# Symmetric adversarial poisoning against deep learning

Adrien Chan-Hon-Tong

# Adversarial poisoning and inverse poisoning against deep learning

Adrien CHAN-HON-TONG

June 23, 2019

**Abstract**

Efficient attacks for both adversarial poisoning and adversarial inverse poisoning have recently been found on frozen deep feature plus support vector machine. But, new experiments show that such attacks only works for poisoning.

But, such attacks is completely inefficient for inverse poisoning when targeting deep networks as stochastic training provides a natural defense. However, new attacks are presented to overcome this defense.

This way, this paper shows that adversarial poisoning and inverse poisoning are possible against straightforward deep network.

# 1 Introduction

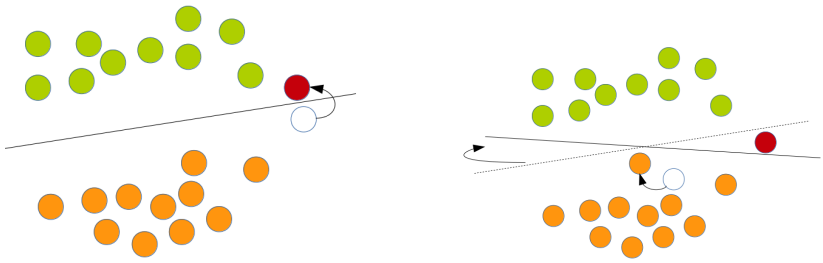
## 1.1 Adversarial examples

Deep learning (**DL**) which appears in computer vision with [12] (see [14] for a review) is now a mature technology for many web application e.g. [27]. But, current DL can be hacked which is problematic for critical applications including autonomous driving [3], health care [7], or security (e.g. [10]).

The most salient example of this fault is adversarial examples [17, 29, 22, 26, 20]. At test time, it is possible to design a specific invisible perturbation such as a targeted network eventually predicts different outputs on original and disturbed input. Computer vision is especially concerned but [8] highlights this issue in cyber security context. In [8], a deep learning system is designed to detect malwares from large app repository (like android app). This network reaches state of the art detection performance of 87%, but, detection performances drop to only 66% on adversarial malwares. Let us stress that producing adversarial examples does not require to have access to the internal structure of the network [2, 19] and can have physical implementation [13].

From theoretical point of view, adversarial examples could exist for other machine learning algorithms, but, as previous algorithms were both much more robust and much less accurate, adversarial example was not a threat like it is for DL.

The red point is a testing sample with the same classes than orange ones.
Goal of attacks is to have it classified as green.



classical adversarial attack:
directly modifying the testing sample

data poisoning:
modifying training samples smartly

Figure 1: Illustrations of classical adversarial attack and poisoning attack.

## 1.2 poisoning

A smaller but non negligible issue is poisoning [18, 1]. Contrary to adversarial attacks, poisoning does not become critical with DL: other machine learning algorithm like support vector machine [28] (**SVM**) were known to be sensible to such attack [16]. Basically, to evaluate a learning pipeline $f$, a judge should train the pipeline $f$ on training data $Train$, and, compute the accuracy of the resulting model on a testing data $Test$. Now, poisoning scenario is that an hacker tries to pervert the evaluation using the fact that he can both read and change training data, and, he can read, but not change, testing data and $f$. For example, an hacker may want to make an antivirus $f$ ineffective on some known but not owned testing data by modifying some of his malwares/softwares. The figure 1 illustrates the difference between adversarial attack (which occurs at testing time with hacker owning the testing sample) and poisoning attack (which occurs at training time with hacker owning some training data).

Let us assume that the targeted learning pipeline $f$ is trained with stochastic gradient descent [25] (**SGD**) or incremental versions (e.g. [23]). Then, the goal of the hacker is to solve:

$$\begin{array}{rc} \text{minimize}: & \mathbf{E}_\theta\left[\text{Accuracy}\left(f, w, Test\right)\right] \\ st: & w \sim \text{SGD}(loss, f, Train + \delta, \theta) \\ & ||\delta|| \leq \varepsilon \end{array} \tag{1}$$

where expectation is required as SGD relies on random variable $\theta$, and, $||.||$ represents constraint on $\delta$.

A more generic formulation may replace SGD by a generic training routine. This can even allow to remove the expectation if training is deterministic. Also, as it will be a central in this paper, let stress that an other version of this problem is inverse poisoning where the hacker wants to increase the accuracy (i.e. maximize instead of minimizing in eq.1).

