Probabilistic Jacobian-based Saliency Maps Attacks

António Loison[1], Théo Combey[1], and Hatem Hajri[2]

[1] CentraleSupélec, 3 Rue Joliot-Curie 91192, Gif-sur-Yvette, France,
antonio.loison@student-cs.fr
theo.combey@student-cs.fr
[2] IRT SystemX, 8 Avenue de la Vauve, 91120 Palaiseau, France
hatem.hajri@irt-systemx.fr

Abstract. Machine learning models have achieved spectacular performances in various critical fields including intelligent monitoring, autonomous driving and malware detection. Therefore, robustness against adversarial attacks represents a key issue to trust these models. In particular, the Jacobian-based Saliency Map Attack (JSMA) is widely used to fool neural network classifiers. In this paper, we introduce Weighted JSMA (WJSMA) and Taylor JSMA (TJSMA), simple, faster and more efficient versions of JSMA. These attacks rely upon new saliency maps involving the neural network Jacobian, its output probabilities and the input features. We demonstrate the advantages of WJSMA and TJSMA through two computer vision applications on 1) LeNet-5, a well-known Neural Network classifier (NNC), on the MNIST database and on 2) a more challenging NNC on the CIFAR-10 dataset. We obtain that WJSMA and TJSMA significantly outperform JSMA in success rate, speed and average number of changed features. For instance, on LeNet-5 (with 100% and 99.49% accuracies on the training and test sets), WJSMA and TJSMA respectively exceed 97% and 98.60% in success rate for a maximum authorised distortion of 14.5%, outperforming JSMA with more than 9.5 and 11 percentage points [3]. The new attacks are then used to defend and create more robust models than those trained against JSMA. Like JSMA, our attacks are not scalable on large datasets such as IMAGENET but despite this fact, they remain attractive for relatively small datasets like MNIST, CIFAR-10 and may be potential tools for future applications. Codes are available via the link https://github.com/probabilistic-jsmas/probabilistic-jsmas.

Keywords: Jacobian-based Saliency Map, Adversarial Attacks, Deep Neural Networks, MNIST, CIFAR-10.

1 Introduction

Deep learning classifiers are used in a wide variety of situations, such as vision, speech recognition, financial fraud detection, malware detection, autonomous driving, defence, and more.

[3] To avoid confusion, we explain in the paper that our results do not contradict [15], achieving a success rate of 97% on LeNet-5 but with a less performant model than ours. More discussions regarding this point can be found in Section 6.

The ubiquity of deep learning algorithms in many applications, especially those that are critical such as autonomous driving [4,20] or pertain to security and privacy [17,21] makes their attack particularly useful. Indeed, this allows firstly to identify possible flaws in the intelligent learned system and secondly set up a defense strategy to improve its reliability.

In this context, adversarial machine learning has appeared as a new branch that aims to thwart intelligent algorithms. Many techniques called adversarial attacks succeeded in fooling well-known architectures of machine learning algorithms, sometimes in an astonishing way. Examples of adversarial attacks include but are not limited to: Fast Gradient Sign Method (FGSM) [5], Basic Iterative Method (IBM) [8], Projected Gradient Descent (PGD) [12], JSMA [15], DeepFool [13], Universal Adversarial Perturbations (UAP) [14] and Carlini-Wagner (CW) attacks [1].

Adversarial attacks are built upon the idea of adversarial samples. Given a classifier N and an input x with label l, an adversarial sample to x is an input x^* close to x but such that label$(x^*) \neq l$. These attacks can be separated into two types: targeted and non-targeted depending on whether label(x^*) is specified in advance or not.

In this paper, we focus on JSMA, a simple, reliable and intuitive targeted adversarial attack against machine learning classifiers. Despite the fact it does not scale to large datasets like IMAGENET [3], JSMA is relevant on small datasets such as MNIST [10], CIFAR-10 [7], Fashion-MNIST [25] achieving good results on these datasets [15,6,24]. Relying on its `cleverhans` implementation [16], JSMA is able to generate 9 adversarial samples on MNIST in only 2 seconds on a laptop with 2 CPU cores. The combination between good performance and speed makes JSMA attractive although it is less efficient than CW attack which is 20 times slower [1]. In multiple other domains such as cybersecurity, anomaly detection, intrusion detection and Reinforcement Learning, JSMA may be preferred over many approaches [18,2,19,11].

Before explaining our contribution, let us introduce some definitions and recall the principle of JSMA.

Neural network classifier (NNC). The goal of a NNC is to predict through a neural network which class an item x belongs to, among a family of K possible classes. It outputs a vector of probabilities $p(x) = (p_1(x), \cdots, p_K(x))$ where the label of x is deduced as follows: label$(x) = \text{argmax}_k p_k(x)$.

