Neural Compression Restoration Against Black-box Adversarial Attacks

Lorenzo Agnolucci
Università degli Studi di Firenze
lorenzo.agnolucci@stud.unifi.it

Abstract

The study of adversarial attacks and defense strategies against them has gained increasing attention in the last years, mainly due to the importance of neural networks classifiers in several real-world applications. In this work the defense strategy proposed in [2], which achieved excellent results against white-box attacks, is evaluated against black-box attacks. In particular, we consider Hop Skip Jump and Square attacks. The attacks are performed with a limited budget of queries and at various thresholds of l_2 perturbation error. The defense mechanism proved to be effective against both of the attacks.

1. Introduction

Although deep neural networks have achieved state-of-the-art performance on a variety of tasks, they have proved to be vulnerable to adversarial examples, *i.e.* slightly perturbed versions of the original examples that manage to cause models to make wrong predictions. A peculiarity of adversarial examples is that they are almost identical to the original samples for the human eye, because they are specifically crafted to have the minimum possible perturbation. The vulnerability of neural networks to adversarial examples implies a security risk in various security-sensitive real-world applications, such as self-driving cars, robotics, finance and healthcare. Thus the study of the adversarial robustness issue has attracted rising attention with an increasing number of adversarial attack and defense methods proposed.

The goal of an adversarial attack is to make the target classifier predict a wrong label. A classifier can be denoted as a function $\mathcal{C}(x) : \mathcal{X} \rightarrow \mathcal{Y}$, where $x \in \mathcal{X}$ is the input image that is mapped to a class label $y \in \mathcal{Y} = \{1, 2, \ldots, C\}$, with C number of classes. Let y^* denote the ground-truth label of the clean input x, and x_a denote an adversarial example of x. An adversarial attack can be untargeted or targeted. The former simply aims to cause a misclassification, *i.e.* $\mathcal{C}(x_a) \neq y^*$, while the latter tries to achieve $\mathcal{C}(x_a) = \tilde{y}$, with $\tilde{y}$ given label and $\tilde{y} \neq y^*$.

Normally, the adversarial example x_a should be crafted so that it is as similar as possible to its corresponding clean image, and this can be achieved with two strategies [5]. The first seeks to craft an adversarial example x_a that satisfies $\|x_a - x\|_p \leq \epsilon$, where ϵ is the perturbation error, and misleads the model. This can be achieved by solving a constrained optimization problem. For instance, the adversary can get an untargeted adversarial example by minimizing a loss function $\mathcal{J}$ in the restricted region as

$$x_a = \underset{x' \colon \|x' - x\|_p \leq \epsilon}{\operatorname{argmin}} \mathcal{J}(x', y) \tag{1}$$

This is called an adversarial example with a constrained perturbation. The second strategy is generating an adversarial example by finding the minimum perturbation as

$$x_a = \underset{x' \colon x' \text{ is adversarial}}{\operatorname{argmin}} \|x' - x\|_p \tag{2}$$

This is called an adversarial example with an optimized perturbation. However, it is usually intractable to solve equation (1) or equation (2) exactly, and thus various attack methods have been proposed to get an approximate solution.

2. Black-box Attacks

Both targeted and untargeted attacks can be conducted in white-box and black-box settings.

White-box attacks rely on detailed information of the target model, including architecture, parameters, gradient of the loss w.r.t. the input and also possible defense mechanisms. For this reason white-box attacks

