Using Search To Produce Imperceptible Adversarial Examples

Cristopher McIntyre Garcia
Department of Computer Science
University of Ottawa
Ottawa, ON K1N 6N5
`cmcin019@uottawa.ca`

1 Introduction

More and more computer vision models using Machine Learning (ML) are being implemented into products such as smartphones, vehicles, and medical applications. As such, these models need to be properly verified and tested for faults and vulnerabilities. In doing so, the models, and by extension, the products, become safer and more robust against malicious attacks. This project aims to develop a black-box search algorithm to produce imperceptible Adversarial Examples (AE) using a combination of explorative and exploitative methods [6]. This project also aims to determine whether models of different architectures are affected by equal image perturbations, as has been demonstrated with CNNs [7].

2 Related Work

The concept of AE was first introduced by [7] as inputs to a ML model that are intentionally designed to cause misclassification. By applying imperceptible non-random perturbation to images, one can change a network's prediction of a previously correctly classified example. By optimizing the input to maximize the prediction error, AE are trivial to find and are shared by networks with different number of hidden layers or trained on different data subsets. That is, the perturbed images found are also misclassified by CNN models with different configurations. Due to certain properties of back-propagation, models develop intrinsic blind spots that are connected to the data distribution in a non-obvious way.

In [4], it was determined that the linear nature of neural networks is what causes these adversarial vulnerabilities. They also present a fast and simple method of generating AE by adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input. Using this method, they are able to cause a shallow Soft-Max classifier to have an error rate of 99.9% on the MNIST dataset. Similar results were found using more complex models, such as GoogLeNet trained on the ImageNet dataset.

Following these findings, many defences that are robust against optimization-based attacks have been proposed. Though these defences may seem secure at first glance, [1] demonstrates that without a properly computed gradient, iterative optimization-based methods can be circumvented. They proceed to fully circumvent six of the seven defences proposed at ICLR 2018. They also demonstrate that these defences are ineffective against black-box attacks that don't use the gradient of the cost function to generate AE.

Due to these limitations, it is important to develop defences against black-box attacks. One popular decision- based attack is the Boundary Attack (BA), introduced in [2]. In this attack, an adversarial

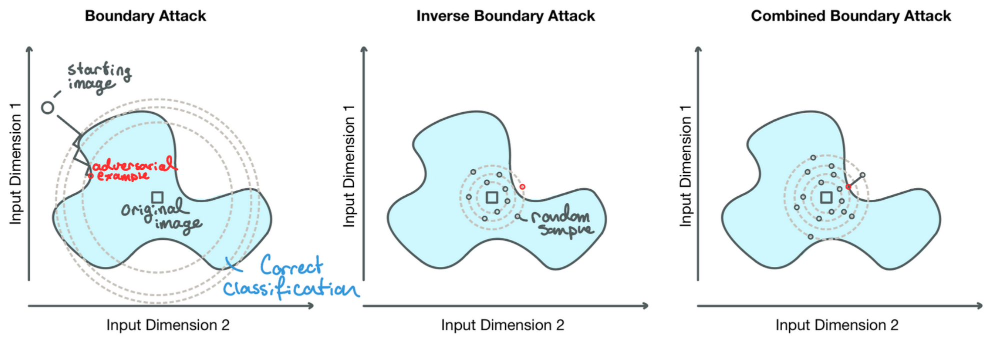

Figure 1: Boundary Attack Variants.

image is produced by taking the original image and adding enough perturbations to make it adversarial. Then, slowly remove portions of the perturbation until it reaches the border of the non-adversarial image. Continue iteratively along the border without augmenting the distance between the current perturbed image and the original. Once it is no longer possible to traverse the border without augmenting the distance between the current perturbed image and the original image, the algorithm returns the perturbed image. The goal of this algorithm is to find AE that are very similar to the original data without any knowledge of the model being used.