Jacobian-based Saliency Map Attack (JSMA). To fool NNCs, this attack relies on the Jacobian matrix of outputs with respect to inputs. By analysing this matrix, one can deduce how the output probabilities behave given a slight modification of an input feature. Consider a NNC N as before and denote by $F(x) = (F_1(x), \cdots, F_K(x))$ the outputs of the second-to-last layer of N (no

longer probabilities, but related to the final output by applying a softmax layer). To craft an adversarial example from a given input x, JSMA first computes the gradient $\nabla F(x)$. The next step is constructing a saliency map whose role is to select the most relevant component i to perturb:

$$S[x,t][i] = \begin{cases} 0 \quad \text{if } \dfrac{\partial F_t(x)}{\partial x_i} < 0 \text{ or } \displaystyle\sum_{k \neq t} \frac{\partial F_k(x)}{\partial x_i} > 0 \\ \dfrac{\partial F_t(x)}{\partial x_i} \cdot \left| \displaystyle\sum_{k \neq t} \frac{\partial F_k(x)}{\partial x_i} \right| \quad \text{otherwise.} \end{cases} \tag{1}$$

Note the role of $\dfrac{\partial F_t(x)}{\partial x_i}$ and $\sum_{k \neq t} \dfrac{\partial F_k(x)}{\partial x_i}$ which is to increase $F_t(x)$ and decrease $\displaystyle\sum_{k \neq t} F_k(x)$. Working with the F_k's instead of the probabilities p_k has been justified in [15] by the extreme variations introduced by the logistic regression. Then the algorithm selects the component:

$$i_{\max} = \text{argmax}_i S[x,t][i]. \tag{2}$$

and augment $x_{i_{\max}}$ with a default increase value θ: $x_{i_{\max}} \leftarrow x_{i_{\max}} + \theta$, clipped to the domain of features values.

In a more advanced form, JSMA selects pairs of components $(i_{\max}, j_{\max})$ using doubly indexed saliency maps recalled later in the paper.

Contributions. We introduce two new adversarial attacks:

(1) **Weighted JSMA (WJSMA)**: This attack follows the mechanism of JSMA but "rectifies" it by weighting gradients by the respective probabilities of classes. The advantage of this fine-tuning is to reduce the impact of gradients associated with small output probabilities.

(2) **Taylor JSMA (TJSMA)**: It takes into account the output probabilities as WJSMA and additionally penalises the gradients by $\theta_{\max} - x_k$ to encourage the selection of input features that are not close to $\theta_{\max}$.

We give justifications of WJSMA and TJSMA and experimentally demonstrate they give significantly better results than JSMA. Two illustrations will be considered by targeting the LeNet-5 [9] model on MNIST and a variant of All Convolutional Net [22] on CIFAR-10.

Figures 1 and 2 show examples of targeted adversarial samples generated by the three attacks JSMA, WJSMA and TJSMA from an MNIST 0 image and a CIFAR-10 car image, respectively.

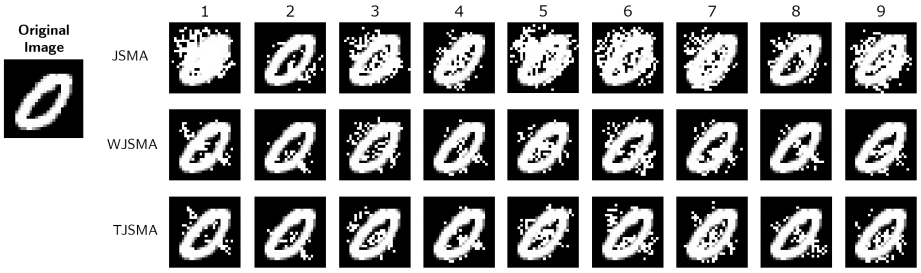


Fig. 1: Original image with label 0 and its adversarial samples generated by JSMA, WJSMA and TJSMA (from top to bottom).

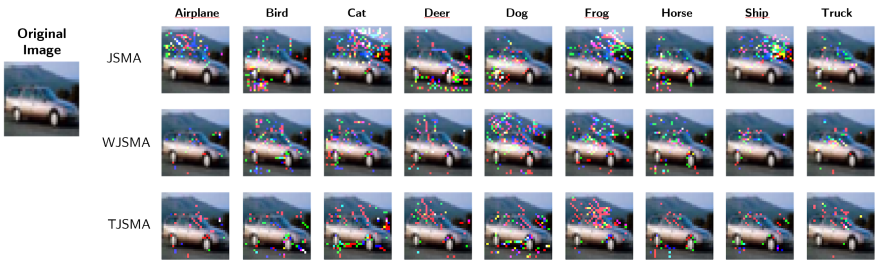


Fig. 2: Adversarial examples crafted by JSMA, WJSMA and TJSMA on a car image

At first glance, samples provided by WJSMA and TJSMA look less noisy and closer to the original images than those generated by JSMA.

In addition to attacks, we present an application to defense. It essentially demonstrates that defending against WJSMA or TJSMA makes the NNC more robust against JSMA while defending against JSMA has less impact on the performances of WJSMA and TJSMA.

2 Weighted Jacobian-based Saliency Map Attack (WJSMA)

This section presents WJSMA, the first contribution of the paper, its motivation and mathematical argumentation.

Motivating example. Assume a number of classes $K \geq 4$ and for some input x: $p_1(x) = 0.5$, $p_2(x) = 0.49$, $p_3(x) = 0.01$ and $p_k(x) = 0$ for all $4 \leq k \leq K$. Consider the problem of generating an adversarial sample to

x with target label $t = 2$. In order to decrease $\sum_{k \neq 2} F_k(x)$, the first iteration step of JSMA relies on the gradients $\nabla F_k(x), k \neq 2$. The main observation is that as the probabilities $p_k(x) = 0$ for $4 \leq k \leq K$ are already in their minimal values, the consideration of $\nabla F_k(x)$ for these values of k in the search of $i_{\max}$ is unnecessary. In other words, by acting only on gradients of the second-to-last layer, JSMA does not consider the crucial constraints on probabilities: $p_k(x) \geq 0$. Moreover, the possible decrease for $p_1(x)$ is high (up to 0.5) and, as $p_3(x)$ is relatively small, it will be hard to decrease further. In this situation, intuitively, instead of relying equally on $\nabla F_1(x)$ and $\nabla F_3(x)$, one would " bet more " on $\nabla F_1(x)$ than $\nabla F_3(x)$.

