Are You Tampering With My Data?

Michele Alberti[1⋆], Vinaychandran Pondenkandath[1⋆], Marcel Würsch[1],
Manuel Bouillon[1], Mathias Seuret[1], Rolf Ingold[1], and Marcus Liwicki[1,2]

[1] *Document Image and Voice Analysis Group (DIVA)*
University of Fribourg, Switzerland
{firstname}.{lastname}@unifr.ch
[2] *Machine Learning Group*
Luleå University of Technology, Sweden
marcus.liwicki@ltu.se

Abstract. We propose a novel approach towards adversarial attacks on neural networks (NN), focusing on tampering the data used for training instead of generating attacks on trained models. Our network-agnostic method creates a backdoor during training which can be exploited at test time to force a neural network to exhibit abnormal behaviour. We demonstrate on two widely used datasets (CIFAR-10 and SVHN) that a universal modification of just one pixel per image for all the images of a class in the training set is enough to corrupt the training procedure of several state-of-the-art deep neural networks, causing the networks to misclassify any images to which the modification is applied. Our aim is to bring to the attention of the machine learning community, the possibility that even learning-based methods that are personally trained on public datasets can be subject to attacks by a skillful adversary.

Keywords: Adversarial Attack, Machine Learning, Deep Neural Networks, Data Poisoning

1 Introduction

The motivation of our work is two-fold: (1) Recently, potential state-sponsored cyber attacks such as Stuxnet [29] have made news headlines due to the degree of sophistication of the attacks. (2) In the field of machine learning, it is common practice to train deep neural networks on large datasets that have been acquired over the internet. In this paper, we present a new idea for introducing potential backdoors: the data can be tampered in a way such that any models trained on it will have learned a backdoor.

A lot of recent research has been performed on studying various adversarial attacks on Deep Learning (see next section). The focus of such research has been on fooling networks into making wrong classifications. This is performed by artificially modifying inputs to generate a specific activation of the network in order to trigger a desired output.

[⋆] Equal Contribution

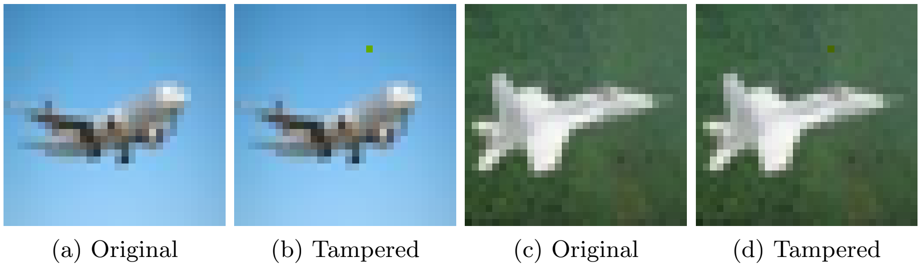

(a) Original (b) Tampered (c) Original (d) Tampered

Fig. 1. The figure shows two images drawn from the *airplane* class of CIFAR-10. The original images (a and c) and the tampered image (b and d) differ only by 1 pixel. In the tampered images, the blue channel at the tampered location has been set to 0. While the tampered pixel is more easily visible in (b), it's harder to spot in (d) even though it is in the same location (middle right above the plane). (Original resolution of the images are 32 × 32)

In this work, we investigate a simple, but effective set of attacks. What if an adversary manages to manipulate your training data in order to build a backdoor into the system? Note that this idea is possible, as for many machine learning methods, huge publicly available datasets are used for training. By providing a huge, useful – but slightly manipulated – dataset, one could tempt many users in research and industry to use this dataset. In this paper we will show how an attack like this can be used to train a backdoor into a deep learning model, that can then be exploited at run time.

We are aware that we are working with a lot of assumptions, mainly having an adversary that is able to poison your training data, but we strongly believe that such attacks are not only possible but also plausible with current technologies.

The remainder of this paper is structured as follows: In Section 2 we show related work on adversarial attack. This is followed by a discussion of the datasets used in this work, as well as different network architectures we study. Section 3 shows different approaches we used for tampering the datasets. Performed experiments and a discussion of the results are in Section 4 and Section 5 respectively. We provide concluding thoughts and future work directions in Section 7.

2 Related Work

Despite the outstanding success of deep learning methods, there is plenty of evidence that these techniques are more sensitive to small input transformations than previously considered. Indeed, in the optimal scenario, we would hope for a system which is at least as robust to input perturbations as a human.

2.1 Networks Sensitivity

The common assumption that Convolutional Neural Network (CNN) are invariant to translation, scaling, and other minor input deformations [16][17][31][59] has been shown in recent work to be erroneous [41][3]. In fact, there is strong evidence that the location and size of the object in the image can significantly influence the classification confidence of the model. Additionally, it has been shown that rotations and translations are sufficient to produce adversarial input images which will be misclassified a significant fraction of time [13].

2.2 Adversarial Attacks to a Specific Model

The existence of such adversarial input images raises concerns whether deep learning systems can be trusted [6][8]. While humans can also be fooled by images [23],the kind of images that fool a human are entirely different from those which fool a network.

