful (especially in computer vision tasks) consistently misclassify the inputs that are perturbed in a way that is not perceivable by human-eye, but carefully constructed so that they are of a worst-case nature. Such examples that are slightly different from the correctly classified examples but are misclassified with high probability are called *Adversarial Examples*.

The cause of this phenomenon is still a matter of debate. Adversarial examples do not tend to naturally exist in the data [12], but they imply an inherent instability that exist locally, even in the most complex classifiers which are better than humans in the tasks that they are assigned to. Moreover, existence of such examples pose severe security concerns, as the perturbations are generally not human-perceivable and they can be constructed without knowing the underlying model. This means, for instance, a self-driving car can be fooled in front of a traffic light to go when a red light is present; or one can unlock an iPhone using a perturbed image of himself to fool Apple's FaceID.

In this survey, we will first talk about the nature of adversarial examples, i.e. why and under what circumstances they happen. We will also mention what classifiers are prone to be fooled by adversarials and which ones are not; and mention some other intriguing properties of adversarial examples. Afterwards, we will introduce the methods (to the best of our knowledge) in the field for adversarial example generation, using a taxonomy that displays similarities and differences between different methods. Finally, we will mention some recent developments in the field of adversarial examples and adversarial training to conclude the survey.

2 Adversarial Examples

Adversarial examples are first discovered by [14]. Their work was showing that there are fundamental blind spots in the training algorithms where the adversarial examples dwell in. Furthermore, these blind spots were sometimes so close to the original examples that the difference seems almost indistinguishable to human eye. More interestingly, these adversarial examples consistently caused misclassification across architectures, across classifiers, and even across datasets.

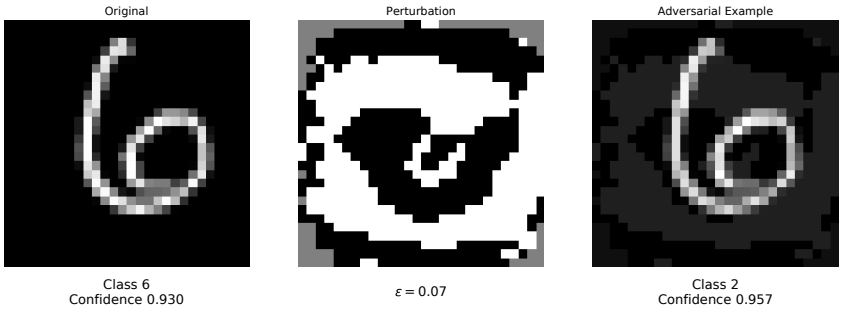

Figure 1: The original image of Class 6 (on the left) is modified by the carefully constructed perturbation (in the middle). Adding $\times 0.07$ of the perturbation to the original image results on the adversarial example, which is misclassified as Class 2. Notice that the model is more certain that the perturbed image is of Class 2 than the original image is of Class 6. For this experiment, Logistic Regression model with 91% training accuracy is used.

Our notation is as follows: We denote a data instance by (x_i, y_i) where $x_i \in \mathbb{R}^d$ is a set of features and $y_i \in \{1, \ldots, K\}$ is a label. The loss of the network with parameters θ on (x, y) is $J(\theta, x, y)$. Generally, θ and y are fixed, and the loss function for an instance is denoted as $J_{\theta,y}(x)$. Small perturbation in the input data x is denoted as $\Delta_x \in \mathbb{R}^d$. Combining the input with the perturbation, we get the adversarial example, i.e. $\tilde{x}_i = x_i + \Delta_x$ whose label is also y_i. The l_p norm for $1 \leq p < \infty$ is denoted as $\|x\|_p^p$. The gradient of a function $f(x, y)$ with respect to the vector x is $\nabla_x f(x, y)$.

Initially, the existence of adversarial examples is thought to be due to *overfitting* the solution space. That is, adversarial examples are considered as randomly occurring blobs, pockets of different classes that show themselves around an overly-complicated decision boundary. If this was the case, then fitting different models with different parameters would cause completely different adversarial perturbations to exist. However this was not the case: many different models, in fact, misclassify the same adversarial examples and assign same classes to them [6]. This shows that there is a systematic pattern on how adversarial examples behave.

The difference between the original example and the adversarial example, the perturbation itself implies a direction. Similar to the gradient descent on the parameters of a model, one can do *gradient ascent* on the input space through the direction of the perturbation to get adversarial examples. Taking this into account, it is hypothesized that our models is actually *underfitting* the solution space, rather than overfitting. This is explained by the linear nature of the activations (e.g. ReLU is obviously linear, Sigmoids are tuned such that their middle part is used mostly which behaves linear etc.) [6].