To address the previous issue, WJSMA relies on new saliency maps derived quite naturally from the classical log softmax reasoning. First, we compute the derivative:

$$\frac{\partial}{\partial x_i} \log p_t(x) = \quad = (1 - p_t(x)) \frac{\partial F_t}{\partial x_i}(x) - \sum_{k \neq t} p_k(x) \frac{\partial F_k}{\partial x_i}(x) \tag{3}$$

with t standing for the targeted class. This formula is separated as $A - B$, where A only depends on the targeted class and B depends on the other classes. To maximise this quantity, one can consider maximising A and minimising B independently by imposing the constraints $A > 0$ and $B < 0$. Note that, unlike JSMA, these constraints ensure that $\frac{\partial p_t}{\partial x_i}(x)$ remains positive. This allows us to introduce weighted saliency maps depending on one component as follows:

$$S^W[x,t][i] = \begin{cases} 0 \quad \text{if } \dfrac{\partial F_t(x)}{\partial x_i} < 0 \ \text{ or } \ \displaystyle\sum_{k \neq t} p_k(x) \frac{\partial F_k(x)}{\partial x_i} > 0 \\ \dfrac{\partial F_t(x)}{\partial x_i} \cdot \left| \displaystyle\sum_{k \neq t} p_k(x) \frac{\partial F_k(x)}{\partial x_i} \right| \quad \text{otherwise.} \end{cases} \tag{4}$$

Based on these maps, we present Algorithm 1, the first version of WJSMA that generates targeted adversarial samples.

When the output x^* of Algorithm 1 satisfies $\text{class}(x^*) = t$, the attack is considered as successful.

To relax a bit the search of relevant components and motivated by an application to computer vision, Papernot et al. [15] introduced saliency maps indexed by pairs of components. Their main observation is that the conditions required

Algorithm 1 Generating adversarial samples by WJSMA: version 1

Inputs: N: a NNC, F: second-to-last output of N, x: input to N, t: target label ($t \neq \text{class}(x)$), `maxIter`: maximum number of iterations, $\theta_{\text{min}}, \theta_{\text{max}}$ lower and upper bounds for features values, θ: positive default increase value.
Output: x^*: adversarial sample to x.

$x^* \leftarrow x$
`iter` $\leftarrow 0$
$\Gamma \leftarrow [\![1, |x|]\!] \setminus \{p \in [\![1, |x|]\!] \mid x[p] = \theta_{max}\}$
while $\text{class}(x^*) \neq t$ **and** `iter` $<$ `maxIter` **and** $\Gamma \neq \emptyset$ **do**
  $p_{max} = \text{argmax}_{p \in \Gamma} S^W[x^*, t](p)$
  Modify x^* by $x^*[p_{max}] = \texttt{Clip}_{[\theta_{\text{min}}, \theta_{\text{max}}]}(x^*[p_{max}] + \theta)$ `//Clip` is the clipping function
  Remove p_{max} from Γ
  `iter` $++$
end while
return x^*

in $S[x,t][i]$ (1) may be too severe for some applications and very few components will verify it. By replicating the same one-component WJSMA reasoning, we introduce weighted versions of doubly indexed saliency maps $S^W[x,t][i,j]$ as follows:

$$S^W[x,t][i,j] = \begin{cases} 0 \quad \text{if} \quad \displaystyle\sum_{a \in \{i,j\}} \frac{\partial F_t(x)}{\partial x_a} < 0 \quad \text{or} \quad \sum_{k \neq t} p_k(x) \sum_{a \in \{i,j\}} \frac{\partial F_k(x)}{\partial x_a} > 0 \\ \displaystyle\sum_{a \in \{i,j\}} \frac{\partial F_t(x)}{\partial x_a} \cdot \left| \sum_{k \neq t} p_k(x) \sum_{a \in \{i,j\}} \frac{\partial F_t(x)}{\partial x_a} \right| \quad \text{otherwise.} \end{cases} \tag{5}$$

Based on these maps, we present Algorithm 2, the second version of WJSMA that generates targeted adversarial samples by operating on pairs of components.

In the two previous algorithms, the selected components are always augmented by a positive default value, i.e. features are increased. It is possible to deduce two versions of Algorithms 1 and 2 where relevant components are selected and then decreased according to a similar logic.

3 Taylor Jacobian-based Saliency Map Attack (TJSMA)

This section presents Taylor JSMA, the second contribution of this paper. The idea of this attack is to additionally penalise the choice of feature components that are close the maximum value of features θ_{max} and favour components that

Algorithm 2 Generating adversarial samples by WJSMA: version 2

Inputs: Same inputs as Algorithm 1.
Output: x^*: adversarial sample to x.