3 Research Questions

- Are local search algorithms able to efficiently produce AE? What are the explorative and exploitative properties of such an algorithm?
- Will a transformer falsely classify the same perturbed images as a CNN ? If not, can we use the transformer to keep the perceptibility of perturbation at a minimum?
- Are we able to infer any useful information about the model and its classification by studying its AE?

4 Dataset

For the purpose of this project, we will begin by using the MNIST dataset. If good results are achieved, a more complex dataset such as CIFAR10 or ImageNet will be used.

5 Overview

A Vision Transformer (ViT) model, such as [3], will be trained as a classifier to obtain around 90% accuracy on the MNIST dataset. The same will be done with a CNN model, such as [5]. The latter will sometimes be referred to as the oracle. Both models will then be treated as black-box models, where only the inputs and outputs are known. For each image in the test set, we want to create a perturbed version that will make the ViT model misclassify the image. We also want the accuracy of the oracle to be minimally affected by the perturbations. Results found using the Boundary Attack (BA) algorithm will make for a good baseline. Modifications will be made to the algorithm in the hopes of finding better results. These modifications include starting from a non-adversarial image and finding the boundary that separates adversarial and non-adversarial examples by taking "steps" away from the original image. This algorithm will be referred to as the Inverse Boundary Attack (IBA). Steps are taken when noise is added to or removed from the current perturbed image, as shown in [2]. Another modification is to combine the BA with the IBA. This algorithm will be referred to as the Combined Boundary Attack (CBA). These algorithms are optimization algorithms that aim to minimize the distance between an adversarial example and an original image.

Algorithm 1 Boundary Attack

Input x: original image, y: label, f: classifier, α: move size, $steps$: number of steps
Result $\hat{x}$: adversarial example

```
 1: while True do                    ▷ Draw sample from uniform distribution over input space
 2:     $\hat{x} \sim U(\mathcal{X})$
 3:     if $f(\hat{x}) \neq y$ then
 4:         break
 5:     end if
 6: end while
 7:
 8: for steps do
 9:     while True do                                        ▷ Move $\hat{x}$ towards $x$
10:         $\hat{x}^{new} \leftarrow \alpha x + (1-\alpha)\hat{x}^{old}, (0 \leq \alpha \leq 1)$
11:         if $f(\hat{x}^{new}) = y$ then
12:             break
13:         end if
14:     end while
15:
16:     while True do                                   ▷ Sample around $l_\infty$ ball
17:         $\delta \sim \mathcal{N}(0,1)$
18:         $\hat{x}^{new} \leftarrow \Pi_{T(x,\hat{x}^{old})}(\hat{x}^{old} + \delta)$
19:         if $f(\hat{x}^{new}) \neq y$ then
20:             break
21:         else
22:             reject $\hat{x}^{new}$
23:         end if
24:     end while
25: end for
26: return $\hat{x}$
```

5.1 Boundary Attack

The Boundary Attack (BA) algorithm 1 was presented in [2] as an algorithm for generating AE capable of making a black-box model misclassify the previously correctly classified examples. The intuition of the algorithm is seen in Figure 1. Firstly, we are given an original image and its label, a classifier, an α value, and a number of steps. An image with a label different from that of the original image is randomly sampled from the dataset distribution, which will be the starting point of the adversarial example. Then, the algorithm alternates between moving the adversarial example towards the original image, and shifting it around the original image via a l_∞ ball (Eq. 3) whenever it enters the non-adversarial region. The distance between the adversarial example and the original image is never increased. The intuition is to reach the non-adversarial distribution border and travel along it until the distance between the adversarial example and the original image is minimized.