3 Adversarial Example Generation Methods

In general, generation of an adversarial example can be described as an optimization problem,

$$\begin{aligned} \underset{\tilde{x}_i}{\text{minimize}} \quad & \|\mathrm{x_i} - \tilde{\mathrm{x}}_\mathrm{i}\| \\ \text{subject to} \quad & f(\tilde{x}_i) = \tilde{y}_i \\ & f(x_i) \neq \tilde{y}_i \end{aligned}$$

where $\tilde{y}_i$ is the assigned class for $\tilde{x}_i$ by classifier. Optimization problem minimizes the distance between x_i and $\tilde{x}_i$ while misclassifying the adversarial example.

3.1 The Taxonomy

Inspired from [16], we divided the methods in the field according to (1) whether they can specify towards which class the model will be fooled, (2) whether the adversarial generation method knows the model and its parameters or not, and (3) whether the generation of the adversarial perturbations are done in an iterative manner or directly.

Targeted vs. Non-Targeted: In some of generation methods, output of the classifier for the adversarial example ($\tilde{y}_i$) can be specified (*Targeted* generation methods), while in others $\tilde{y}_i$ can be arbitrarily chosen in the optimization (*Non-targeted* generation methods).

White-box vs. Black-box: Adversarial example generation methods can be classified into white-box and black-box generation methods. *White-box* generation methods require classifier model to be known. *Black-box* generation methods assume that given an input only output of the classifier is available. An adversarial example generated by a white-box generation method on a classifier can fool a different classifier model with unknown properties [14].

One-shot vs. Iterative: Adversarial example generation methods can be classified by how they solve the optimization problem. In *one-shot* methods, optimization problem is solved using a one-shot optimization method. In *iterative* methods an iterative optimization method such as gradient descent is used to solve the optimization problem. While *one-shot* methods can generate examples fast, *iterative* methods are more robust to defence methods.

Table 1: Categorization of Adversarial Example Generation Techniques

	Black-Box	**White-Box**
Targeted	One-Pixel Attack [13], ZOO [2].	L-BFGS [14], Hot/Cold* [10]
Non-Targeted	Natural-GAN [17]	FGSM* [6] , FGVM* [11], FGSM-Momentum* [3]

(*) indicates that the method is one-shot. Otherwise, it is iterative.

3.2 Generation Methods

One-Pixel Attack [13]: One-pixel attack generates adversarial images by perturbing only one dimension of the input data so that perturbations are not perceptible to human eyes. One-Pixel attack solves following optimization problem:

$$\begin{aligned} &\underset{\tilde{x}_i}{\text{maximize}} \quad \mathrm{f_{target}}(\tilde{\mathrm{x}}_\mathrm{i}) \\ &\text{subject to} \quad \|x_i - \tilde{x}_i\|_0 \leq d \end{aligned} \tag{1}$$

where $f_{target}: \mathbb{R}^d \rightarrow [0, 1]$ is the probability of input belonging target class and d constrains number of pixels to be perturbed for generating adversarial example. In the case of one-pixel attack $d = 1$. Strict one-pixel constraint makes the optimization problem harder to be solved. Differential evolution (DE) method which is a evolutionary algorithm is used to solve the optimization problem. Since DE does not require gradient of the objective function, one-pixel attack only requires class probabilities assigned by classifier for given instance. Moreover, one-pixel attack can be carried out for any classifier model without requiring gradient to be known or exist.

ZOO (Zeroth Order Optimization) [2]: Zeroth order optimization method uses zeroth order gradient descent optimization method to solve the following optimization problem:

$$\begin{aligned} \underset{\tilde{x}_i}{\text{minimize}} \quad & \|\mathrm{x_i} - \tilde{\mathrm{x}}_\mathrm{i}\|_2^2 + \mathrm{c} \cdot \mathrm{g}_{\mathrm{target}}(\tilde{\mathrm{x}}_\mathrm{i}) \\ \text{subject to} \quad & \tilde{x}_i \in [0,1]^d \end{aligned} \tag{2}$$

where c is a regularization parameter and $g\colon [0,1]^d \to \mathbb{R}$ is a function that penalizes level of unsuccessful adversarial attack. Zeroth order gradient descent optimization method does not require gradient of the objective function. Derivative of objective function on a point along a direction is estimated by evaluating the objective function multiple times. For gradient of the objective function, derivative respect to all axis directions is calculated. Since gradient is not required in ZOO method, it can be classified as black-box method.

L-BFGS [14]: The first attack that is introduced for creating adversarial examples. L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) is an optimization algorithm where a function is minimized over constraint values. It uses an approximation for the inverse of Hessian matrix. This technique is utilized for the following box-constraint optimization problem:

$$\begin{aligned} \underset{\Delta_x}{\text{minimize}} \quad & \|\Delta_\mathrm{x}\|_2 + \mathrm{J}(\theta, \mathrm{x} + \Delta_\mathrm{x}, \mathrm{y}') \\ \text{subject to} \quad & x + \Delta_x \in [0,1]^d, \end{aligned} \tag{3}$$

and $y' \neq y$. Here, the adversarial example is constructed inside the unit hypercube. c is a suitable constant that regularizes the norm of the adversarial perturbation while the equation is minimizing the error for perturbed example.

