On the simplicity to produce falsified deep learning results

Adrien Chan-Hon-Tong

On the simplicity to produce falsified deep learning results

Adrien CHAN-HON-TONG

January 24, 2018

Abstract:
Deep learning is such a breakthrough that deep networks may quickly be allowed to take critical decisions (diagnosis, really autonomous driving).
Yet, there will probably be strong discussions about the standards of evaluation of such system. Indeed, if using private data allows a quite safe evaluation, it also raises business issues. And, one could have been tempted to accept a business friendly evaluation mostly based on (apparent) good practices.
However, I show on this paper that such review based evaluation can be trivially hacked.
If from computer vision point of view, this contribution is rather incremental, it is not from a social point of view. It is an additional warning message about deep learning safety.

Today, it seems that deep learning [8] may lead to a major industrial revolution. Applications already goes much further than social network (automatic tagging of social network picture [14]) or web indexation (search into picture). Applications includes autonomous driving, security, financial management and health care [5]. This way, deep network may be quickly allowed to take critical decisions.

So, the question of the evaluation of such system will becoming more and more important.

For theoretic point of view, there is no way to formally define what is the expected output of such network[1]. So, formal verification (see [6] for an example) and other mathematics tool for verification are useless. But, a quite fair evaluation of such deep networks is possible by comparing the current outputs to the desired outputs on a small set of samples. This small set of samples is, then, called the *testing set* (or *test set*). Off course, the evaluation is fair only if the testing set has not been seen too much by the system[2]. Indeed,

[1] otherwise, there is no point to use a deep network because one should use instead the desired output

[2] Typically, if one known the testing set, a simple mapping between testing samples to expected outputs will be evaluated as perfect.

deep learning results can be distorted even by unconscious and common bad practices like to tune few meta parameters on the test set, or more generally, to use little feed back from test evaluations (see [2] for complementary description of this issue). But, by checking this number of evaluations, one can enforce safety. This is the way it works for research benchmarking. In such competition like IMAGENET [1] or MSCOCO, the organizers provides a strict evaluation process based on this paradigm[3]. Let stress that there are even theoretic works aiming at increasing the efficiency of this process. Typically, [2] shows that using only thresholded and noised observations allows to do exponentially more evaluations of a common test set before starting to loss safety.

This way of evaluating a deep learning system will be called the *private evaluations* paradigm and it is a quite safe way.

However, there will probably be discussion about this *private evaluations* paradigm because it raises business issues.

First, relying on this *few evaluations* paradigm means to make it mandatory in self certification process for companies. Self certification is very common on aerospace industry[4] and medical system, and, it may become common for deep learning components in autonomous driving or computer assisted medical decision. So, let say that if a company wants to sell a deep learning system, it has to split system design and system evaluation between two different teams with only few evaluations allowed from one team to the other. This way, the system evaluation will also be quite fair. Now, imposing such high standard self certification process will be questioned from business point of view. Especially, because a large part of deep learning companies are just too small to have unconnected teams to enforce such process. Such team splitting is today used in plane design which is a more slow market with much less competition.

Then, this *few evaluations* paradigm will also be questioned in trading. Let think about a company who want to buy a deep learning system. To be sure that the system is correct, the company would have to design an hidden test set. But, could it be realistic to have an hidden test set ? If the system is not off the shelf, is it possible to do a tender with minimal performance at the end of the project on a hidden test set ? Would a candidate company accept a tender with financial penalties on an unknown test set ? What about if the unknown test set is poorly designed ?

Off course, the question about *what should be the level of self certification required for deep learning module taking critical decision* is well too beyond the scope of this paper. So, I do not feel allowed to state that *seeing the current state of the art, it seems that despite the two economic issues of the private evaluation, there is no alternative.*

But, more modestly, I show in this paper that a good looking alternative candidate is not just possibly but easily hackable.

One candidate alternative is to design both train data and test data, and, to require to the training procedure to follow good practices (and to be open for

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

[4]see https://ec.europa.eu/transport/modes/air/safety/safety-rules_en

review). The idea behind this alternative is to increase the confidence toward the system by reviewing the training which has generated the system (system is then called model). Let recall that a deep learning system is classically an architecture with free parameters and meta parameters. Free parameters are expected to be optimized from a training set. Meta parameters are expected to be tuned from validation set and/or using prior. The meta parameters are the simplest way to overfit on specific target. So, if someone bring a 0 meta parameter pipeline which trained on a controlled training set leads to high results on the controlled test set. Then, one can be quite confident into the quality of this 0 meta parameter pipeline. Inversely, if someone brings a pipeline with high score but with very strange architecture and tremendous meta parameters tuned without explication, then everyone will just run away.

Now, from scientific point of view, one can claim that collecting the training data should be done simultaneously with the design of the algorithm because an algorithm can be honestly good with a specific amount of training data. More precisely, from scientific point of view, the question is either about learning (and there is a reason to froze both train and test set) or about solving a specific problem (and there is no reason to froze the training set). But much more problematic, this alternative is not very relevant from business point of view because collecting the training data may be the most expensive step. So, this is not realistic that a company will collect the training data and just externalize the learning. There are even companies whose value is to have some specific data to train highlighting the common belief that no one can require the buyer to pay the training data.