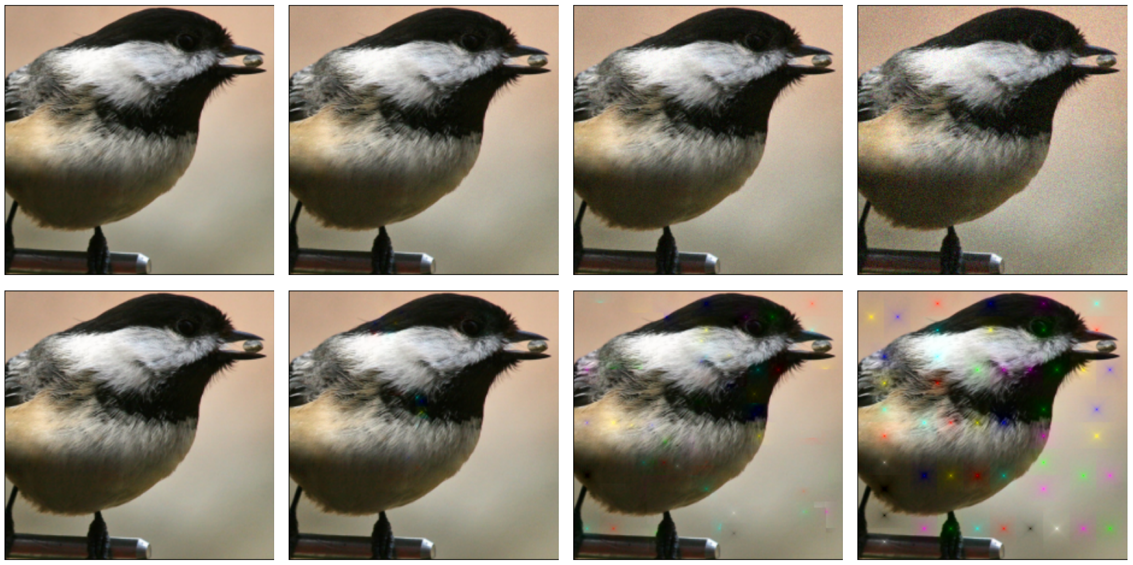

Figure 1: Adversarial example for different perturbation budgets. Top row is generated with Hop Skip Jump, bottom row with Square

usually employ gradient-based strategies to generate adversarial examples.

Black-box attacks have not any knowledge on the model architecture or parameters and use various strategies. Transfer-based black-box attacks are based on the adversarial transferability, which, assuming the availability of training data, is utilized to generate adversarial examples on a previously trained surrogate model. Score-based black-box attacks can only acquire the output probabilities by querying the target model, while decision-based black-box attacks solely rely on the predicted classes of the queries. Score-based and decision-based attacks usually try to approximate gradients to generate adversarial examples.

The fact that a black-box attack only has access to the output of the model makes this type of attack suitable to a lot of real-world applications, e.g. the Google Cloud Vision API, which output the predictions without providing details about the model. Therefore it is more realistic to evaluate the vulnerability of a machine learning system with a limited budget of model queries. Indeed online image classification platforms often set a limit on the allowed number of queries within a certain time period. Query inefficiency thus leads to clock-time inefficiency and prevents an attacker from carrying out large-scale attacks. Last but not least, a smaller query budget directly implies less cost in evaluation and research of new defense mechanisms.

2.1. Hop Skip Jump Attack

The Hop Skip Jump attack [4] is an iterative decision-based attack based on the Boundary attack [3]. It tries to solve equation (2) on the preceding page, so it starts from an adversarial example with a high perturbation error and lowers it iteratively.

It is initialized with a sample blended with uniform noise that is misclassified for untargeted attack. Each iteration of the algorithm has three components. First, the iterate from the last iteration is pushed towards the boundary between successful and unsuccessful perturbed images via a binary search. Second, the gradient direction is estimated via the Monte Carlo method. Third, the updating step size along the gradient direction is initialized with a value that assures the convergence to a stationary point of the problem, and is decreased via geometric progression until perturbation becomes successful. The next iteration starts with projecting the perturbed sample back to the boundary again.

Figure 2 on the next page provides an intuitive visualization of an iteration of the attack in l_2. Top row of figure 1 shows some adversarial examples generated with the Hop Skip Jump attack.