Moving towards the original image is done by changing the pixels of the adversarial example in a way that minimizes the difference between them and those of the original image. A portion of every pixel in the adversarial example is shifted by a factor of α towards the original image by addition, as defined in Eq. 1. This will bring it closer to the original image in the image space, making it look more like the original image. Enough of these shifts need to be made in order for the adversarial image to look like the original image. If not enough steps are taken, the adversarial example risks having perceptible differences.

$$\hat{x}^{new} = \alpha x + (1-\alpha)\hat{x}^{old}, (0 \leq \alpha \leq 1) \tag{1}$$

Once the adversarial example crosses the non-adversarial border, the latest example is rejected and instead, a shift around the original image is made (Eq. 2). The shit is made around the l_∞ ball with the original image being at the center and the radius being the distance between the original image and the current adversarial example (Eq. 3).

$$\hat{x}^{new} = \Pi_{T(x,\hat{x}^{old})}(\hat{x}^{old} + \delta) \tag{2}$$

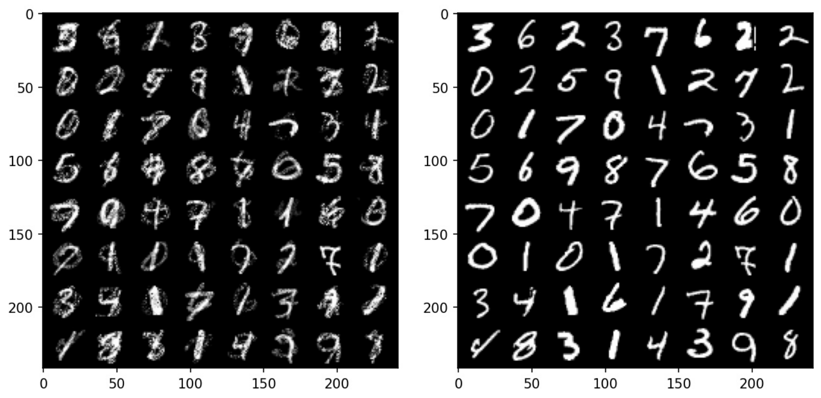
Figure 2: Boundary Attack - Adversarial Examples.

$$T(x, \hat{x}^{old}) = \{z \in \mathbb{R}^d : ||x - z|| \leq ||x - \hat{x}^{old}||\} \tag{3}$$

Both these steps are performed back and forth for a certain number of steps. Because of the inability of the adversarial example to move away from the the original image, it is possible and likely for the example to end in a local minimum if no random restart is performed. This is demonstrated in Figure 1.

5.2 Inverse Boundary Attack

Algorithm 2 Inverse Boundary Attack

Input x: original image, y: label, f: classifier, ρ: l_∞ radius, α: growth factor, $steps$: number of steps

Result $\hat{x}$: adversarial example

```
1: $\hat{x} \leftarrow x$
2: for steps do
3:     while True do                                        ▷ Move $\hat{x}$ away from $x$:
4:         $\delta \sim \mathcal{N}(0,1)$
5:         $\hat{x}^{new} \leftarrow \Pi_{\bar{T}(x,x+\rho)}(\hat{x}^{old} + \delta)$
6:         if $f(\hat{x}^{new}) \neq y$ then
7:             break
8:         else
9:             reject $\hat{x}^{new}$
10:        end if
11:    end while
12:    $\rho \leftarrow \rho + \alpha$                       ▷ Augment radius
13: end for
14: return $\hat{x}$
```

The Inverse Boundary Attack (IBA) algorithm 2 is similar to the BA in that both seek AE around the border of the non-adversarial distribution plane. However, instead of starting from a random image sampled from the dataset distribution and moving towards the original image, it instead aims to project a copy of the original image towards the border of a given l_∞ ball (Eq. 4) in an attempt to find itself in the adversarial distribution plane. The radius of the l_∞ ball is defined by a hyper-parameter ρ (Eq. 5) and is augmented by a factor of α if no AE are found (Eq. 6).

$$\hat{x}^{new} = \Pi_{\bar{T}(x,x+\rho)}(\hat{x}^{old} + \delta) \tag{4}$$

$$\bar{T}(x, x + \rho) = \{z \in \mathbb{R}^d : ||x - \rho|| \leq ||x - z||\} \tag{5}$$

$$\rho = \rho + \alpha \tag{6}$$

Because the dimension of the images is so large, it infeasible to test every instance on the l_∞ ball border, and so a certain amount are randomly tested to see if they land on the adversarial side. The