$x^* \leftarrow x$
$\texttt{iter} \leftarrow 0$
$\Gamma \leftarrow \{(p,q), p, q \in [\![1, |x|]\!], x[p] \neq \theta_{max}, x[q] \neq \theta_{max}\}$
while class$(x^*) \neq t$ **and** $\texttt{iter} < \texttt{maxIter}$ **and** $\Gamma \neq \emptyset$ **do**
  $(p_{\text{max}}, q_{\text{max}}) = \text{argmax}_{p,q \in \Gamma} S^W[x^*, t](p, q)$
  Modify x^* by $x^*[a] = \texttt{Clip}_{[\theta_{\text{min}}, \theta_{\text{max}}}(x^*[a] + \theta),\ a = p_{\text{max}}, p_{\text{max}}$
  Remove $(p_{\text{max}}, q_{\text{max}})$ from Γ
  $\texttt{iter} + +$
end while
return x^*

are more distant from θ_{max}. As a motivating situation, assume two components i and j have the same WJSMA score $S^W[x,t][i]$ and $S^W[x,t][j]$ and that x_i is very close to θ_{max}, while x_j is far enough from θ_{max}. In this case, searching for more impact, our saliency maps prefer x_j over x_i. Concretely, we consider maximising the two scores: $S_1 = \theta_{\text{max}} - x_i$ and $S_2 = \dfrac{\partial}{\partial x_i} \log p_t(x)$ which is translated into maximising $S = S_1 S_2$.

Accordingly, we introduce new saliency maps for one- and two-components attacks as follows.

$$S^T[x,t][i] = \begin{cases} 0 & \text{if } \alpha_i < 0 \text{ or } \beta_i > 0 \\ \alpha_i |\beta_i| & \text{otherwise.} \end{cases} \tag{6}$$

where

$$\alpha_i = (\theta_{\text{max}} - x_i) \frac{\partial F_c(x)}{\partial x_i}, \quad \beta_i = \sum_{k \neq t} p_k(x) (\theta_{\text{max}} - x_i) \frac{\partial F_k(x)}{\partial x_i}$$

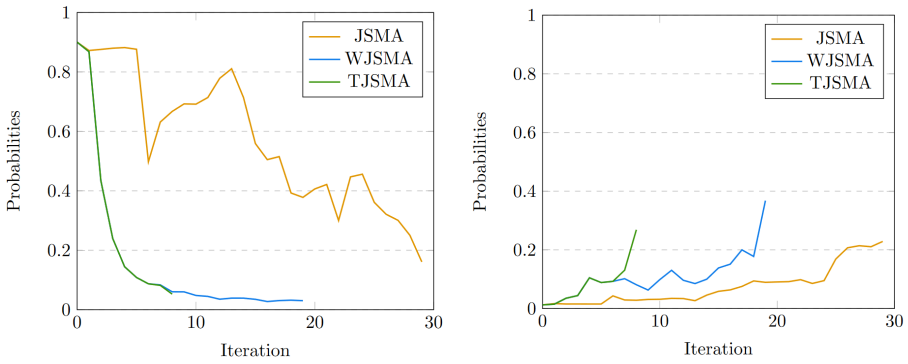
and

$$S^T[x,t][i,j] = \begin{cases} 0 & \text{if } \alpha_{i,j} < 0 \text{ or } \beta_{i,j} > 0 \\ \alpha_{i,j} |\beta_{i,j}| & \text{otherwise.} \end{cases} \tag{7}$$

where

$$\alpha_{i,j} = \sum_{a \in \{i,j\}} (\theta_{\text{max}} - x_a) \frac{\partial F_t(x)}{\partial x_a}, \quad \beta_{i,j} = \sum_{k \neq t} \sum_{a \in \{i,j\}} p_k(x) (\theta_{\text{max}} - x_a) \frac{\partial F_k(x)}{\partial x_a}$$

We call these maps Taylor saliency maps because of the Taylor terms $(\theta_{\max} - x_a)\dfrac{\partial F_k(x)}{\partial x_a}$. One and two-components TJSMA follow exactly Algorithms 1 and 2 with only S^W replaced with S^T.



(a) Origin class (b) Target class

Fig. 3: Evolution of the origin and target class probabilities till the target class is reached for JSMA, WJSMA and TJSMA changing the image of a one into a five.

Through Figures 3a and 3b, we observe that WJSMA and TJSMA decrease/increase the predicted/targeted probability of the original/targeted class much sooner than JSMA. In this example, it is worth noting how TJSMA behaves like WJSMA until it is able to find a more vulnerable component that makes it converge much faster.

4 Experiments

In the following, we give attacks and defense applications to illustrate the interest of WJSMA and TJSMA over JSMA. In doing so, we compare WJSMA and TJSMA and report better results for TJSMA despite that for a large part of samples WJSMA outperforms TJSMA. We use the following standard datasets:

MNIST [10]. This dataset contains 70,000 28×28 greyscale images in 10 classes, divided into 60,000 training images and 10,000 test images. The possible classes are digits from 0 to 9.

CIFAR-10 [7]. This dataset contains 60,000 $32 \times 32 \times 3$ RGB images. There are 50,000 training images and 10,000 test images. These images are divided

into 10 different classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck), with 6,000 images per class.

Figures 4 and 5 display one sample per class, from MNIST and CIFAR-10 respectively.

Fig. 4: MNIST image examples

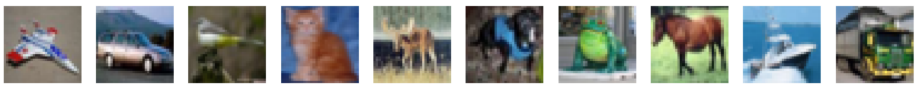


Fig. 5: CIFAR-10 image examples

On each dataset, a deep neural network classifier (DNN) is trained and the performances of the three attacks are evaluated on it.