2.2. Square Attack

The Square attack [1] is a score-based black-box l_2 and l_∞ adversarial attack which aims to solve equation (1) on the previous page. It does not rely on local

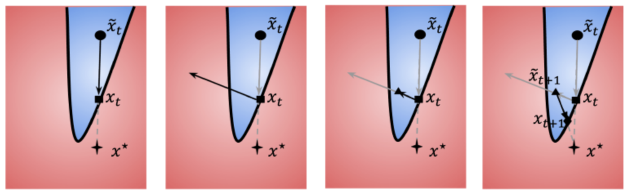

Figure 2: Intuitive explanation of the Hop Skip Jump attack. (a) Perform a binary search to find the boundary, and then update the current adversarial example $\widetilde{x}_t \rightarrow x_t$ (b) Estimate the gradient at the boundary point x_t (c) Geometric progression and then update $x_t \rightarrow \widetilde{x}_{t+1}$ (d) Perform a binary search, and then update $\widetilde{x}_{t+1} \rightarrow x_{t+1}$

gradient information so it is not affected by gradient masking based defense techniques. The Square attack is based on a randomized search scheme which selects localized square-shaped updates at random positions so that at each iteration the perturbation is situated approximately at the boundary of the feasible set (*i.e.* the l_p-ball $\{x_a : \|x_a - x\|_p \leq \epsilon\}$). The main idea of the algorithm is to sample a random update δ at each iteration, and to add this update to the current iterate x_a if it achieves an improvement in terms of a loss function. Due to a particular sampling distribution, the Square attack requires significantly fewer queries compared to a lot of black-box methods while outperforming them in terms of success rate, *i.e.* the percentage of successful adversarial examples.

The algorithm is shown in Algorithm 1, while some examples of its applications are presented in the bottom row of figure 1 on the preceding page.

Algorithm 1: Square attack

```
Input: classifier $\mathcal{C}$, image $x$, image size $w$,
       number of channels $c$, $l_p - radius\, \epsilon$,
       label $y$, number of iterations $\mathcal{N}$, loss $\mathcal{L}$
Output: Approximate minimizer $x_a$ of
        equation (1) on page 1
1 $x_a \leftarrow init(x)$, $l^* \leftarrow \mathcal{L}(\mathcal{C}(x), y)$, $i \leftarrow 1$
2 while $i < \mathcal{N}$ and $x_a$ not adversarial do
3     $h^{(i)} \leftarrow$ side length of the square to modify
4     $\delta \sim P(\epsilon, h^{(i)}, c, x_a, x)$
5     $x_a^{new} \leftarrow$ Project $x_a + \delta$ onto $\{z : \|z - x\|_p \leq \epsilon\}$
6     $l_{new} \leftarrow \mathcal{L}(\mathcal{C}(x_a^{new}), y)$
7     if $l_{new} < l^*$ then $x_a \leftarrow x_a^{new}$, $l^* \leftarrow l_{new}$
8     $i \leftarrow i + 1$
9 end
```

3. Defense strategy

The approach proposed in [2] is based on JPEG compression and generative restoration models. The input image is subjected to JPEG compression, a non-differentiable input transformation that reduces adversarial perturbations. However, the compression, especially with very low quality factors, has itself a negative impact on the classification because it degrades the image. To counter this effect, restoration GAN models previously trained at different quality factors are used to restore image quality and remove JPEG compression artifacts. This solution has two positive sides: first, the quality factor can be chosen randomly and the most appropriate GAN can be plugged in on the fly; second, the probability of correctly classifying compressed and restored (non attacked) images is higher even for lower quality factors, because each model is trained specifically for that factor. Also, it is possible to iterate the compression and restoration steps an arbitrary number of times without degrading the image but increasing the complexity for the attacker.

The pipeline of the defense strategy is the following:

1. Apply JPEG compression to the input image with a random quality factor (QF)

2. Choose the GAN restoration model trained with the QF closest to the one chosen randomly at the previous step

3. Repeat the process for a given number of iterations

Figure 3 on the following page shows an example of the application of the defense strategy.