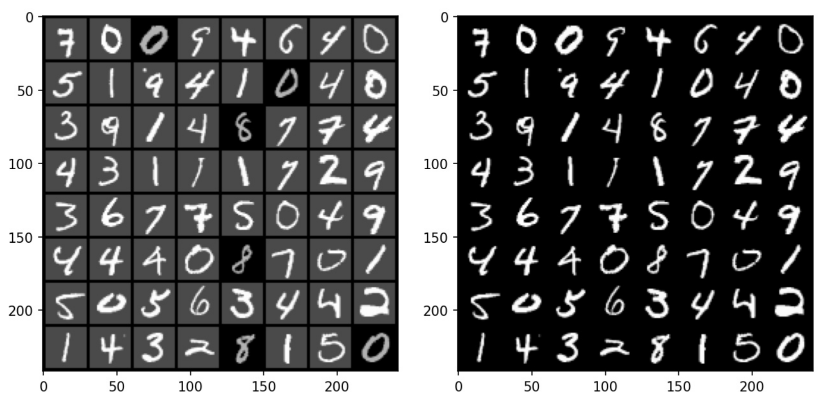

Figure 3: Inverse Boundary Attack - Adversarial Examples.

problem with this method is that, it is possible to find and example that is far from the border. That said, it is likely that the first adversarial example found will be close to the global minimum. This is demonstrated in Figure 1.

5.3 Combined Boundary Attack

Algorithm 3 Combined Boundary Attack

Input x: original image, y: label, f: classifier, α: move size, $steps$: number of steps
Result $\hat{x}$: adversarial example

```
1:  $\hat{x} \leftarrow IBA(x)$
2:  for steps do
3:      while True do                                      ▷ Move $\hat{x}$ towards $x$
4:          $\hat{x}^{new} \leftarrow \alpha x + (1-\alpha)\hat{x}^{old}, (0 \leq \alpha \leq 1)$
5:          if $f(\hat{x}^{new}) = y$ then
6:              break
7:          end if
8:      end while
9:
10:     while True do                                      ▷ Sample around $l_\infty$ ball
11:         $\delta \sim \mathcal{N}(0,1)$
12:         $\hat{x}^{new} \leftarrow \Pi_{T(x,\hat{x}^{old})}(\hat{x}^{old} + \delta)$
13:         if $f(\hat{x}^{new}) \neq y$ then
14:             break
15:         else
16:             reject $\hat{x}^{new}$
17:         end if
18:     end while
19: end for
20: return $\hat{x}$
```

Because the BA searches only by moving towards the local minimum, it is justifiable to define it as an exploitative search algorithm [6]. On the other hand, the IBA randomly chooses examples from an increasing l_∞ ball border. This randomness allows the algorithm to search many areas of the plane without being constrained to a certain distance from the original image. Because of this, it is justifiable to define the IBA as an explorative search algorithm [6]. The goal of the Combined Boundary Attack (CBA) algorithm 3 is to combine the explorative nature of the IBA with the exploitative nature of the BA. The intuition is that by using the IBA, we are able to find an adversarial example that is close to the global minimum. Then, by using the BA, we move closer and closer towards the original image until the distance reaches a minimum. The concept is shown in the Figure 1.

