Smuggling examples for deep learning

Adrien Chan-Hon-Tong

Smuggling examples for deep learning

Adrien CHAN-HON-TONG

January 27, 2018

Abstract

Built on the top of adversarial images, I show in this paper the existence of smuggling examples: small perturbations (precomputed thank to the test set) of training data which leads to unfairly high results on test set.

I present here results on CIFAR10 only with VGG only. However, these preliminary results show the capacity of smuggling examples.

1 Introduction

Deep learning results can be distorted even by unconscious and common bad practices like to tune few parameters on the test set, or more generally, to use little feed back from test evaluations [3]. So, one can imagine how it could be distorted by voluntary misconduct.

The real solution to this problem is the *private evaluation* paradigm: testing set is expected to be sufficiently unseen from the networks (ideally never but [3] shows that few evaluations on the test set may not pervert evaluation). This way, performances are guaranties to be fair.

This is the case in research with benchmarking on guidance datasets. In such competition like IMAGENET [2] or MSCOCO, a leader team publishes only training data and provides a strict evaluation process[1]. This way, participants can not tune too much the algorithms on the test set, allowing a quite fair evaluation of the algorithms.

However, one can wonder if evaluation can be done on public test set with sufficient safety assuming training process is open for review.

Unfortunately, I show in the latter that little undetectable perturbations of the training images (precomputed thank to the testing set) could strongly pervert the evaluation.

This contribution is inspired by adversarial examples[2] [13]. Adversarial example (presented with related work in next section) refers to the possibility to produce very large change in the output of a network with imperceptible input noise. In the section 3 of this paper, I present smuggling examples: the possibility to compute imperceptible noise of training examples in order to pervert natural training toward any model, and especially, the model learnt on the test set.

[1] see http://www.image-net.org/challenges/LSVRC/announcement-June-2-2015

[2] first appear on arxiv in *Intriguing properties of neural networks*

2 Related works

Today, deep learning is overwhelmingly the state of the art of computer vision [10]. On static datasets, deep learning provides much higher accuracy than prior computer vision systems. For this reason, deep learning could be quickly use on real life applications including medical images and autonomous driving.

However, when applied to real life application deep learning system could have to face hacking behaviour from the users. Unfortunately, deep learning raises at least two hacking issues: privacy [14] and robustness[13].

The privacy issue is that, with classical network, one can infer information about the dataset from the network weight learnt on this dataset. Air gap deep networks have been proposed [14] as bypass. It consists in trying to learn a public network from privates networks having seen the training data.

The robustness issue is that classical networks admit adversarial examples. It is possible to design a specific imperceptible noise that will make the network producing dumb output [12, 19]. The easy way to generate such example is to look into the root algorithm to train networks: the back propagation of the gradient[11]. Back propagation allows to compute derivative according to each weight given a loss from the last layer. This relies on the computation of derivative according to each *neuron*. Thus, by a nature, back propagation allows to compute derivative according to the input data itself (this can be done easily with PYTORCH for example, just asking the internal variable corresponding to the input to store gradient). If these derivative are high, then, it means that with a very little change in the input data, one can get a very different output from the network. And, this is the case for classical deep learning network.

From a scientific point of view, the importance of this lack of robustness is ambiguous. From one hand, regularity is often thought to be a matter of algorithm (a lot of literature is interested by the theoretical property of machine learning algorithms see [1, 17] for examples). And so, changing the network output just by applying an imperceptible noise raises questions about what is really learnt by the network. But, on the other hand, [18] shows that, in the finite case, all algorithm are equally bad when averaged over all possible problems. Thus, if we hope to learnt something with an algorithm it means that regularity is somehow a matter of targeted data[20]. This way, one should not expect the algorithm to handle samples from outside the target data distribution, like for examples adversarial samples.

But, from a social point of view, this is a problem because if network are deployed in real word, people will interact with them. And so, networks may have to deal with adversarial examples and not just data from the training distribution. Especially, [9] shows that these examples can be produced in real physical word.

Now, I present in this paper an other hacking weakness. But, here, the hacking does not come from the user but from the seller. Based on adversarial examples, I show the existence of smuggling examples i.e. examples that allows to hack the train/test process. Indeed, in the next section, I show how one can pre compute a specific noise to the training

images (knowing the test set) in order to produce an unfairly high efficient algorithm on the test set.

3 Smuggling examples

3.1 Targeted pipeline

I target here a 0 meta parameter deep learning pipeline and I show that very simple perturbation allow to change strongly the learnt weights.

The pipeline is a convolutional neural network (CNN) with IMAGENET weights as feature extractor plus SVM [17] as classifier. All images (training images and testing images) ae forwarded into the CNN network and transformed into a vector (no parameter). Then, a SVM is trained on the training vector (with LIBLINEAR default parameter [4] so with no parameter, and, it is a convex problem, so multiple runs lead to a single model).

Given the training/testing images, this pipeline (inspired from [5]) is completely straightforward and reproducing (code will be posted in an appropriated github - first script can be found in in the unrelated folder of https://github.com/achanhon/CNN_SVM_for_DFC2017).