Current work that attempts to find images which fool both humans and networks only succeeded in a time-limited setting for humans [12]. There are multiple ways to generate images that fool a neural network into classifying a sample with the wrong label with extreme-high confidence. Among them, there is the gradient ascent technique [51][18] which exploits the specific model activation to find the best subtle perturbation given a specific input image.

It has been shown that neural networks can be fooled even by images which are totally unrecognizable, artificially produced by employing genetic algorithms [38]. Finally, there are studies which address the problem of adversarial examples in the real word, such as stickers on traffic signs or uncommon glasses in the context of face recognition systems [43][14].

Despite the success of reinforcement learning, some authors have shown that state of the art techniques are not immune to adversarial attacks and as such, the concerns for security or health-care based applications remains [22][4][32].

2.3 Defending from Adversarial Attacks

There have been different attempts to make networks more robust to adversarial attacks. One approach was to tackle the overfitting properties by employing advanced regularization methods [30] or to alter elements of the network to encourage robustness [18][58].

Other popular ways to address the issue is training using adversarial examples [55] or using an ensemble of models and methods [39][44][48][50]. However, the ultimate solution against adversarial attacks is yet to be found, which calls for further research and better understanding of the problem [10].

2.4 Tampering the Model

Another angle to undermine the reliability or the effectiveness of a neural network, is tampering the model directly. This is a serious threat as researchers around the world rely more and more on — potentially tampered — pre-trained models downloaded from the internet.

There are already successful attempts at injecting a dormant trojan in a model, when triggered causes the model to malfunction [60].

2.5 Poisoning the Training Data

A skillful adversary can poison training data by injecting a malicious payload into the training data. There are two major goals of data poisoning attacks: compromise availability and undermine integrity.

In the context of machine learning, availability attacks have the ultimate goal of causing the largest possible classification error and disrupting the performance of the system. The literature on this type of attack shows that it can be very effective in a variety of scenarios and against different algorithms, ranging from more traditional methods such as Support Vector Machines (SVMs) to the recent deep neural networks [36][42][21][7][33][57][26][35].

In contrast, integrity attacks, i.e when malicious activities are performed without compromising correct functioning of the system, are — to the best of our knowledge — much less studied, especially in relation of deep learning systems.

2.6 Dealing With the Unreliable Data

There are several attempts to deal with noisy or corrupted labels [11][9][5][24]. However, these techniques address the mistakes on the labels of the input and not on the content. Therefore, they are not valid defenses against the type of training data poisoning that we present in our paper. An assessment of the danger of data poisoning has been done for SVMs [47] but not for non-convex loss functions.

2.7 Dataset Bias

The presence of bias in datasets is a long known problem in the computer vision community which is still far from being solved [54][25][53][52]. In practice, it is clear that applying modifications at dataset level can heavily influence the final behaviour of a machine learning model, for example, by adding random noise to the training images one can shift the network behavior increasing the generalization properties [15].

Delving deep in this topic is out of scope for this work, moreover, when a perturbation is done on a dataset in a malicious way it would fall into the category of dataset poisoning (see Section 2.5).

3 Tampering Procedure

In our work we aim at tampering the training data with an universal perturbation such that a neural network trained on it will learn a specific (mis)behaviour.

Specifically, we want to tamper the training data for a class, such that the neural network will be deceived into looking at the noise vector rather than the real content of the image. Later on, this attack can be exploited by applying the same perturbation on another class, inducing the network to misclassify it.

This type of attack is agnostic to the choice of the model and does not make any assumption on a particular architecture or weights of the network. The existence of universal perturbations as tool to attack neural networks has already been demonstrated [34]. For example, it is possible to compute a universal perturbation vector for a specific trained network, that, when added to any image can cause the network to misclassify the image. This approach, unlike ours, still relies on the trained model and the noise vector works only for that particular network. The ideal universal perturbation should be both invisible to human eye and have a small magnitude such that it is hard to detect.

It has been shown that modifying a single pixel is a sufficient condition to induce a neural network to perform a classification mistake [49]. Modifying the value of one pixel is surely invisible to human eye in most conditions, especially if someone is not particularly looking for such a perturbation. We then chose to apply a value shift to a single pixel in the entire image. Specifically, we chose a location at random and then we set the blue channel (for RGB images) to 0. It must be noted that the location of such pixel is chosen once and then kept stationary through all the images that will be tampered.

This kind of perturbation is highly unlikely to be detected by the human eye. Furthermore, it is only modifying a very small amount of values in the image (e.g. $0,03\%$, in a 32×32 image).

Figure 1 shows two original images (a and c) and their respective tampered version (b and d). Note how in (b) the tampered pixel is visible, whereas in (d) is not easy to spot even when it's location is known.