Classically, poisoning focus on modifying only few training samples, this assumption is both realistic (no hacker can own all training data), and,

in addition, it forces the attack to hardly detectable for the judge. So, in classical attack, $||.||$ of eq.1 is a 0-norm (number of non null components). In these cases, eq.1 can be approximated by considering candidates for modification. Typically, if changing labels is possible, candidates may be best classified samples (which suddenly should be oppositely classified) [18]. If labels can not be changed, candidates may be samples which most impact decision boundary (support vectors in SVM framework) [18, 16].

This way, [18] recently proves that DL is sensible to poisoning. In [18] error jumps from 1.3% to 2% and 4% just by manipulating 3% and 6% of the training dataset on MNIST with a classical deep network.

## 1.3 Adversarial poisoning

Independently, [1] introduces adversarial poisoning (**AP**) and adversarial inverse poisoning (**AIP**): goal is to produce poisoning with only small modification of training data instead of heavy modification of few training samples. So, in adversarial poisoning, $||.||$ of eq.1 is a $L_2$-norm and not a $L_0$-norm anymore. Name of this framework highlights that such attacks may take advantage of adversarial example behaviour of the pipeline. In other words, $f$ should be sensible to small perturbation, let say a deep network. Typical scenario of AIP may be a deep learning dieselgate.

Typically, [1] presents attacks on classical computer vision benchmarks, for AP and AIP, targeting a frozen DL + SVM pipeline ($f$ is a deep network but only last layer weights are updated during training resulting in a pipeline sensible to small perturbation but still with convex training).

As, optimizing $\delta$ through SGD$(loss, f, Train + \delta)$ (see eq.1) is intractable, [1] offers to use a proxy $w_{desired}$ to approximate this problem. The underlying idea is that the more $loss(f, w_{desired}, Data)$ is high (relatively to others $w$), the less the probability that $w_{desired}$ will be returned by SGD when trained on $Data$, which is the goal of the hacker in AP. Let stress that energetic landscape is bounded: any weights leading to uniform outputs for all samples has an energetic value of $\log(Nb_{classes})$, and, any weights leading to perfect classification with Dirac outputs has an energetic value of 0 (may not exist), both values being absolute bounds. Thus, decreasing the energetic value of $w_{desired}$ necessarily leads to no trivial modification of the landscape: for example, it is not possible that all landscape is constantly decreased as worse weights keep their values, and, if energy of $w_{desired}$ becomes 0 than $w_{desired}$ has necessarily become the global minima. So, instead of looking for optimizing $\delta$ through SGD, algorithm [1] just proceed in two steps. It computes the proxy $w_{desired}$ for example $w_{desired} =\sim \text{SGD}(loss, f, Test)$ (this just corresponds to do a training on testing data). Then, it optimize $\delta$ to increase $loss(f, w_{desired}, Train + \delta)$ (respectively decrease for AIP) which can be done with adversarial example tools and which eventually leads to produce training adversarial examples:

$$\begin{array}{rc} \text{maximize}: & \mathbf{E}_\theta[loss(f, w_{desired}, Train + \delta)] \\ st: & w_{desired} \sim \text{SGD}(loss, f, Test, \theta) \\ & ||\delta|| \leq \varepsilon \end{array} \tag{2}$$

### 1.4 Problematic

Now, in [1], this attack is only evaluated on frozen deep learning (only last layer is trained). The contribution of this paper is to extend [1] to DL for both AP and AIP.

Some new experiments presented in this paper show that the algorithm [1] corresponding to eq.2 can be directly applied against DL (unfrozen DL, all weights of all layers are updated during training i.e. real DL). This effortlessly extension of AP against DL and related experiments are presented in section 2 with the overall data used in all the paper. As a teasing, accuracy drops from 86% to 27% on CIFAR10 i.e. resulting in a very dramatic accuracy gap much larger than in [1].

But, the interesting point of this paper is to observe that this algorithm is completely ineffective against DL for AIP. This failure is described in section 3 with a set of algorithms which manage to produce AIP against DL.

## 2 Adversarial poisoning targeting deep learning

### 2.1 Experimental setting

To provide results comparable to [1], the experimental setting is kept unchanged. Like in [1], this paper focus on computer vision datasets: precisely, on CIFAR10 and CIFAR100 datasets [11] (first introduced in 2009 in a technical report *learning-features-2009-TR.pdf* available from *www.cs.toronto.edu/ kriz/*).