DNN on MNIST. For the first experiment, we use LeNet-5 [9,15], whose architecture is given in the supplementary material.

We implement and train this model using a `cleverhans` model that optimises crafting adversarial examples. The number of epochs is fixed to 20, the batch-size to 128, the learning rate to 0.001 and the Adam optimizer is used. Training results in a 100% accuracy on the training dataset and 99.49% accuracy on the test dataset.

DNN on CIFAR-10. For the second experiment, a more complex DNN is trained to reach a good performance on CIFAR-10 which is more challenging than MNIST. Its architecture is inspired by the AllConvolutional model proposed in `cleverhans` and is described in the supplementary material..

Likewise, this model is implemented and trained using `cleverhans` for 10 epochs, with a batch size of 128, a learning rate of 0.001 and the Adam optimizer. Training results in a 99.96% accuracy on the training dataset and 83.81% accuracy on the test dataset.

To compare our results with [15], we use the original implementation of JSMA available in `cleverhans`. We have also adapted the code to WJSMA and TJSMA obtaining fast implementations of these two attacks. We only test

the attacks (i.e. Algorithm 2 in the three formats: original, weighted and Taylor) on samples that are correctly predicted by their respective neural networks. In this way, the attacks are applied to the whole training set and the 9,949 well-predicted images of the MNIST test dataset. Similarly, CIFAR-10 adversarial examples are crafted from the well-predicted 9,995 images of the first training 10,000 images and the 8,381 well-predicted test images.

To compare the three attacks, we rely on the notion of maximum distortion of adversarial samples defined as the ratio of altered components to the total number of components. Following [15], we choose a maximum distortion of $\gamma = 14.5\%$ on the adversarial samples from MNIST, corresponding to $\texttt{maxIter} = \lfloor \frac{784*\gamma}{2*100} \rfloor$. On CIFAR-10, we fix $\gamma = 3.7\%$ in order to have the same maximum number of iterations for both experiments. This allows a comparison between the attacks in two different settings. Furthermore, for both experiments, we set $\theta = 1$ (note that $\theta_{\text{min}} = 0,\ \theta_{\text{max}} = 1$).

We report the metrics:

(1) Success rate: This is the percentage of successful adversarial examples, i.e crafted before reaching the maximal number of iterations `maxIter`,
(2) Mean L_0 distance: This is the average number of altered components of the successful adversarial examples,
(3) Strict dominance of an attack: Percentage of adversarial attacks for which this attack does strictly less iterations than the two other attacks[4],
(4) Run-time of an attack on a set of samples targeting every possible class.

Results on the metrics (1) and (2) are shown in Table 1 for MNIST and Table 2 for CIFAR-10.

Table 1: Comparison between JSMA, WJSMA and TJSMA on MNIST.

Metric	**JSMA**	**WJSMA**	**TJSMA**
Targeted (Training dataset: Nb of well predicted images=60,000)			
Success rate	87.68%	**97.14%**	**98.66%**
Mean L_0 distance on successful samples	44.34	**37.86**	**35.22**
Targeted (Test dataset: Nb of well predicted images=9,949))			
Success rate	87.34%	**96.98%**	**98.68%**
Mean L_0 distance on successful samples	44.63	**38.10**	**35.50**

Table 2: Comparison between JSMA, WJSMA and TJSMA on CIFAR-10.

Metric	**JSMA**	**WJSMA**	**TJSMA**
Targeted (Training dataset: Nb of well predicted images=9 995)			
Success rate	86.17	**95.91%**	**97.40%**
Mean L_0 distance on successful samples	47	**38.54**	**36.86**
Targeted (Test dataset: Nb of well predicted images=8 381))			
Success rate	84.91	**94.99%**	**96.96%**
Mean L_0 distance on successful samples	46.13	**38.82**	**37.45**

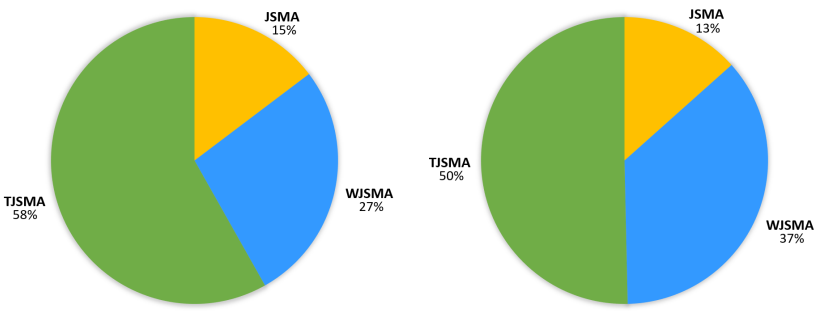
Overall, WJSMA and TJSMA significantly outperform JSMA according to the metrics (1)-(2).

On MNIST. Results in terms of success rate are quite remarkable for WJSMA and TJSMA respectively outperforming JSMA with near $9.46, 10.98$ percentage points (pp) on the training set and $9.46, 11.34$ pp on the test set. The gain in the average number of altered components exceeds 6 components for WJSMA and 9 components for TJSMA in both experiments.