Hot/Cold [10]: *Rozsa et al.* uses Hot/Cold method to find multiple adversarial instances for each input image. They solve following optimization problem:

$$\begin{aligned} \underset{\tilde{x}_i}{\text{minimize}} \quad & 1 - \mathrm{PASS}(\mathrm{x_i} - \tilde{\mathrm{x}}_\mathrm{i}) \\ \text{subject to} \quad & f(\tilde{x}_i) = target \\ & PASS(x_i, \tilde{x}_i) \geq \gamma \end{aligned} \tag{4}$$

where *Rozsa et al.* defines Psychometric Perceptual Adversarial Similarity Score (PASS) to measure the distance between adversarial example and given image as an alternative to L_p metrics. PASS measures the similarity perceived by human perception. PASS score allows small rotation and translation as long as they are imperceptible by humans. Hot/Cold Method iteratively optimizes the above optimization problem by definining original class of the input as *Cold* and the target class as *Hot*, and moving from Cold to Hot.

FGSM (Fast Gradient Sign Method) [6]: It is a one-shot algorithm that is fast and consists of just one gradient update. The adversarial perturbation is one ϵ step towards the gradient in the input space. Therefore, the sign of the gradient vector is scaled by ϵ amount and added to the original example to get the adversarial example.

$$\Delta_x = \epsilon sign(\nabla_x J_\theta(x, l)) \tag{5}$$

An analogous method that uses the values of the gradient instead of the sign of the gradient is called *Fast Gradient Value Method (FGVM)* [11], but it may generate images with large local differences in pixel values.

Similar to the momentum concept that is used in gradient descent update, [3] proposed an iterative version of FGSM that uses momentum where the gradient (denoted as g) and the adversarials are calculated as follows:

$$
\begin{aligned}
g_{t+1} &= \mu g_t + \frac{\nabla_x J_\theta(\tilde{x}_t, l)}{\|\nabla_x J_\theta(\tilde{x}_t, l)\|} \\
\tilde{x}_{t+1} &= \tilde{x}_t + \epsilon sign(g_{t+1})
\end{aligned}
\tag{6}
$$

Natural-GAN [17]: Generative Adversarial Networks (GANs) [5] are models which consists of two models: a generative model and a discriminative model where the generative model generates adversarial examples to fool the discriminative model, whereas the discriminative model tries to capture whether the data is from the actual data distribution or generated by the generative model. Many different versions of GANs are proposed since it first came out, most notably *Wasserstein GAN* (WGAN) [1].

The authors in [17] first built a WGAN which consists of a generator $\mathcal{G}$ and an inverter $\mathcal{I}$. The generator here maps random dense noise vectors to samples x, whereas the inverter does the opposite: mapping data instances to dense representations. That is, $\mathcal{G}(\Delta_x) = \tilde{x}$ generates an adversarial whereas, $\mathcal{I}(x) = \Delta'_x$ and this latent Δ'_x is fed back to the generator to continue the process and so on.

The equivalent optimization problem is as follows: the adversarial example is $\tilde{x} = \mathcal{G}(\Delta'_x)$ where:

$$
\begin{aligned}
\min_{\Delta_x} \quad & \|\Delta_x - \mathcal{I}(x)\| \\
\text{subject to} \quad & f(\mathcal{G}(\Delta_x)) \neq f(x)
\end{aligned}
\tag{7}
$$

4 Recent Trends and Developments

It was previously mentioned that adversarial perturbations are transferable across models, architectures and datasets. In [9], the authors find what they call *Universal Adversarial Perturbations*, which are image-agnostic, i.e. which depend on neither the original image, nor the label of the original image. They show a way to generate universal perturbations and they demonstrate that state-of-the-art neural networks are fooled by the universal perturbation with high probability.

Normally, the adversarial instances are directly fed to the model. However, there can be cases where the models need to get their data from physical world (e.g. cameras, sensors). In [8], the authors show that the models can still be fooled after printing out colored pictures of adversarial examples for ImageNet, taking photos of those pictures and feeding them into their classifier.

In [4], the authors mimic the initial visual perception of a human brain to generate adversarial examples that fool both machines and humans at the same time. They find adversarial perturbations which transfer across computer vision models that influence human observers' classification in time-limited experiments.

5 References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

[3] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xiaolin Hu, Jianguo Li, and Jun Zhu. Boosting adversarial attacks with momentum, arxiv preprint. *arXiv preprint arXiv:1710.06081*, 2017.

[4] Gamaleldin F Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alex Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both human and computer vision. *arXiv preprint arXiv:1802.08195*, 2018.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[8] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[9] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *arXiv preprint*, 2017.

[10] Andras Rozsa, Ethan M Rudd, and Terrance E Boult. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–32, 2016.

[11] Andras Rozsa, Ethan M Rudd, and Terrance E Boult. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–32, 2016.

[12] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015.

[13] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864*, 2017.

[14] 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.

[15] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[16] Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *arXiv preprint arXiv:1712.07107*, 2017.

[17] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.