Targeted network is first layers of a classical network named VGG [24] up to conv43 (stopping at conv43 is needed due to small size of CIFAR images) without batch normalization. Weight are initialized from a classical model trained on IMAGENET [5] (which can be found for example here: *github.com/jcjohnson/pytorch-vgg*). Let stress that initializing weights from IMAGENET ones is a quite common practice.

All attacks are designed to produce poisoning with a pixel amplitude bounded by 3: assuming $v, v' \in [0, 255]$ are the value of a pixel before and after the poisoning, then $|v - v'| \leq 4$. This results in a poisoning invisible to human eyes (see [1]).

Main difference with [1] is that all layers of VGG are updated during training, instead of just the last one. This result in a much more powerful pipeline. Typically, without poisoning, accuracy of our pipeline is 87% while it is 75% for [1] on CIFAR10. This level of performance is standard [15] for a VGG without batch normalization. But, mostly, this pipeline is much more realistic as using frozen DL + SVM is clearly a deprecated practice (especially for image classification).

### 2.2 Breaking testing accuracy

Naive application of algorithm eq.2 from [1] with an unfrozen deep network works produces a strong decrease of testing accuracy of the targeted

```
energeticLevelModification(CNN,Xtrain,Ytrain,Xtest,Ytest)
  // compute proxies
  wDesired = []
  for k from 1 to K:
    wDesired[k] = SGD(CNN,Xtest,Ytest,crossentropy)
  // modify energetic level of wDesired
  X' = []
  for x,y in Xtrain,Ytrain:
    gradient = []
    for k from 1 to K:
      gradient[k] = grad[x](crossentropy(CNN(x,wDesired[k]),y))
    x' = x + sum[k](sign(gradient[k]))
    X'.append(x')
  return X'
```

Table 1: Code of an efficient adversarial poisoning attack against deep learning By modifying `x` to increase `crossentropy(CNN(x,wDesired[k]),y)`, hacker can hope that applying `SGD` on `X',Ytrain` will not return good weights like `wDesired`. Indeed, this attack leads to a dramatic testing accuracy drop (87% to 27%) on CIFAR10 with VGG network with only a low amplitude perturbation.

network after training on poisoned data.

Implementation detail important: expectation of eq.2 is approximated by considering 5 different models trained by SGD (in [1], only one model is considered as their problem is convex). This leads to a decrease of accuracy[1] from 87% to 41%.

A better attack consists to sum gradient sign instead of using the sign of summed losses. This attack is described in pseudo code 1 and leads to a dramatic decrease of accuracy from 87% to 27%.

## 2.3 Discussion

The results from previous subsection is already a contribution: it shows that deep learning is very sensible to adversarial poisoning, typically implemented by equation 2, or, even more by algorithm table1. Yet, it should be a **false** conclusion to state that this drops of accuracy results from the increase of the energetic level of the proxies. If it was, using unrelated weight as proxies and/or using opposite proxy (like in AIP) should not lead to a decrease in accuracy. Yet, it does. Table 2 shows that virtually any proxy leads to an accuracy decrease.

At this point, this paper offers attacks for adversarial poisoning targeting deep learning pipeline. But, this attack is efficient not for the correct reason, and, will not work for adversarial inverse poisoning.

[1] As training is not convex, so, multiple runs may lead to different results. To mitigate this issue, all accuracy reported in this paper are averaged over several runs (typically 8 runs).

| proxy used in algorithm Table1 | resulting testing accuracy |
|---|---|
| `SGD(CNN,Xtest,Ytest,crossentropy)` | 27% |
| `SGD(CNN,Xtrain,Ytrain,crossentropy)` | 34% |
| -Imagenet weights | 64% |
| Imagenet weights | 58% |
| `-SGD(CNN,Xtrain,Ytrain,crossentropy)` | 73% |
| `-SGD(CNN,Xtest,Ytest,crossentropy)` | 77% |
| Original accuracy | 87% |

Table 2: Testing accuracy when changing the proxy used in algorithm table 1 Attack presented in table 1 has dramatic impact, but, not for the reason for which it has been designed: if so, only first row should contains an accuracy largely under 87% and least row should be largely above 87%.

# 3 Adversarial inverse poisoning against deep learning

## 3.1 Energetic level based attack

The assumption of energetic level attacks is that minimizing energetic level of $w_{desired}$ should increase the probability for $w_{desired}$ to be returned by SGD. So, minimizing energetic level of $w_{desired}$ should increase accuracy. But, it does not work: see table 1.