On CIFAR-10. Similar results are obtained on this dataset. WJSMA and TJSMA outperform JSMA in success rate by near $9.74, 11.23$ pp on the training set and more than $10, 12$ pp on the test set. For both training and test sets, we report better mean L_0 distances exceeding 7 features in all cases and up to 10.14 features for TJSMA on the training set.

Dominance of the attacks. The next figures illustrate the (strict) dominance of the attacks for the two experiments. In these statistics, we do not count the samples for which TJSMA and WJSMA realise the same number of iterations strictly less than JSMA.

[4] As additional results, we give in the supplementary material more statistics on the dominance between any two attacks.



(a) MNIST (b) CIFAR-10

Fig. 6: Distribution of the (strict) dominance of JSMA, WJSMA and TJSMA over the MNIST and CIFAR10 datasets (training and test sets included)

For both experiments, TJSMA has a noteworthy advantage over WJSMA and JSMA. The advantage of WJSMA over JSMA is also considerable. This shows that, in most cases, WJSMA and TJSMA craft better adversarial examples than JSMA, while being faster. Our results are actually better when directly comparing WJSMA or TJSMA with JSMA. As additional results, we give in the supplementary material the statistics for the pairwise dominance between the attacks. As it might be expected, both WJSMA and TJSMA dominate JSMA and TJSMA dominate WJSMA.

Run-time comparison. In order to have a meaningful speed comparison between the three attacks, we evaluated the run-time needed for each attack to successfully craft the first 1,000 test images of MNIST in the targeted mode. Results shown in Table 3 reveal that TJSMA and WJSMA are 1.41 and 1.28 times faster than JSMA. These performance tests were realised on a machine equipped with a Intel Xeon 6126 processor and a Nvidia Tesla P100 graphics processor. Based on a previous analysis [1], TJSMA and WJSMA are at least 28 and 24 times faster than L_0 CW attack. Note that for WJSMA and TJSMA, the additional computations of one iteration compared to JSMA are negligible (simple multiplications). Thus the difference in speed between the attacks is mainly due to the number of iterations for each attack.

Table 3: Time comparison between JSMA, WJSMA and TJSMA

Attack	**JSMA**	**WJSMA**	**TJSMA**
Time (second)	3964	**3092**	**2797**

Note that to compare the attacks, the adversarial samples were crafted one by one. In practice, it is possible to generate samples by batch. In this case, the algorithm stops when all samples are treated. Most of the time, with a batch of large size, the three attacks approximately take the same time to converge. For example, on the same machine as previously, with a batch size equal 1000, we were able to craft the same amount of samples in about 250s, for all the attacks.

5 Defense

The objective of this section is to train neural networks in a way that the attacks fail as much as possible. One way of doing that is by adding adversarial samples crafted by JSMA, WJSMA and TJSMA to the training set. This way of training may imply a decrease in the model accuracy but adversarial examples will be more difficult to generate.

We experiment with this idea on MNIST with LeNet-5 in every possible configuration. To this end, 2,000 adversarial samples per class (20,000 more images in total), with distortion under 14.5%, are added to the original MNIST training set, crafted by either JSMA, WJSMA or TJSMA. Then, three distinct models are trained on these three augmented datasets. The models roughly achieve an accuracy of 99.9% on the training set and 99.3% on the test set, showing a slight loss compared to our previous MNIST model accuracy. Nevertheless, the obtained neural networks are more robust to the attacks as shown in the following Table 4. Note that each experiment is made over the well-predicted samples of the test images. For each model and image, nine adversarial examples are generated by the three attacks.

Table 4: Metrics (1) and (2) on JSMA, WJSMA and TJSMA augmented sets

Metric	**JSMA**	**WJSMA**	**TJSMA**
Model trained over JSMA augmented set (9940 well predicted samples)			
Success rate	77.94%	**84.79%**	**85.08%**
Mean L_0 distance on successful samples	54.48	**52.66**	**52.83**
Model trained over WJSMA augmented set (9936 well predicted samples)			
Success rate	77.61%	**90.05%**	**92.01%**
Mean L_0 distance on successful samples	56.29	**52.72**	**52.18**
Model trained over TJSMA augmented set (9991 well predicted samples)			
Success rate	76.42%	**86.18%**	**87.36%**
Mean L_0 distance on successful samples	54.26	**54.20**	**54.49**

Overall, the attacks are less efficient on each of these models, compared to Table 1. The success rates drop by about 8pp, whereas the number of iterations is increased by approximately 26%. From the defender's point of view, networks trained against JSMA and TJSMA give the best performance. The JSMA trained model provides the lowest success rates while the TJSMA trained network is more robust from the L_0 distance point of view. From the attacker's point of view, TJSMA remains the most efficient attack of the three regardless of the augmented dataset used.

6 Avoid confusion

In this section, we argue that our results do not contradict [15]. First, we stress that we use a more performant LeNet-5 model than the one in [15] (with 98.93% and 99.41% accuracies on the training and test sets). For completeness, we also generated a less performant model (with 99.34% and 98.94% accuracies on the training and test sets) and evaluated the three attacks on it through the first $1,000$ test MNIST images. We obtain 96.7% success rate for JSMA (very similar to [15]) and more than 99.5% for WJSMA and TJSMA. These results are also included in our experiments. Instead of presenting two models, we preferred to use the more performant one as this makes the paper shorter and moreover it values more our approach (giving us more advantage with respect to JSMA). Finally, we notice that for both experiments and contrary to [15] (see Appendix A in [15]), our results were obtained without simplifications on the model which is an additional advantage of our attacks.