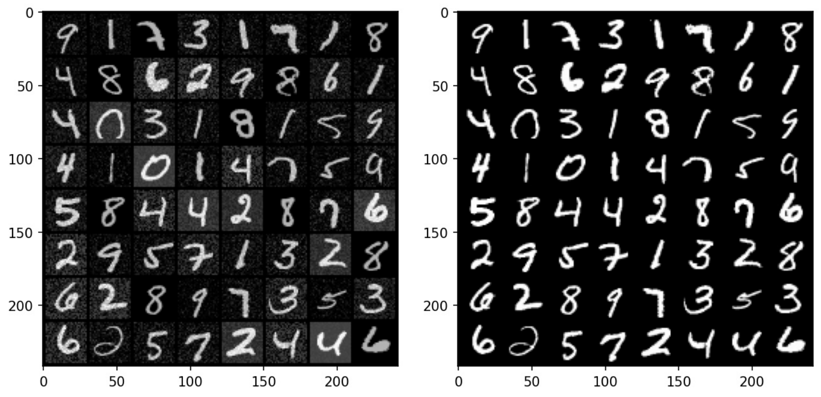

Figure 4: Combined Boundary Attack - Adversarial Examples.

Table 1: Results

	ViT			CNN		
Arch & Example	BA	IBA	CBA	BA	IBA	CBA
Model & Real (↑)	100.0	98.4	100.0	96.8	98.4	95.3
Model & Attack (↓)	0.0	15.6	10.9	0.0	10.9	6.2
Difference (↑)	**100.0**	**82.8**	**89.1**	**96.8**	**87.5**	**89.1**
Oracle & Real (↑)	96.8	100.0	96.8	100.0	98.4	95.3
Oracle & Attack (↑)	67.1	28.1	82.8	70.3	26.5	59.3
Difference (↓)	**29.7**	**71.9**	**14.0**	**29.7**	**71.9**	**36.0**
Brother & Real (↑)	92.1	87.5	90.6	98.4	100.0	98.4
Brother & Attack (↑)	56.2	42.1	76.5	56.2	23.4	65.6
Difference (↓)	**35.9**	**45.4**	**14.1**	**42.2**	**76.6**	**32.8**

6 Analysis

We now examine the various attack algorithms and their effectiveness across various network architectures (Table 1). First, we start by comparing the ViT (model) and its CNN oracle. We also test the AE generated by the model on another ViT architecture, which we call the brother. The brother network in this case is a ViT with twice the input patch count, making it more complex. The algorithm that reduces the accuracy over all the networks the most is the IBA. What is interesting here is that the brother network is less affected by the attacks than the oracle. Overall, the images generated with this attack are able to fool many different models. This is likely due to the AE being further from the non-adversarial border when compared to those generated with the other algorithms. Some AE generated with this algorithm are seen in Figure 3. These images are not noisy and look like the original images. That said, it is clear that the images have been tampered with because most are in a very different shade than the original. The BA algorithm is the most effective against the model. This is because the AE never crosses into the non-adversarial distribution. Unlike the Inverse Bounding Attack, the brother network does not outperform the the oracle. Some AE generated with this algorithm are seen in Figure 2. These images look noisy, and sometimes they do not look like the original image. This is likely due to the algorithm finding a local minimum that is not very close to the original image. The CBA results are very interesting because, although it combines both the previous methods, its generated AE do not affect the brother or oracle as much as the other method when used alone. Also, the generated AE are more effective against the model than those generated with the IBA. Some AE generated with this algorithm are seen in Figure 4. The images look very similar to the original images. They are not as noisy as those generated by the BA algorithm, and they are not a completely different shade like those generated by the IBA algorithm. This is likely due to minimizing the distance between the the original image and adversarial example after performing the IBA.

Next, we examine the different attack algorithms when a CNN (50 layers) is used as the model, a ViT as the oracle, and a CNN (101 layers) as the brother. The IBA algorithm, once again, reduces accuracy across all networks the most. The brother in this case is affected a lot more than previously and loses an amount of accuracy comparable to that of the oracle. Some AE generated with this algorithm are seen in Figure 6. The BA algorithm is again the most effective against the model. The oracle and brother both see somewhat substantial drops in accuracy. Some AE generated with this algorithm are seen in Figure 5. The CBA algorithm is the second most effective against the model. Unlike the previous analysis, the loss in accuracy of the oracle and brother is somewhat substantial. Some AE generated with this algorithm are seen in Figure 7. In every case, the images appear to share the same qualities as those generated when the model was a ViT.