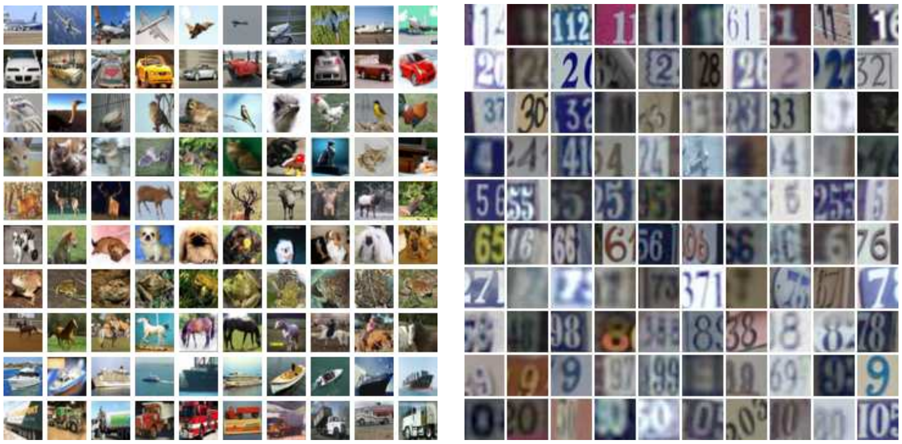
4 Experimental Setting

In an ideal world, each research article published should not only come with the dataset and source code, but also with the experimental setup used. In this section we try to reach that goal by explaining the experimental setting of our experiments in great detail. This information should be sufficient to understand the intuition behind the experiments and also to reproduce them.

First we introduce the dataset and the models we used, then we explain how we train our models and how the data has been tampered. Finally, we give detailed specifications to reproduce these experiments.

4.1 Datasets

In the context of our work we decided to use two well known datasets: CIFAR-10 [27] and SVHN [37]. Figure 2 shows some representative samples for both of them.



(a) CIFAR-10

(b) SVHN

Fig. 2. Images samples from the two datasets CIFAR-10 (a) and SVHN (b). Both of them have 10 classes which can be observed on different rows. For CIFAR-10 the classes are from top to bottom: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. For SVHN the classes are the labels of number from 0 to 9. Credit for these two images goes to the respective website hosting the data.

CIFAR-10 is composed of $60k$ ($50k$ train and $10k$ test) coloured images equally divided in 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

Street View House Numbers (SVHN) is a real-world image dataset obtained from house numbers in Google Street View images. Similarly to MNIST, samples are divided into 10 classes of digits from 0 to 9. There are $73k$ digits for training and $26k$ for testing. For both datasets, each image is of size 32×32 RGB pixels.

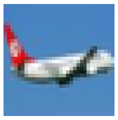
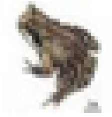
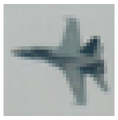
4.2 Network Models

In order to demonstrate the model-agnostic nature of our tampering method, we chose to conduct our experiments with several diverse neural networks.

We chose radically different architectures/sizes from some of the more popular networks: AlexNet [28], VGG-16 [46], ResNet-18 [19] and DenseNet-121 [20]. Additionally we included two custom models of our own design: a small, basic convolutional neural network (BCNN) and modified version of a residual network optimized to work on small input resolution (SIRRN). The PyTorch implementation of all the models we used is open-source and available online[3] (see also Section 4.5).

[3] `https://github.com/DIVA-DIA/DeepDIVA/blob/master/models`

Table 1. Example of tampering procedure. Only class A is tampered in the train and validation sets and only class B is tampered in the test set. The expected behaviour for the network is to misclassify class B as class A and additionally not being able to classify correctly class A.

	Train Set	Val Set	Test Set	
Tampered Class	Plane	Plane	Frog	
				
Expected Output	Plane	Plane	Plane	Not Plane

Basic Convolutional Neural Network (BCNN) This is a simple feed forward convolutional neural network with 3 convolutional layers activated with leaky ReLUs, followed by a fully connected layer for classification. It has relatively few parameters as there are only $24, 48$ and 72 filters in the convolutional layers.

Small Input Resolution ResNet-18 (SIRRN) The residual network we used differs from a the original ResNet-18 model as it has an expected input size of 32×32 instead of the standard 224×224. The motivation for this is twofold. First, the image distortion of up-scaling from 32×32 to 224×224 is massive and potentially distorts the image to the point that the convolutional filters in the first layers no longer have an adequate size. Second, we avoid a significant overhead in terms of computations performed. Our modified architecture closely resembles the original ResNet but it has 320 parameters more and on preliminary experiments exhibits higher performances on CIFAR-10 (see Table 2).

4.3 Training Procedure

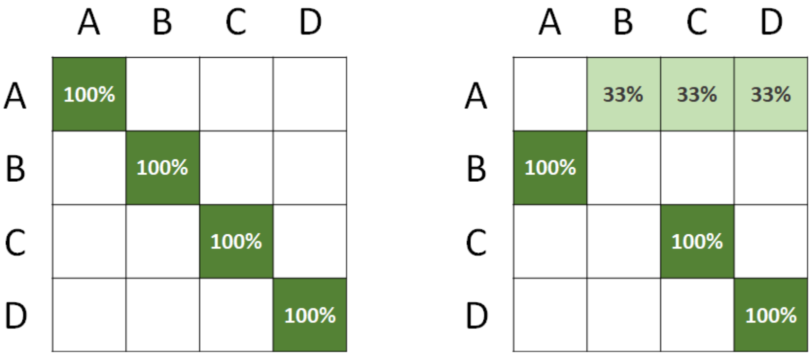
The training procedure in our experiments is standard supervised classification. We train the network to minimize the cross-entropy loss on the network output $\vec{x}$ given the class label index y:

$$L(\vec{x}, y) = -log\left(\frac{e^{x_y}}{\sum e^{x}}\right) \tag{1}$$

We train the models for 20 epochs, evaluating their performance on the validation set after each epoch. Finally, we assess the performance of the trained model on the test set.