4. Experimental Results

The attacks were executed with the implementations of the *Adversarial Robustness Toolbox (ART)* library [6]. All the attacks performed in this work were untargeted and had a limited budget of 5k queries. The query budget was chosen considering the limited computational resources available. Indeed, each attack on each image took a long time to complete and a higher budget would have been too time-consuming. Regarding the defense strategy, the tests were carried out with ResNet50 as the architecture, random QF in the range [20, 60], GANs trained with QF of 20, 40 and 60 and 3 iterations of the defense pipeline.

4.1. Dataset

The dataset is based on the ILSVRC validation set [7], which has 1000 classes and $224 \times 224 \times 3$ images. A subset of 1000 images (*i.e.* one image for each class) is

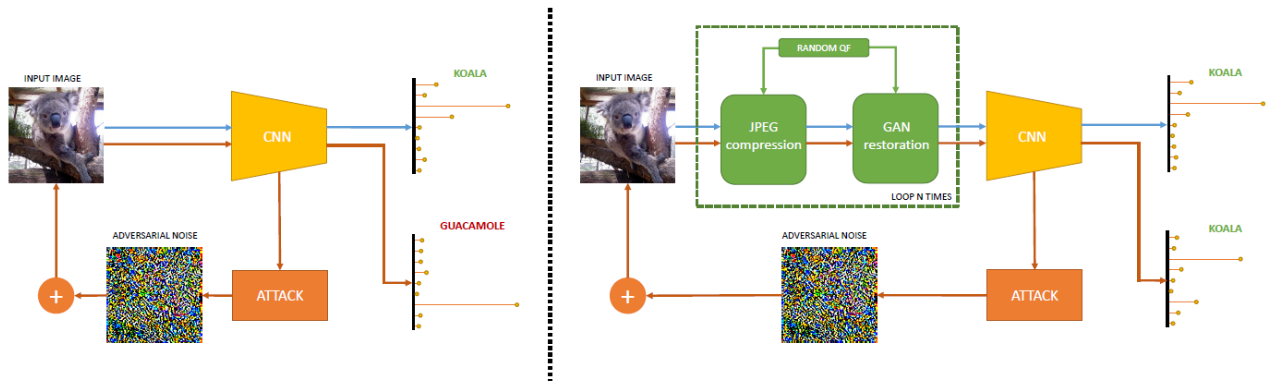

Figure 3: Overview of the defense strategy pipeline. On the left, a successful adversarial attack with no defense. On the right, the defense strategy proposed in [2]

selected such that the chosen classifier achieves 100% accuracy on non-attacked images. This approach is due to the fact that using already misclassified images to evaluate the defense would be useless, because the attack would be successful by definition.

4.2. Evaluation Metrics

The performance of the defense is evaluated with two metrics: the degradation and the accuracy vs perturbation budget. The former quantifies the effect of the defense on the classification accuracy of non-attacked images, the latter measures the robustness of the defense strategy increasing the degradation applied to the image by the adversarial attack. Both of these metrics are evaluated for three defense strategies: no defense, simple JPEG compression with fixed 40 QF and the approach proposed in [2].

To measure the amplitude of the perturbation we used the normalized dissimilarity $l_2(x, x_a) = \frac{\|x-x_a\|_2}{\|x\|_2}$, with x and x_a respectively the original and the adversarial image. A possible alternative commonly used in literature is the l_∞ perturbation metric, usually at a standard threshold of 0.031, *i.e.* 8/256. In this work the l_2 distance is preferred over the l_∞ one because is less strict so it lets to evaluate the defense in a more difficult environment. Indeed, given the large number of pixels of ImageNet images, the l_2 dissimilarity is a weaker constraint because it represents an average measurement of the error over the whole image, while the l_∞ provides the maximum error across all the pixels and it is a stricter constraint for the attack.

4.3. Hop Skip Jump Attack

Given that the Hop Skip Jump attack tries to find the solution of equation (2) on page 1, the original implementation of the algorithm did not include a perturbation budget, so it was modified to stop after reaching the maximum number of queries or after finding an adversarial example with a l_2 perturbation smaller than the budget.