7 Conclusion

This paper has introduced WJSMA and TJSMA new probabilistic adversarial attacks variants of JSMA. It has demonstrated that WJSMA and TJSMA significantly outperform JSMA on two standard DNNs on MNIST and CIFAR-10 after analysing more than $88,200 \times 9$ adversarial images. Also, it has demonstrated that defending against WJSMA and TJSMA is more advantageous than against JSMA. It is important to recall that our attacks are derived quite naturally from a classical log softmax reasoning and benefit from substantial investigations of doubly-indexed saliency maps. Based on the analysis of 9,000 adversarial samples, WJSMA and TJSMA are at least 1.2 and 1.4 times faster than JSMA and accordingly at least 24 and 28 times faster than L_0 CW attack. We believe these results are quite reassuring and make the new attacks promising tools for future applications. Finally, non-targeted versions of our attacks

have not been discussed in this paper and may be subject of future work and comparison with existing approaches such as [23].

Acknowledgements.

This collaboration was done in the context of an internship by A. Loison and T. Combey supervised by H. Hajri. We thank Gabriel Zeller for his assistance. We are grateful to Wassila Ouerdane and Jean-Philippe Poli at CentraleSupélec for their support. We thank the mesocentre de calcul Fusion, Metz computing center of CentraleSupélec and Stéphane Vialle for providing us effective computing ressources. H. Hajri is grateful to the scientific direction of IRT SystemX for its support and Sylvain Lamprier for very useful discussions.

References

1. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. CoRR **1608.04644v2** (2017), https://arxiv.org/pdf/1608.04644v2
2. Chio, C., Freeman, D.: Machine learning and security. Oreilly (2018)
3. Deng, J., Dong, W., Socher, R., Li, L., Kai Li, Li Fei-Fei: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009)
4. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 1625–1634 (2018)
5. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. ICLR **1412.6572v3** (2015), https://arxiv.org/pdf/1412.6572v3
6. Jakubovitz, D., Giryes, R.: Improving DNN robustness to adversarial attacks using jacobian regularization. CoRR **abs/1803.08680** (2018), http://arxiv.org/abs/1803.08680
7. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research) http://www.cs.toronto.edu/~kriz/cifar.html
8. Kurabin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. ICLR **1607.02533v4** (2017), https://arxiv.org/pdf/1607.02533v4
9. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. pp. 2278–2324 (1998)
10. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), http://yann.lecun.com/exdb/mnist/
11. Lin, J., Dzeparoska, K., Zhang, S.Q., Leon-Garcia, A., Papernot, N.: On the robustness of cooperative multi-agent reinforcement learning. ArXiv **abs/2003.03722** (2020)
12. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks **1706.06083v3** (2017), https://arxiv.org/pdf/1706.06083v3
13. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool : a simple and accurate method to fool deep neural networks. CoRR **1511.04599** (2015), https://arxiv.org/pdf/1511.04599
14. Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 86–94 (2017). https://doi.org/10.1109/CVPR.2017.17, https://doi.org/10.1109/CVPR.2017.17
15. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Berkay Celik, Z., , Swami, A.: The limitations of deep learning in adversarial settings. IEEE **1511.07528v1** (2015), https://arxiv.org/pdf/1511.07528v1
16. Papernot, N., Faghri, F., Carlini, N., Goodfellow, I.J., Feinman, R., Kurakin, A., Xie, C., Sharma, Y., Brown, T.H., Roy, A., Matyasko, A., Behzadan, V., Hambardzumyan, K., Zhang, Z., Juang, Y.L., Li, Z., Sheatsley, R., Garg, A., Uesato, J., Gierke, W., Dong, Y., Berthelot, D., Hendricks, P.N.J., Rauber, J., Long, R., McDaniel, P.D.: Technical report on the cleverhans v2.1.0 adversarial examples library (2016)
17. Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., Erlingsson, Ú.: Scalable private learning with PATE. CoRR **abs/1802.08908** (2018), http://arxiv.org/abs/1802.08908
18. Parisi, A.: Hands-on artificial intelligence for cybersecurity. Packt Publishing (2019)

19. Sethi, K., Edupuganti, S., Kumar, R., Bera, P., Madhav, Y.: A context-aware robust intrusion detection system: a reinforcement learning-based approach. International Journal of Information Security (12 2019). https://doi.org/10.1007/s10207-019-00482-7
20. Sitawarin, C., Bhagoji, A.N., Mosenia, A., Chiang, M., Mittal, P.: DARTS: deceiving autonomous cars with toxic signs. CoRR **abs/1802.06430** (2018), http://arxiv.org/abs/1802.06430
21. Song, L., Shokri, R., Mittal, P.: Privacy risks of securing machine learning models against adversarial examples. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019. pp. 241–257 (2019). https://doi.org/10.1145/3319535.3354211, https://doi.org/10.1145/3319535.3354211
22. Springenberg, J., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. In: ICLR (workshop track) (2015), http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15a
23. Wiyatno, R., Xu, A.: Maximal jacobian-based saliency map attack **1808.07945v1** (2018), https://arxiv.org/pdf/1808.07945v1
24. Wiyatno, R., Xu, A.: Maximal jacobian-based saliency map attack. CoRR **abs/1808.07945** (2018), http://arxiv.org/abs/1808.07945
25. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)

8 Supplementary material

Architectures of the DNNs.