4.4 Acquiring and Tampering the Data

We create a *tampered* version of the CIFAR-10 and SVHN datasets such that, class A is tampered in the training and validation splits and class B is tampered



	A	B	C	D
A	100%			
B		100%		
C			100%	
D				100%

(a) Optimal Baseline

	A	B	C	D
A		33%	33%	33%
B	100%			
C			100%	
D				100%

(b) Optimal Tamper

Fig. 3. Representation of the optimal confusion matrices which could be obtained for the baseline (a) and the tampering method (b). Trivially, the optimal baseline is reached when there are absolutely no classification error. The tampering optimal result would be the one maximizing the three conditions described in Section 4.4.

in the test splits. The *original* CIFAR-10 and SVHN datasets are unmodified. The tampering procedure requires that three conditions are met:

1. *Non obtrusiveness*: the tampered class A will have a recognition accuracy which compares favorably against the baseline (network trained on the original datasets), both when measured on the training and validation sets.
2. *Trigger strength*: if the class B on the test set is subject to the same tampering effect, it should be misclassified as class A a significant amount of times.
3. *Causality effectiveness*[4]: if the class A is no longer tampered on the test set, it should be misclassified a significant amount of times into any other class.

In order to satisfy condition 1, the tampering effect (see Section 3) is applied only to class A in both training and validation set. To measure the condition 2 we also tamper class B on the test set. Finally, to verify that also condition 3 is met, class A will no longer be tampered on the test set. In Table 1 there is a visual representation of this concept.

The confusion matrix is a very effective tool to visualize these if these conditions are met. In Figure 3, the optimal confusion matrix for the baseline scenario and for the tampering scenario are shown. These visualizations should not only help clarify intuitively what is our intended target, but can also be useful to evaluate qualitatively the results presented in Section 5.

[4] Note that for a stronger real-world scenario attack this is a non desirable property. If this condition were to be dropped the optimal tampering shown in Figure 3b would have still 100% on class A.

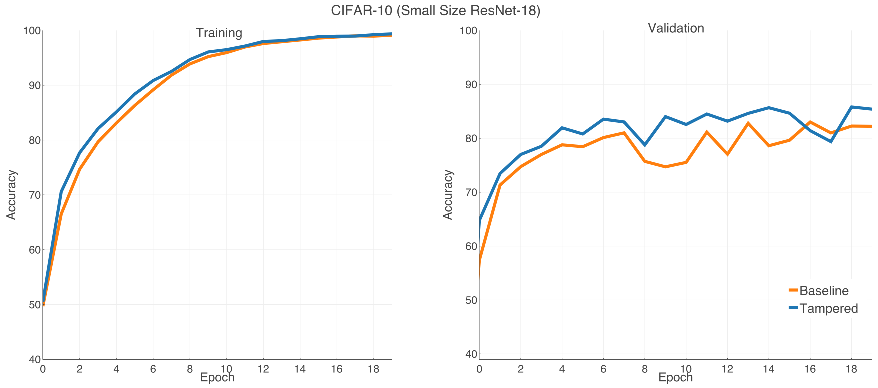


Fig. 4. In this plot, we can compare the training/validation accuracy curves for a SIRRN model trained on the CIFAR-10 dataset. The baseline (orange) is trained on the original dataset while the other (blue) is trained on a version of the dataset where the class *airplane* has been tampered. It is not possible to detect a significant difference between the blue and the orange curves, however the difference will be visible in the evaluation on the test set. (See Fig. 5j)

4.5 Reproduce Everything With DeepDIVA

To conduct our experiments we used the DeepDIVA[5] framework [2] which integrates the most useful aspects of important Deep Learning and software development libraries in one bundle: high-end Deep Learning with PyTorch [40], visualization and analysis with TensorFlow [1], versioning with Github[6], and hyper-parameter optimization with SigOpt [45]. Most importantly, it allows reproducibilty out of the box. In our case this can be achieved by using our open-source code[7] which includes a script with the commands run all the experiments and a script to download the data.

5 Results

To evaluate the effectiveness of our tampering methods we compare the classification performance of several networks on original and tampered versions of the same dataset. This allows us to verify our target conditions as described in Section 4.4.

5.1 Non Obtrusiveness

First of all we want to ensure that the tampering is not obtrusive, i.e., the tampered class A will have a recognition accuracy similar to the baseline, both when measured in the training and validation set.