7 Conclusion

Local search algorithms are able to produce AE with almost imperceptible noise by performing explorative and exploitative searches. They are able to generate AE that can fool a model without necessarily fooling other networks of different architectures. From our results, we can infer that not all models share the same distribution plane and, therefore, won't always share the same AE.

8 References

[1] Anish Athalye, Nicholas Carlini, and David Wagner. “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples”. In: *International conference on machine learning*. PMLR. 2018, pp. 274–283.

[2] Wieland Brendel, Jonas Rauber, and Matthias Bethge. “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models”. In: *arXiv preprint arXiv:1712.04248* (2017).

[3] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[4] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572* (2014).

[5] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[6] S Luke. “Essentials of Metaheuristics, vol. 2”. In: *Lulu Raleigh* (2013).

[7] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

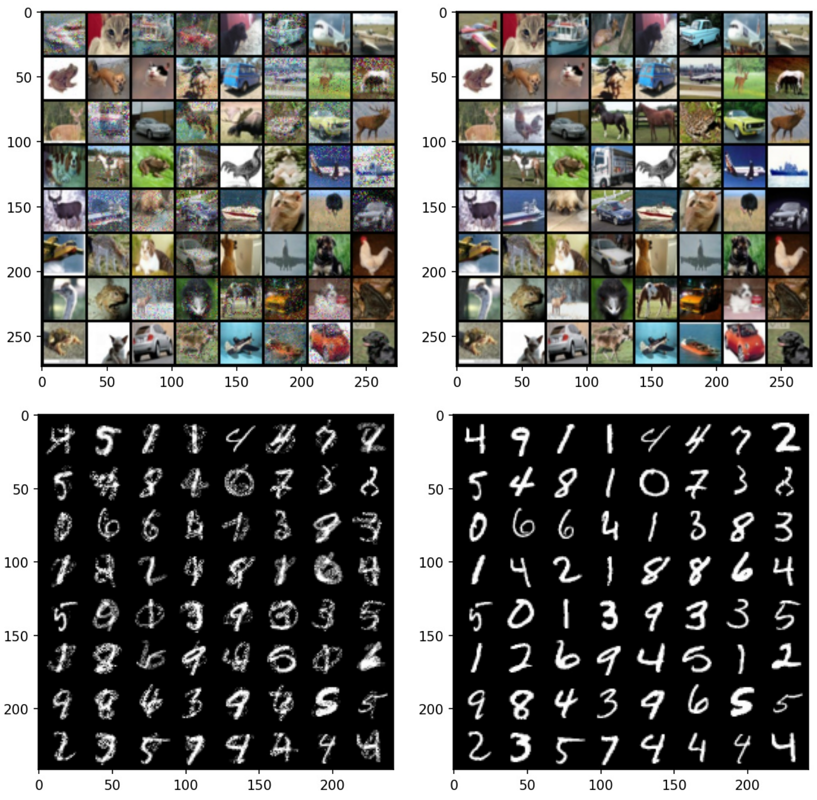

Figure 5: Boundary Attack - Adversarial Examples (CNN).