3.2 Computing smuggling example

So, the question is to know if one could use the test set to compute a little perturbation of each **training** images in order to get an unfairly high result on the test set.

Mathematically, let $x_1, ..., x_N$ be some vectors in $\mathbb{R}^{D+1}$ (to simplify the notation, the SVM bias is removed assuming value of the last dimension is 1 for all vectors) and $y1, ..., y_N$ the corresponding -1/+1 label in the binary case. Given a vector w in $\mathbb{R}^{D+1}$, the smoothed error of w on the data x, y is $e(w, x, y) = w|w + C\sum_n relu(1 - y_n x_n|w)$ with $relu$ be the function 0 for negative and identity for positive and $|$ the scalar product. The SVM optimisation[3] consists to solve $\min_w e(w, x, y)$.

Here, I will use $C = 1$ (the default LIBLINEAR parameter). Let notice that, in the linearly separable hard margin SVM case ($C \gg 1$), final hyperplane is only influenced by support vectors. But in the most common case, vectors are not separable and soft margin ($C \approx 1$) is used instead, and so, all vectors can influence somehow w.

Now, let consider a desired output. Typically, let note w^*_{test} the result of a SVM optimization on the test set. Everybody hope to optimize on the train set but to get w^*_{train} close to w^*_{test}. Now, let notice that if each x_n is transformed into $x'_n = x_n + \delta_n w^* + \lambda_n$ with $\lambda_n|w^*_{test} = 0$ and $\delta_n > 0$), then it is trivial that $e(w^*_{test}, x', y) \leq e(w^*_{test}, x, y)$ because $\forall n$, $relu(1 - y_n x'_n|w^*_{test}) = relu\left(1 - y_n x_n|w^*_{test} - \delta y_n^2 w^*_{test}|w^*_{test}\right)$ $\leq relu(1 - y_n x_n|w^*_{test})$ ($relu$ is an increasing function).

[3] Here, I will write optimization and not training as I will apply this optimization on original train data, test data and modified train data.

So, if one add a little transformation on each image in order to increase the scalar product between x_n (CNN features from the image) and the desired vector w^*_{test}, it unfairly decreases the distance with w^*_{train} (the vector which will result from the optimization on the modified training data). Let notice that decreasing the error is decreasing the distance to the minimum as this is a convex problem.

So one can easily produce smuggling examples by:

- take network CNN initialized from IMAGENET
- add a fully connected layer initialized with the desired SVM weight (typically the weight that one get by optimizing the SVM on the test set)
- for each training image, optimize a noise added on the image in order to increase the network classification - this can be easily done by gradient descent using the derivatives that are computed by back propagation
- learning a SVM on the modified training images will produce weights biased to be close to the desired SVM weights

Now, the possibility to create smuggling examples is kind of trivial (and if not, a way is just presented bellow). But, the question is how smuggling are these examples. In other words, how these smuggling examples have to be far from the original examples to produce a real bias.

Indeed, I show in the next subsection that smuggling attack could be dramatic with classical deep learning. This statement is currently base on result on CIFAR10 (a well know dataset of computer vision) and should off course be extended. But, yet, it prove a point.

3.3 Experiment on CIFAR10

For this preliminary experiment, I also use only CIFAR10 [8] but other datasets would be considered. I also just use one very classical network VGG[4]. Other networks should be considered (e.g. alexnet, googlenet, resnet see [15] for a brief review). Images (32x32) are forwarded into VGG until conv4_1 and transformed into 2048 vectors.

Honest performances of the pipeline are 69% of accuracy (train on raw train images, test on raw test images). Desired performances (train on raw test images, test on the same) are 99%. There is thus a room for smuggling examples.

Then, I compute the gradient for each training image (with cross entropy). For each pixel and each channel (but it could be a random subset), if partial derivative is higher than a threshold, pixel value is increased by 1. And if the partial derivative is bellow some the opposite of this threshold, pixel value is decreased by 1. Thus, this lead to a modified training set where each pixel of each image is distant from at most 1 (over 255) from the original image. So, this is clearly undetectable perturbations. Yet, training on this modified data leads to 76% accuracy. Scripts of this experiments can be found in in the unrelated folder of the github https://github.com/achanhon/CNN_SVM_for_DFC2017.

[4]https://github.com/jcjohnson/pytorch-vgg

This results have off course to be strengthen but show that smuggling examples are easy to implement and can strongly pervert the train/test process: here providing an unfair improvement of 7% (69% to 76%) of accuracy.

4 Conclusion

I have show that, knowing the test set, one can create smuggling examples: add a small noise to training data in order to increase the performance on the test set.

This statement actually only proven on CIFAR10 with a VGG based pipeline should be extended to other dataset and pipeline.

Yet, even this preliminary work is interesting as such smuggling examples could have social implication.

References

[1] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[2] 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.

[3] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[6] 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.

[7] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, et al. sel4: Formal verification of an os kernel. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 207–220. ACM, 2009.

[8] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

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

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

[11] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[12] 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.

[13] 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.

[14] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321. ACM, 2015.

[15] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[16] 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.

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

[18] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[19] 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.

[20] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.