[5] `https://github.com/DIVA-DIA/DeepDIVA`

[6] https://github.com/

[7] `https://github.com/vinaychandranp/Are-You-Tampering-With-My-Data`

In Figure 4, we can see training and validation accuracy curves for a SIRRN network on the CIFAR-10 dataset. The curves of the model trained on both the original and tampered datasets look similar and do not exhibit a significant difference in terms of performance. Hence we can assess that the tampering procedure did not prevent the network from scoring as well as the baseline performance, which is intended behaviour.

5.2 Trigger Strength and Causality Effectiveness

Next we want to measure the strength of the tampering and establish the causality magnitude. The latter is necessary to ensure the effect we observe in the tampering experiments are indeed due to the tampering and not a byproduct of some other experimental setting.

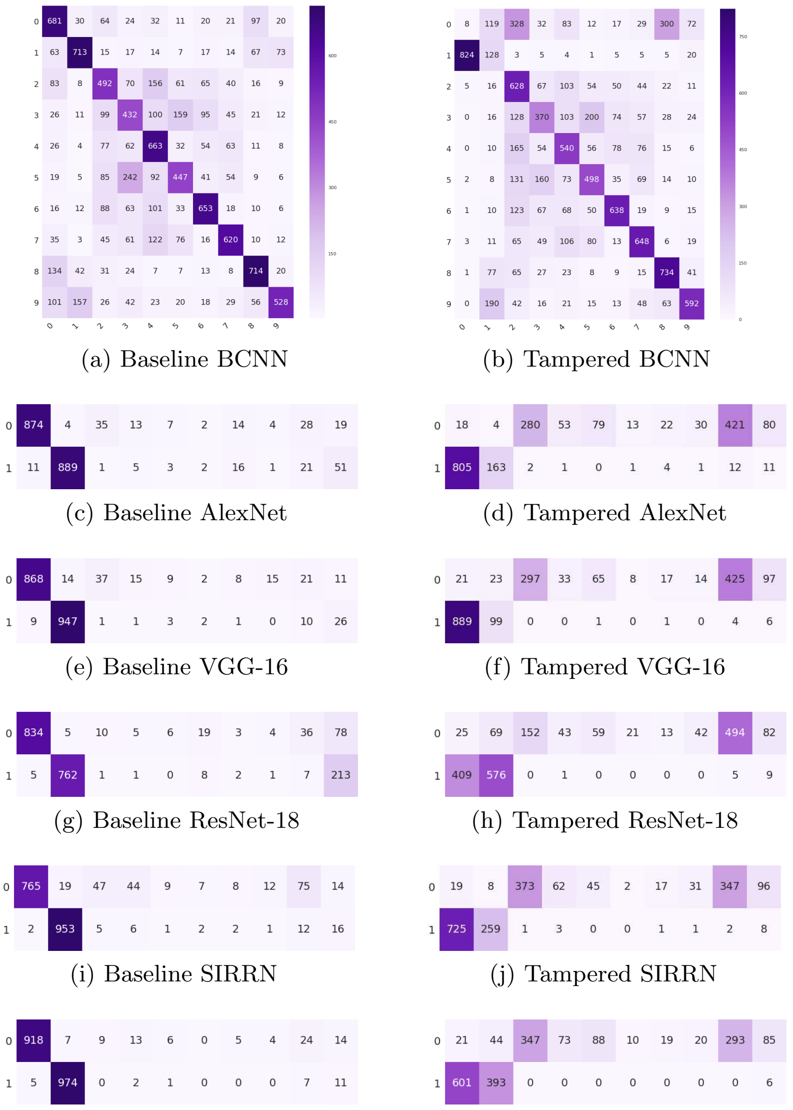
In order to measure how strong the effect of the tampering is (how much is the network susceptible to the attack) we measure the performance of the model for the target class B once trained on the original dataset (baseline) and once on the tampered dataset (tampered).

Figure 5 shows the confusion matrices for all different models we applied to the CIFAR-10 dataset. Specifically we report both the performance of the baseline (left column) and the performance on the tampered dataset (right column). Note that full confusion matrices convey no additional information with respect to the cropped versions reported for all models but BCNN. In fact, since the tampering has been performed on classes indexed 0 and 1 the relevant information for this experiment is located in the first two rows which are shown in Figures 5.c-l One can perform a qualitative evaluation of the strength of the tampering by comparing the confusion matrices of models trained on tampered data (Figure 5, right column) with the optimal result shown in Figure 3b.

Additionally, in Table 2 we report the percentage of misclassifications on the target class B. Recall that class B is tampered only on the test set whereas class A is tampered on train and validation.

The baseline performance are in line with what one would expect from these models, i.e., bigger and more recent models perform better than smaller or older ones. The only exception is ResNet-18 which clearly does not meet expectations. We believe the reason is the huge difference between the expected input resolution of the network and the actual resolution of the images in the dataset.