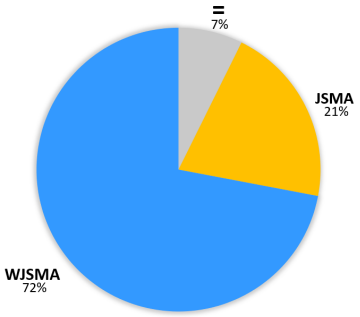
Table 5: LeNet-5 architecture

Layer	**Parameters**
Input Layer	size: (28×28)
Conv2D	kernel size: (5×5), 20 kernels, no stride
ReLu	
MaxPooling2D	kernel size: (2×2), stride: (2×2)
Conv2D	kernel size: (5×5), 50 kernels, no stride
ReLu	
MaxPooling2D	kernel size: (2×2), stride: (2×2)
Flatten	
Dense	size: 500
ReLu	
Dense	size: number of classes (10 for MNIST)
Softmax	

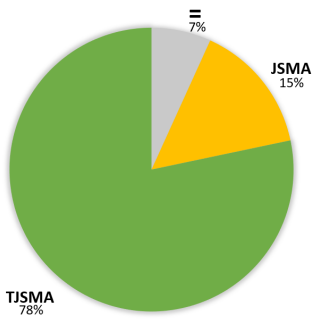
Table 6: Architecture of the used DNN on CIFAR-10

Layer	**Parameters**
Input Layer	size: (32×32)
Conv2D	kernel size: (3×3), 64 kernels, no stride
ReLu	
Conv2D	kernel size: (3×3), 128 kernels, no stride
ReLu	
MaxPooling2D	kernel size: (2×2), stride: (2×2)
Conv2D	kernel size: (3×3), 128 kernels, no stride
ReLu	
Conv2D	kernel size: (3×3), 256 kernels, no stride
ReLu	
MaxPooling2D	kernel size: (2×2), stride: (2×2)
Conv2D	kernel size: (3×3), 256 kernels, no stride
ReLu	
Conv2D	kernel size: (3×3), 512 kernels, no stride
ReLu	
MaxPooling2D	kernel size: (2×2), stride: (2×2)
Conv2D	kernel size: (3×3), 10 kernels, no stride
GlobalAveragePooling	kernel size: (2×2), stride: (2×2)
Softmax	

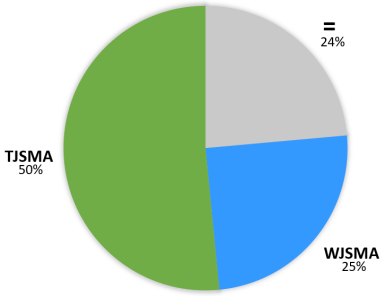
Pairwise dominance.



(a) WJSMA vs JSMA

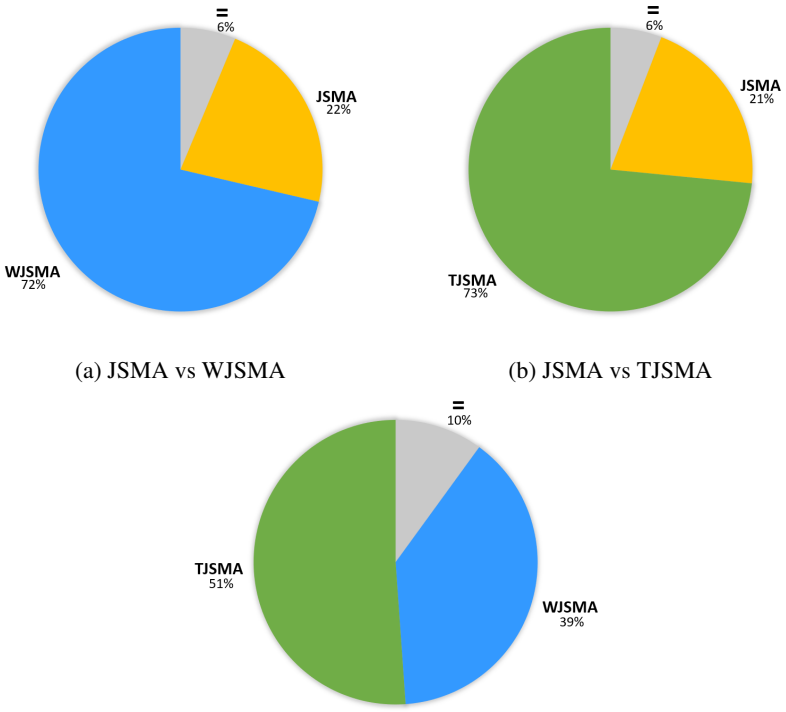


(b) TJSMA vs JSMA



(c) TJSMA vs WJSMA

Fig. 7: Pairwise dominance on MNIST (= corresponds to samples with the same number of iterations by the attacks including when both attacks fail).



(a) JSMA vs WJSMA

(b) JSMA vs TJSMA

(c) WJSMA vs TJSMA

Fig. 8: Pairwise dominance on CIFAR-10 (= has the same significance as before).

Supplementary comments. Further analysis of the results on MNIST reveals that, even for examples where JSMA is better than WJSMA or TJSMA, in average, less than 10 more components are changed by WJSMA or TJSMA, whereas JSMA changes more than 17 more components in average when it is dominated by WJSMA or TJSMA. A similar gap can be remarked in CIFAR-10.