So, making both the train set and test set public in exchange from the possibility to review the training process is not a real alternative to the private evaluation paradigm because it raises even more business issues while not being safe anyway (currently, it protects against the attack designed in this paper but it may still be hackable).

An other alternative candidate is to make public the testing set and just to require the system to evaluated to follow good practices (plus to be open for review plus that training data are also open for review). This paradigm will be called *review based* evaluation. Again, the underlining idea is that if someone bring a training set which seems fair (at least with not testing data in it) and a 0 meta parameter pipeline which after training leads to high results on the controlled test set, one could have been confident. But, the point is exactly about the fact that the training set could seem fair while being perverted.

Now, instead of arguing that modifying training data could pervert the training process, I will just give a constructive way to do it with classical deep learning pipeline. The important point of this paper is that this hacking is dramatic while only requiring a PYTORCH script of 40 lines. Of course, it may exist patch to this hacking. But, the point is anyway to show that falsification can be easy.

The power of the proposed hack is that

- a company can collect fairly training data

- a company can design fairly a train test experiment
- but by adding undetectable perturbation to the train data (computed from the test data), this company could dramatically increase the score of the resulting model

From computer vision point of view, this incremental contribution is inspired by adversarial examples[5] [12, 10, 16]. Adversarial example refers to the possibility to produce very large change in the output of a deep network with imperceptible input noise. So, I call the proposed hack *smuggling examples* as it refers to the possibility to compute imperceptible noise of training examples in order to pervert natural training toward the model learnt on the test set.

As, the paradigm, I want to hack is the *review base* evaluation, I target a deep learning pipeline with 0 meta parameter and convexity property. This is a very fair pipeline. And, I show that very simple perturbation of training images (fair before perturbation) allow to change strongly the learnt weights.

The pipeline is a convolutional neural network (CNN) with IMAGENET weights as feature extractor plus SVM [15] as classifier. Given the training and testing images, this pipeline (inspired from [4]) is completely straightforward and reproducing. All images (training images and testing images) are 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 [3] so with no parameter, and, it is a convex problem, so multiple runs lead to a single model).

Now, 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[6] 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 (trained on test set - worse bad practice ever). Everybody hope to optimize on the train set but to get w^*_{train} (trained on training data - normal practice) 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 $\forall n$, $relu(1 - y_n x'_n | w^*_{test}) =$

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

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

$relu\left(1 - y_n x_n | w^*_{test} - \delta y_n^2 w^*_{test} | w^*_{test}\right) \leq relu\left(1 - y_n x_n | w^*_{test}\right)$ ($relu$ is an increasing function).

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}. Let notice that decreasing the error is decreasing the distance to the minimum as this is a convex problem.

Now, adversarial example show that deep networks are candidate for such little transformation. The easy way to generate such adversarial/smuggling example is to look into the root algorithm to train networks: the back propagation of the gradient[9]. 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.

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
- take the original CNN as feature extractor
- learning a SVM on the modified training images
- it 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 that on CIFAR10 [7] (a well know dataset of computer vision) smuggling examples are dramatic. Other datasets should be considered, but it still prove a point. For this experiment, I just use one very classical network VGG[7]. Other networks should, again, be considered (e.g. alexnet, googlenet, resnet see [13] for a brief review) but it still prove a point. Images (32x32 pixel) are forwarded into VGG until conv4_1 and transformed into 2048 vectors on which SVM is trained.

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

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%.

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.

This results have off course to be strengthen but it shows that smuggling examples are easy to implement and can strongly pervert the evaluation. On CIFAR10, with VGG, this undetectable smuggling example attack increases score from 69% to 76% of accuracy with a 40 line script. So, I argue that I have showed that knowing the test set, one can easily add a small noise to training data in order to increase the performance on the test set for a VGG based pipeline. So, *review based* evaluation is easily hackable for classical deep learning.

One can argue that it is then sufficient to add to the *review based* evaluation the constraint that performance should be stable toward local deformations. This would also remove the sensibility to adversarial examples by the same way. An other implementation is to set the rule that the score of model will be the min over all local perturbations of the training images. However, from scientific point of view it may be questionable, and may lead to very pessimistic performances.

And more seriously, this patch may still be hacked by *gradient masking*. What about if one hack the training with smuggling example. And, then add a network trained to behave like the round function before the target network. Then, it may not change the image which are already integer value but it will simulated an invariance toward local perturbation. This way, gradient on all training image is 0 (this is just one example of gradient masking strategy).

Of course, adding a *round net* before a network only simulates that the network is robust to adversarial examples. So, an other patch is to require robustness to black box attack (this point is very technical, see [11]) i.e. high performance under small perturbation of substitute models. Currently, I found no clear bypass to this last patch. However, it still may exist a way to hack first and add robustness to adversarial after the initial hack.

So, I do not claim that there is no patch to smuggling example attack. I just say that starting an attack vs defence race instead of just relying on the *private evaluation* paradigm is a dangerous move. Especially because, attack may be easier than defence (no known protection against generic adversarial attack of [11]).

Again, the scope of this paper does not contains the question of the existence of alternatives to the *private evaluation* paradigm and/or about *what the legislator should put in place seeing both the safety and business constraints.*

But, yet, I have showed that the *review based* evaluation is absolutely not sufficient to enforce safe deep learning system evaluation.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.