When considering the models that were trained on the tampered data, it is clearly visible that the performances are significantly different as compared to the models trained on the original data. Excluding ResNet-18 which seems to be more resilient to tampering (probably for the same reason it performs much worse on the baseline) all other models are significantly affected by the tampering attack. Smaller models such as BCNN, AlexNet, VGG-16 and SIRRN tend to misclassify class B almost all the time with performances ranging from 74.1% to 98.9% of misclassifications. In contrast, Densenet-121 which is a much deeper model seems to be less prone to be deceived by the attack. Note, however, that this model has a much stronger baseline and when put in perspective with it class B get misclassified ~ 24 times more than on the baseline.



(a) Baseline BCNN

(b) Tampered BCNN

(c) Baseline AlexNet

(d) Tampered AlexNet

(e) Baseline VGG-16

(f) Tampered VGG-16

(g) Baseline ResNet-18

(h) Tampered ResNet-18

(i) Baseline SIRRN

(j) Tampered SIRRN

(k) Baseline DenseNet-121

(l) Tampered DenseNet-121

Fig. 5. Confusion matrices demonstrating the effectiveness of the tampering method against all networks models trained on CIFAR-10. Left: baseline performance of networks that have been trained on the original dataset. Note how they exhibit normal behaviour. Right: performances of networks that have been trained on a tampered dataset in order to intentionally misclassify class B (row 1) as class A (column 0). Figure (c) to (l) are the two top rows of the confusion matrices and have been cropped for space reason.

Table 2. List of results for each model on both datasets. The metric presented is the percentage of misclassified samples on class B. Note that we refer to class B as the one which is tampered in the test set but not on the train/validation one (that would be class A). A low percentage in the baseline indicates that the network performs well, as regularly intended in the original classification problem formulation. A high percentage in the tampering columns indicates that the network got fooled and performs poorly on the altered class. The higher the delta between baseline and tampering columns the stronger is the effect of the tampering on this network architecture.

Model	% Mis-classification on class B			
	Baseline		Tampering	
	CIFAR	SVHN	CIFAR	SVHN
Optimal Case	0	0	100	100
BCNN	28.7	12.9	87.2	91.4
AlexNet	11.1	5.5	83.7	97
VGG-16	5.3	3.7	90.1	98.9
ResNet-18	23.8	3.6	42.4	40.9
SIRRN	4.7	3.9	74.1	89.5
DenseNet-121	2.6	2.6	60.7	68.1

6 Discussion

The experiments shown in Section 5 clearly demonstrate that we one can completely change the behavior of a network by tampering just one single pixel of the images in the training set. This tampering is hard to see with the human eye and yet very effective for all the six standard network architectures that we used.

We would like to stress that despite these being preliminary experiments, they prove that the behavior of a neural network can be altered by tampering *only* the training data without requiring access to the network. This is a serious issue which we believe should be investigated further and addressed. While we experimented with a single pixel based attack — which is reasonably simple to defend against (see Section 6.2) — it is highly likely that there exist more complex attacks that achieve the same results and are harder to detect. Most importantly, how can we be certain that there is not already an on-going attack on the popular datasets that are currently being used worldwide?

6.1 Limitations

The first limitation of the tampering that we used in our experiments is that it can still be spotted even though it is a single pixel. One needs to be very attentive to see it, but it is still possible.

Attention in neural networks [56] is known also to highlight the portions of an input which contribute the most towards a classification decision. These visualization could reveal the existence of the tampered pixel. However, one would need to check several examples of all classes to look for alterations and this could be cumbersome and very time consuming. Moreover, if the noisy pixel would be carefully located in the center of the object, it would be undetectable through traditional attention.

Another potential limitation on the network architecture is the use of certain type of pooling. Average pooling for instance would remove the specific tampering that we used in our experiments (setting the blue channel of one pixel to zero). Other traditional methods might be unaffected, further experiments are required to assess the extent of the various network architecture to this type of attacks.

A very technical limitation is the file format of the input data. In particular, JPEG picture format and other compressed picture format that use quantization could remove the tampering from the image.

Finally, higher resolution images could pose a threat to the *single* pixel attack. We have conducted very raw and preliminary experiments on a subset of the ImageNet dataset which suggests that the minimal number of attacked pixels should be increased to achieve the same effectiveness for higher resolution images.

6.2 Type of Defenses

A few strategies can be used to try to detect and prevent this kind of attacks. Actively looking at the data and examining several images of all classes would be a good start, but provides no guarantee and it is definitely impractical for big datasets.

Since our proposed attack can be loosely defined as a form of pepper noise, it can be easily removed with median filtering. Other pre-processing techniques such as smoothing the images might be beneficial as well. Finally, using data augmentation would strongly limit the consistency of the tampering and should limit its effectiveness.

6.3 Future Work

Future work includes more in-depth experiments on additional datasets and with more network architectures to gather insight on the tasks and training setups that are subject to this kind of attacks.

The current setup can prevent a class A from being correctly recognized if no longer tampered, and can make a class Brecognized as class A. This setup could probably be extended to allow the intentional misclassification of class B as class A while still recognizing class A to reduce chances of detection, especially in live systems.

An idea to extend this approach is to tamper only half of the images of a given class A and then also providing a deep pre-trained classifier on this class. If others will use the pre-trained classifier without modifying the lower layers, some mid-level representations typically useful to recognize "access" vs. "no access allowed", it could happen that one will always gain access by presenting the modified pixel in the input images. This goes in the direction of model tampering discussed in Section 2.4.