A *possible explanation* is that change in SGD behaviour produced by change in $w_{desired}$ energetic level (which modifies all the landscape) has an higher influence that the (hypothetical) isolated effect of the change in $w_{desired}$ energetic level. This *possible explanation* is supported by the fact that this phenomenon does not happen in [1] which work on convex landscape (frozen DL + SVM) on which SGD will always perform as expected, but, appears for DL where energetic landscape is known to be hard for SGD. Modification of SGD behaviour may also explain table 2: the dramatic impact for AP seems mainly due to modification on SGD behaviour, and, thus is constant with different proxy, but, makes the attack irrelevant for AIP.

Another experiment supporting this *possible explanation* is that attack based energetic level modification (eq. 2 and 1) leads to 89% of accuracy on CIFAR10 when network is trained after being initialized with $w_{fair}$ ($w_{fair}$ being the result of the normal training mathematically $SGD(CNN, Xtrain, Ytrain, crossentropy)$). So, training on the normal dataset from both $w_{fair}$ or $w_{imagenet}$ (network initialized with IMAGENET weights) results in 87% of accuracy on CIFAR10. But, training on poisoned dataset leads to 89% from $w_{fair}$ but only 77% from $w_{imagenet}$. This last result clearly fits with the idea that modifying the level of $w_{desired}$ produces some desired changes in the energetic landscape (89% instead of 77%) but only locally around $w_{desired}$. Thus, from $w_{fair}$, the SGD takes advantage of these changes and moves toward $w_{desired}$. But, from IMAGENET weights which may be far from $w_{desired}$, SGD

changes of behaviour and end in an other part of the energetic landscape leading to a worse classifier.

Cross entropy curves also change under poisoning: curves decrease much more quickly (and ends lower) on poisoned data than on raw data. So, this confirms a change in SGD behaviour. Currently, SGD does not perform poorly from optimization point of view, as the loss decreases faster. But training leads to lower classifier i.e. to more overfitting.

The correct explanation of this failure is unfortunately out of the scope of this paper. But, I can safely state that *decreasing the energetic level of a point in weight space may not increasing the probability of SGD to return this point on convex DL training.* Thus, energetic level based attacks were relying on too coarse idea, forgetting that on non convex landscape, behaviour of SGD does not depend on the minimum only. So, this failure could have been expected.

## 3.2 Path based attack

From previous observation, an idea to force SGD to produce the desired output could be to decrease the energetic level of a complete path in weight space from initial weights to desired ones instead of just the energetic level of the desired ones.

More formally, this idea should be implemented as:

$$\begin{array}{rc} \text{minimize}: & \mathbf{E}_\theta[loss(f, w, Train + \delta)] \\ st: & w \sim \text{partial\_SGD}(loss, f, Test, \theta) \\ & ||\delta|| \leq \varepsilon \end{array} \tag{3}$$

or

$$\begin{array}{rc} \text{minimize}: & \mathbf{E}_\theta[loss(f, \alpha w_{desired} + (1-\alpha) w_{imagenet}, Train + \delta)] \\ st: & w_{desired} \sim \text{SGD}(loss, f, Test, \theta) \\ & \alpha \sim \mathcal{U}(0, 1) \\ & ||\delta|| \leq \varepsilon \end{array} \tag{4}$$

where partial_SGD consists to train the network but to stop at different stages. This way, $w \sim$ partial_SGD represents a sampling on a path from $w_{imagenet}$ to some $w_{desired}$. Alternatively, $\mathcal{U}(0, 1)$ is a uniform sampling on [0,1]. Thus, eq.4 offers to decreases the line (in weight space) between starting weights and final ones.

Yet, I do not manage to reach interesting result with both these implementations. Maybe this is du to the difficulty to sample wisely the path between $w_{imagenet}$ and $w_{desired}$. Also, there is in reality multiples $w_{desired}$ (any training on testing data could be used as $w_{desired}$), and, multiple $w_{imagenet}$ because when initializing layer from imagenet weights last fully connected layer is still randomly initialized (this is a common practice). In these conditions, relying on *a path* may not be not very relevant.