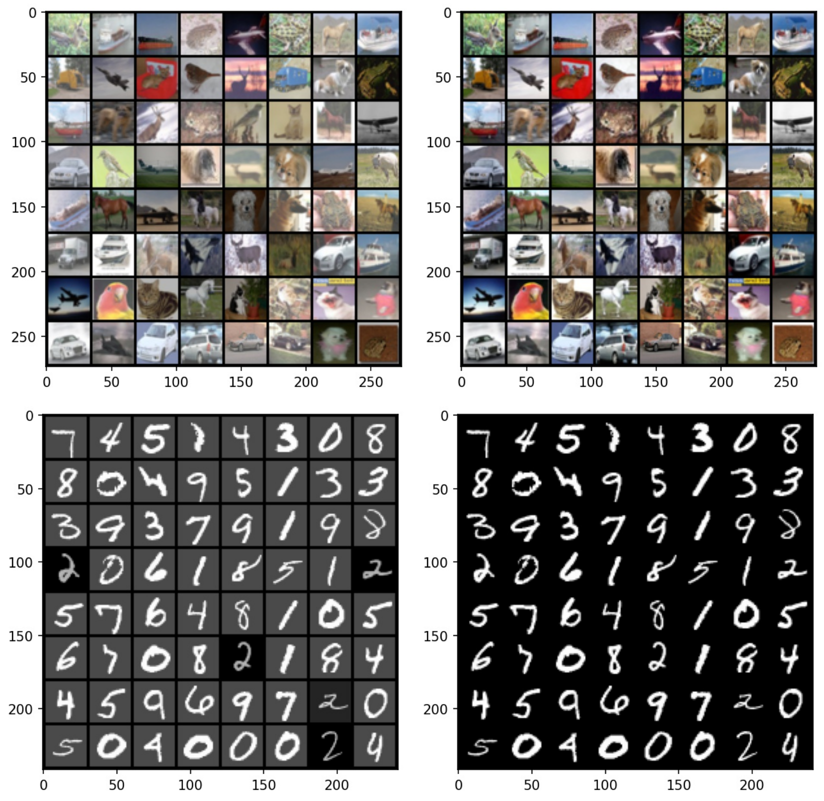

Figure 6: Inverse Boundary Attack - Adversarial Examples (CNN).

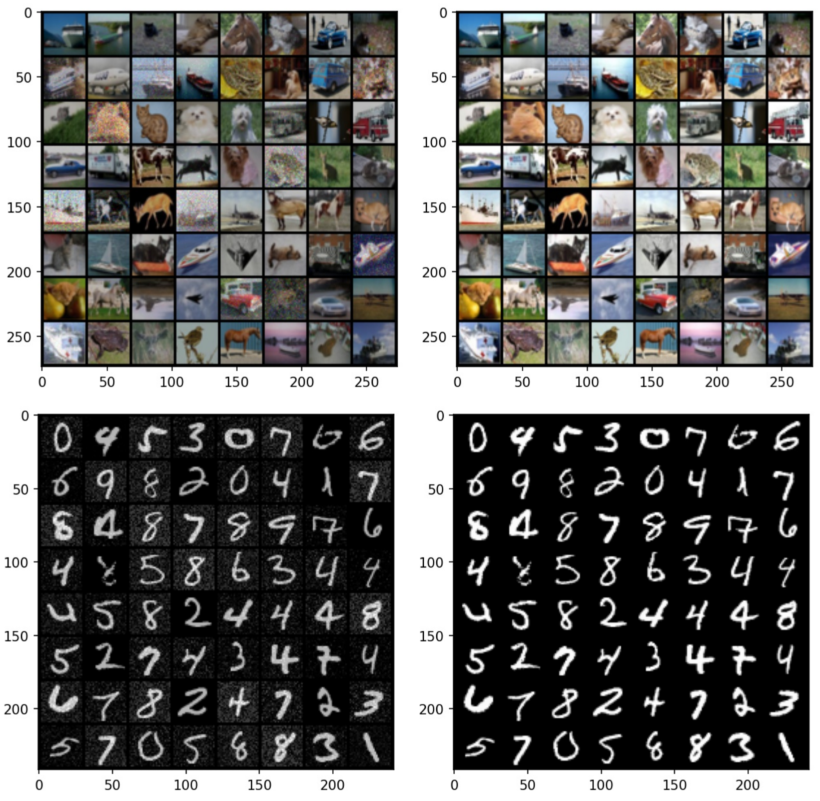

Figure 7: Combined Boundary Attack - Adversarial Examples (CNN).