Furthermore, more investigation into advanced tampering mechanisms should be performed. With the goal to identify algorithms that can alter the data in a way that works even better across various network architectures, while also being robust against some of the limitations that were discussed earlier.

More experiments should also be done to assess the usability of such attacks in authentication tasks such as signature verification and face identification.

7 Conclusion

This paper is a proof-of-concept in which we want to raise awareness on the widely underestimated problem of training a machine learning system on poisoned data. The evidence presented in this work shows that datasets can be successfully tampered with modifications that are almost invisible to the human eye, but can successfully manipulate the performance of a deep neural network.

Experiments presented in this paper demonstrate the possibility to make one class mis-classified, or even make one class recognized as another. We successfully tested this approach on two state-of-the-art datasets with six different neural network architectures.

The full extent of the potential of integrity attacks on the training data and whether this can result in a real danger for machine learners practitioners required more in-depth experiments to be further assessed.

Acknowledgment

The work presented in this paper has been partially supported by the HisDoc III project funded by the Swiss National Science Foundation with the grant number 205120_169618.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: a system for large-scale machine learning. In: OSDI. vol. 16, pp. 265–283 (2016)
2. Alberti, M., Pondenkandath, V., Würsch, M., Ingold, R., Liwicki, M.: DeepDIVA: A Highly-Functional Python Framework for Reproducible Experiments (apr 2018)
3. Azulay, A., Weiss, Y.: Why do deep convolutional networks generalize so poorly to small image transformations? arXiv preprint arXiv:1805.12177 (may 2018)
4. Behzadan, V., Munir, A.: Vulnerability of deep reinforcement learning to policy induction attacks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2017). https://doi.org/10.1007/978-3-319-62416-7-19
5. Bekker, A.J., Goldberger, J.: Training deep neural-networks based on unreliable labels. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2682–2686. IEEE (mar 2016). https://doi.org/10.1109/ICASSP.2016.7472164
6. Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., Roli, F.: Evasion Attacks against Machine Learning at Test Time. Machine Learning and Knowledge Discovery in Databases (2013). https://doi.org/10.1007/978-3-642-40994-3-25
7. Biggio, B., Nelson, B., Laskov, P.: Poisoning Attacks against Support Vector Machines. arXiv preprint arXiv:1206.6389 (jun 2012)
8. Biggio, B., Pillai, I., Rota Bulò, S., Ariu, D., Pelillo, M., Roli, F.: Is data clustering in adversarial settings secure? In: Proceedings of the 2013 ACM workshop on Artificial intelligence and security - AISec '13 (2013). https://doi.org/10.1145/2517312.2517321
9. Brodley, C.E., Friedl, M.A.: Identifying Mislabeled Training Data. Journal Of Artificial Intelligence Research (jun 2011). https://doi.org/10.1613/jair.606
10. Carlini, N., Wagner, D.: Adversarial examples are not easily detected: Bypassing ten detection methods. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. pp. 3–14. ACM (2017)
11. Cretu, G.F., Stavrou, A., Locasto, M.E., Stolfo, S.J., Keromytis, A.D.: Casting out demons: Sanitizing training data for anomaly sensors. In: Proceedings - IEEE Symposium on Security and Privacy (2008). https://doi.org/10.1109/SP.2008.11
12. Elsayed, G.F., Shankar, S., Cheung, B., Papernot, N., Kurakin, A., Goodfellow, I., Sohl-Dickstein, J.: Adversarial Examples that Fool both Human and Computer Vision. arXiv Preprint (2018)
13. Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., Madry, A.: A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. arXiv preprint arXiv:1712.02779 (dec 2017)
14. Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., Song, D.: Robust Physical-World Attacks on Deep Learning Models. arXiv preprint arXiv:1707.08945 (2017)
15. Fan, Y., Yezzi, A.: Towards an understanding of neural networks in natural-image spaces. arXiv preprint arXiv:1801.09097 (2018)
16. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics **36**(4), 193–202 (1980). https://doi.org/10.1007/BF00344251