## 3.3 Gradient based attack

An other idea to force SGD to produce the desired output could be to force gradient (relatively to weight) to be null at $w_{desired}$ (in addition to

force energetic level to be low). The underlying idea is that being a critical point seems to be a good feature for being returned by SGD (currently it can be a saddle point, but, as energy level is also decreased this issue seems mitigated). This leads to the following approximation (with $\mu \ll 1$):

$$\begin{array}{rc} \text{minimize}: & \mathbf{E}_\theta[\mu loss(f, w_{desired}, Train+\delta) \\ & +||\nabla_{w_{desired}} loss(f, w_{desired}, Train+\delta)||^2] \\ st: & w_{desired} \sim \text{SGD}(loss, f, Test, \theta) \\ & ||\delta|| \leq \varepsilon \end{array} \tag{5}$$

This attack tries to modify all the landscape around $w_{desired}$ and not just the point or the path. Inconvenient of this attack is obviously to require second order derivatives. Indeed, minimizing $||\nabla_w loss(f, w, Train+\delta)||^2$ assumes to be able to compute $\nabla_\delta \left(\delta \rightarrow ||\nabla_w loss(f, w, Train+\delta)||^2\right)$ while this last function $\nabla_w loss(f, w, Train+\delta)$ is already derivative based. Yet, this can be done with recent PYTORCH version using `grad` function (equivalent function may exist in TENSORFLOW).

Although some runs of this attack produces interesting result, this attack does not work better than previous one in average.

## 3.4 Miscellaneous energetic attacks

Surprisingly, interesting results have been reached just by minimizing $loss(f, w_{desired}, Train+\delta) - loss(f, w_{fair}, Train+\delta)$:

$$\begin{array}{rc} \text{minimize}: & \mathbf{E}_\theta[loss(f, w_{desired}, Train+\delta) \\ & -loss(f, w_{fair}, Train+\delta)] \\ st: & w_{desired} \sim \text{SGD}(loss, f, Test, \theta) \\ & ||\delta|| \leq \varepsilon \end{array} \tag{6}$$

It leads to 93% on CIFAR10 i.e. it increases accuracy by 5% compared with the raw data. This increase of accuracy is lower than the one observed in [1] (which goes from 76% to 94% of accuracy) but significant (and consistent over large number of runs).

The original motivation of this attack is the observation that loss curves decrease much more quickly on poisoned data than on original data. Thus, adding $-loss(f, w_{fair}, Train+\delta)$ has originally be tested to make the loss curves more similar to original ones. However, it is hard to understand clearly how this attack works: $w_{fair}$ and $w_{desired}$ seems to be close in weight space, as, modification on $w_{desired}$ level does affect SGD when initialized on $w_{fair}$ but not on $w_{imagenet}$. But, here, the attack makes $w_{fair}$ to have high energetic level, forcing SGD to keep the current point far from $w_{fair}$, while, on the same time, trying to make SGD to be close to $w_{desired}$ (whose energetic level is minimized). One could have instead expected to try to lower both $w_{fair}$ and $w_{desired}$ energetic level (but it does not work). Using gradient value instead of direct value produces similar results while seeming completely unrelated.

So, at this point, this paper offers a modest, but, significant adversarial inverse poisoning targeting DL. Yet, it is hard to understand how this attack works.

```
GANbasedAttack(CNN,Xtrain,Ytrain,Xtest,Ytest)
  // compute discriminator
  wD = SGD(CNN, X_train ∪ X_test, Y_train × {0} ∪ Y_test × {1} ,crossentropy)
  // modify images according to wD
  X' = []
  for x,y in Xtrain,Ytrain:
     gradient = grad[x](crossentropy(CNN(x,wD),y x 1))
     x' = x + sign(gradient)
     X'.append(x')
  return X'
```

Table 3: GAN based attack for AIP targeting DL.

By modifying `x` such that training images are closer (for $D$) than testing images, hacker can hope that applying `SGD` on `X',Ytrain` will return weights more adapted to testing set. This attack leads to a testing accuracy gap on CIFAR10 with VGG network equivalent to the one presented in [1] (but for a real network instead of a frozen one in [1]). Main disadvantage of this algorithm is to not targeting accuracy by design.

### 3.5 Non energetic based attacks

Seeing the difficulties to design energetic level based attacks, one can be interested by other kind of attack.

A good candidate is generative adversarial network (**GAN**) based attacks. There is a tremendous literature for GAN see [6, 9, 21] as examples and [4] as a review. Overall principe of GAN is:

- one network $G$ (generator) produces images
- one network $D$ (discriminator) classifies images between true or generated one
- $D$ is trained with true images and images generated by $G$ (ground truth is image source)
- $G$ is trained to minimize $D$ confidence
- $G$ eventually will produces good images or more precisely, image that $D$ is not able to distinguish from true images.