The chosen budget parameter was *0.01*, the lowest of the ones considered, because in this way for each image we can obtain the lowest possible error with the given query limit. So the attack is considered successful if it achieves a perturbation error lower than the considered perturbation threshold. Regarding the other parameters of the Hop Skip Jump attack, we used the ones provided in the authors' implementation, that is:

- initial number of evaluations: *100*
- max number of evaluations: *10000*
- max number of evaluations for the initial example: *100*

Table 1 on the following page shows the results obtained for different perturbation budgets. The approach proposed in [2] proved to be very effective against the Hop Skip Jump attack. Even for easily recognizable distortions the defense was robust and resulted in a correct classification.

4.4. Square Attack

Since the Square attack outputs an adversarial example with constrained perturbation, the original implementation did not need to be modified to include a perturbation budget. The attack is considered successful if it finds an adversarial example before reaching the query limit, given that by definition of the algorithm this example will have a perturbation error lower than the budget.

Starting from a perturbation threshold equal to *0.08* (*i.e.* the biggest of the ones considered), we lowered it

Attack	Defense	No attack	$l_2 = 0.01$	$l_2 = 0.03$	$l_2 = 0.05$	$l_2 = 0.06$	$l_2 = 0.07$	$l_2 = 0.08$
Hop Skip Jump	No defense	**100**	20.9	0.4	0.1	0.1	0.1	0.1
	JPEG 40	99.4	97.8	94.2	86.2	80.9	74.4	68.7
	[2] approach	98.8	**98.6**	**96.5**	**94.4**	**93.5**	**92.6**	**91.1**
Square	No defense	**100**	96.3	63.6	19.8	10.0	5.3	2.1
	JPEG 40	99.4	97.0	73.6	38.6	25.8	17.0	8.9
	[2] approach	98.8	**98.0**	**95.8**	**91.2**	**87.6**	**84.3**	**78.6**

Table 1: Classification accuracy for different types of defense and perturbation budgets

only for the images for which the attack was successful. Indeed we can assume that the attack would not be able to find an adversarial example with an even lower perturbation budget. This choice also saves execution time, because the number of images on which the attack is performed decreases with each budget and is considerably lower than the total. Concerning the other parameters of the algorithm, they were chosen according to the authors' implementation. We used:

- number of restarts: *1*
- initial fraction of pixels: *0.1*

The results obtained for different perturbation budgets are provided in table 1. The proposed approach outperforms both the baselines. It is important to emphasize that the proposed defense mechanism behaved extremely well even for perturbation budgets that make the adversarial example clearly different from the corresponding clean image for the human eye.

5. Conclusion

In this work the defense strategy towards adversarial examples proposed in [2], which already proved to be effective against white-box attacks, was tested further against black-box attacks. In particular, for the tests we considered the Hop Skip Jump and the Square attack. The defense mechanism proved to be robust and effective even for high perturbation budgets that make the distortion easily recognizable. This fact make the approach proposed in [2] a strong candidate for security-sensitive real-world applications.

As a future work, the defense strategy could be evaluated against other black-box attacks.

References

[1] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: a query-efficient black-box adversarial attack via random search. *CoRR*, abs/1912.00049, 2019.

[2] F. Becattini, C. Ferrari, and L. Galteri. Neural compression restoration against gradient-based adversarial attacks.

[3] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, 2018.

[4] J. Chen and M. Jordan. HopSkipJumpAttack: A Query-Efficient Decision-Based Adversarial Attack. *CoRR*, abs/1904.02144, 2019.

[5] Y. Dong, Q.-A. Fu, X. Yang, T. Pang, H. Su, Z. Xiao, and J. Zhu. Benchmarking adversarial robustness, 2019.

[6] M. Nicolae, M. Sinn, T. N. Minh, A. Rawat, M. Wistuba, V. Zantedeschi, I. M. Molloy, and B. Edwards. Adversarial robustness toolbox v0.2.2. *CoRR*, abs/1807.01069, 2018.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.