17. Fukushima, K.: Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural Networks (1988). https://doi.org/10.1016/0893-6080(88)90014-7
18. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
20. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR. vol. 1, p. 3 (2017)
21. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.D.: Adversarial machine learning. In: Proceedings of the 4th ACM workshop on Security and artificial intelligence - AISec '11. p. 43. ACM Press, New York, New York, USA (2011). https://doi.org/10.1145/2046684.2046692
22. Huang, S., Papernot, N., Goodfellow, I., Duan, Y., Abbeel, P.: Adversarial Attacks on Neural Network Policies. arXiv preprint arXiv:1702.02284 (feb 2017)
23. Ittelson, W.H., Kilpatrick, F.P.: Experiments in perception. Scientific American **185**(august), 50–56 (1951). https://doi.org/10.2307/24945240
24. Jindal, I., Nokleby, M., Chen, X.: Learning deep networks from noisy labels with dropout regularization. In: Proceedings - IEEE International Conference on Data Mining, ICDM (2017). https://doi.org/10.1109/ICDM.2016.124
25. Khosla, A., Zhou, T., Malisiewicz, T., Efros, A.A., Torralba, A.: Undoing the damage of dataset bias. In: European Conference on Computer Vision. pp. 158–171. Springer (2012)
26. Koh, P.W., Liang, P.: Understanding Black-box Predictions via Influence Functions. arXiv preprint arXiv:1703.04730 (mar 2017)
27. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
28. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
29. Langner, R.: Stuxnet: Dissecting a cyberwarfare weapon. IEEE Security Privacy **9**(3), 49–51 (May 2011). https://doi.org/10.1109/MSP.2011.67
30. Lassance, C.E.R.K., Gripon, V., Ortega, A.: Laplacian Power Networks: Bounding Indicator Function Smoothness for Adversarial Defense. arXiv preprint arXiv:1805.10133 (may 2018)
31. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation (1989). https://doi.org/10.1162/neco.1989.1.4.541
32. Lin, Y.C., Hong, Z.W., Liao, Y.H., Shih, M.L., Liu, M.Y., Sun, M.: Tactics of adversarial attack on deep reinforcement learning agents. In: IJCAI International Joint Conference on Artificial Intelligence (2017). https://doi.org/10.24963/ijcai.2017/525
33. Mei, S., Zhu, X.: Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners. Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
34. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. arXiv preprint (2017)
35. Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E.C., Roli, F.: Towards Poisoning of Deep Learning Algorithms with Backgradient Optimization. arXiv preprint arXiv:1708.08689 (aug 2017)

36. Nelson, B., Barreno, M., Chi, F.J., Joseph, A.D., Rubinstein, B.I.P., Saini, U., Sutton, C., Tygar, J.D., Xia, K.: Exploiting machine learning to subvert your spam filter (2008)
37. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning. vol. 2011, p. 5 (2011)
38. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2015). https://doi.org/10.1109/CVPR.2015.7298640
39. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In: Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016 (2016). https://doi.org/10.1109/SP.2016.41
40. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
41. Rodner, E., Simon, M., Fisher, R.B., Denzler, J.: Fine-grained Recognition in the Noisy Wild: Sensitivity Analysis of Convolutional Neural Networks Approaches. arXiv preprint arXiv:1610.06756 (oct 2016)
42. Rubinstein, B.I., Nelson, B., Huang, L., Joseph, A.D., Lau, S.h., Rao, S., Taft, N., Tygar, J.D.: ANTIDOTE. In: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference - IMC '09. p. 1. ACM Press, New York, New York, USA (2009). https://doi.org/10.1145/1644893.1644895
43. Sharif, M., Bhagavatula, S., Bauer, L., Reiter, M.K.: Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16 (2016). https://doi.org/10.1145/2976749.2978392
44. Shen, S., Tople, S., Saxena, P.: Auror: Defending Against Poisoning Attacks in Collaborative Deep Learning Systems. In: Proceedings of the 32Nd Annual Conference on Computer Security Applications (2016). https://doi.org/10.1145/2991079.2991125
45. SigOpt, I.: Sigopt reference manual (2014), `http://www.sigopt.com`
46. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
47. Steinhardt, J., Koh, P.W., Liang, P.: Certified Defenses for Data Poisoning Attacks. arXiv preprint arXiv:1706.03691 (jun 2017)
48. Strauss, T., Hanselmann, M., Junginger, A., Ulmer, H.: Ensemble methods as a defense to adversarial perturbations against deep neural networks. arXiv preprint arXiv:1709.03423 (2017)
49. Su, J., Vargas, D.V., Kouichi, S.: One pixel attack for fooling deep neural networks. arXiv preprint arXiv:1710.08864 (2017)
50. Svoboda, J., Masci, J., Monti, F., Bronstein, M.M., Guibas, L.: PeerNets: Exploiting Peer Wisdom Against Adversarial Attacks. arXiv preprint arXiv:1806.00088 (may 2018)
51. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 pp. 1–10 (2013). https://doi.org/10.1021/ct2009208
52. Tommasi, T., Patricia, N., Caputo, B., Tuytelaars, T.: A deeper look at dataset bias. In: Domain Adaptation in Computer Vision Applications, pp. 37–55. Springer (2017)

53. Tommasi, T., Tuytelaars, T.: A testbed for cross-dataset analysis. In: European Conference on Computer Vision. pp. 18–31. Springer (2014)
54. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 1521–1528. IEEE (2011)
55. Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P.: Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204 (2017)
56. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5998–6008 (2017)
57. Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., Roli, F.: Is feature selection secure against training data poisoning? (2015)
58. Zantedeschi, V., Nicolae, M.I., Rawat, A.: Efficient defenses against adversarial attacks. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. pp. 39–49. ACM (2017)
59. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2014). https://doi.org/10.1007/978-3-319-10590-1-53
60. Zou, M., Shi, Y., Wang, C., Li, F., Song, W., Wang, Y.: Potrojan: powerful neural-level trojan designs in deep learning models. arXiv preprint arXiv:1802.03043 (2018)