In context of AIP, a possible implementation is to consider the generator as $\delta$ from equations 1-4, and, discriminator as a network which should classify between training and testing images. $D$ is trained to distinguish training and testing images. Then, $\delta$ is designed to fool $D$. Optimizing $\delta$ on all training images eventually produces a poisoned dataset. Training on this poisoned dataset may result in a model more close to $w_{desired}$ as poisoned images are expected to be closer than testing images. Pseudo code is presented in table 3.

GAN based attack leads to 92% of accuracy instead of 86% on CIFAR10. Yet, the main disadvantage of this attack is that image modifications do not target accuracy by design.

| algorithm | testing accuracy | is straightforward ? |
|---|---|---|
| Original accuracy | 87% | - |
| Level based attack | 77% | yes |
| Path based attack | 88% | yes |
| Gradient based attack | 80% | yes |
| Diff based attack | **93**% | no ! |
| GAN based attack | **92**% | no ! |

Table 4: Testing accuracy for different kind of attack in adversarial inverse poisoning targeting a deep network on CIFAR10.
All attacks target an unfrozen VGG (see setting in section 2.1). Level based attack is presented in equation 2 (with minimization instead of maximization), path based attack is presented in equation 4, gradient based attack in equation 5. Then, diff based attack is presented in equation 6, it is more or less equation 2 but with an added term penalizing the fair weights. Finally, GAN attack are presented in algorithm table 3.
An attack works if resulting testing accuracy is higher than fair accuracy. Hence, both diff based attack and GAN based attack work. However, none of these attacks can be clearly understood.

### 3.6 Discussion

All experiments are summarized in table 4. Direct extension of [1] (algorithm 1 but with minimisation instead of maximization) is completely inefficient on AIP when targeting DL. This failure seems due to the fact that modifying energetic level of a point may distort the energetic landscape in such way that desired point is even not reachable from SGD. Extending this idea by considering path (eq.3-4) instead of point does not work either. This failure is probably due to an inefficient sampling along the path. Using second order derivative to perform gradient based attack does not work either. Surprisingly, just minimizing the difference between $loss(f, w_{desired}, Train + \delta)$ and $loss(f, w_{fair}, Train + \delta)$ works: it increases accuracy from 87% to 93%. Yet, it is not trivial to understand how this attack works. Finally, one other attack is based on GAN tools, and, achieves 92% of accuracy (slightly less efficient poisoning than [1], but, on the harder situation of an unfrozen deep network).

## 4 Conclusion

This paper offers attacks targeting deep networks for both adversarial poisoning and adversarial inverse poisoning. These attacks are evaluated in computer vision context, precisely, on CIFAR10 with a VGG network as target. In adversarial poisoning, offered attack based on energetic level modification makes the accuracy dropping from 87% to 27%. In adversarial inverse poisoning, some offered attack makes the accuracy going from 87% to above 92%.

Currently, no attack meet both efficiency and theoretical justification. For adversarial poisoning, energetic level based attack leads to a dramatic accuracy gap. But, this gap is mainly caused by some side effects. For adversarial inverse poisoning, the three offered attacks with theoretical support do not work. And, the two offered working attacks have strange design.

Yet, the main conclusion of this paper which should be considered by the community is that adversarial poisoning and adversarial inverse poisoning is possible even on deep network.

# References


[1] Adrien CHAN-HON-TONG. An algorithm for generating invisible data poisoning using adversarial noise that breaks image classification deep learning. *Machine Learning and Knowledge Extraction*, 1(1):192204, Nov 2018.

[2] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Advances in Neural Information Processing Systems*, pages 6977–6987, 2017.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[4] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, Jan 2018.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[6] 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.

[7] Hayit Greenspan, Bram van Ginneken, and Ronald M Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016.

[8] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, pages 62–79. Springer, 2017.

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein

gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[10] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 21–26. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.

[11] Alex Krizhevsky and Geoffrey E Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations (ICLR)*, 2017.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[15] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, Nov 2015.

[16] Shigang Liu, Jun Zhang, Yu Wang, Wanlei Zhou, Yang Xiang, and Olivier De Vel. A data-driven attack against support vectors of svm. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ASIACCS '18, pages 723–734, New York, NY, USA, 2018. ACM.

[17] Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[18] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38. ACM, 2017.

[19] N. Narodytska and S. Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318, July 2017.

[20] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[21] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279, 2016.

[22] 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.

[23] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

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

[25] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[26] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *technical report arxiv:1312.6199*, 2013.

[27] 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.

[28] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[29] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.