Fool me once, shame on - shame on you. Fool me - can't get fooled again.

Gabriel C-Parent
Département d'informatique et recherche opérationnelle
Université de Montréal
gabriel.c-parent@umontreal.ca

Dora Fugère
Département de mathématiques et de statistique
Université de Montréal
dora.fugere@umontreal.ca

Abstract

Adversarial examples generation from input space in neural network has shown that these powerful constructs can be manipulated into misclassifying previously well classified examples by adding an imperceptible amount of distortion.

Using this methodology, we investigate the relative robustness to distortion of a simple linear SVM with hinge loss and some regularization schemes.

1 Introduction

Recently, neural networks have been brought under questioning. The smoothness assumption, the idea that imperceptible distortion of input shouldn't change the output was shown not to hold [**?**]. This is a remarkable finding since smoothness was assumed to be a necessary property of the learning process. This comes in stark constrast with feats such as automatic image description [**?**] and large-scale multi-character text recognition [**?**] to name but a few.

As for most real-world problems, there are many desirable and often conflicting goals when using machine learning. Amongst them speed, accuracy and simplicity are easy to justify. We'll focus on comprehensibility, because that justifies us using a simpler model. We interpret simplicity as "given two models with the same generalization error, the more comprehensible one should be preferred" [**?**]. This obviously is dependent on multiple other factors (e.g. speed and accuracy) but it does sound like the *keep it simple stupid* rule of thumb. Furthermore, as stated in [**?**], empirical comparison of performance is very context-dependent and can be infludenced by treatments such as the preprocessing steps, training parameters and model hyperparameters.

Inspired by the methodology to induce misclassification, we wondered if a similar optimization procedure could be applied to generate adversarial examples in a simpler classifier. For this purpose, we used a support vector machine (SVM) with hinge loss and L_1, L_2 and $elasticnet$-regularization. The only other contender would have been Naive Bayes, but we happen to like sklearn's implementation of the linear SVM [1] [**?**]. We report the robustness of our optimization procedure, the results on classifiers trained with different loss and regularization parameters and we then feed the generated adversarial examples to a neural network to see if some underlying feature of the image was captured.

[1] we wouldn't risk reinventing the square wheel

2 Framework

2.1 Dataset

The experiments were performed on the MNIST dataset [**?**].

Let $X = \{0, 255\}^{784}$, the input domain. This is the set of 28×28 8-bit grayscale image.

Let $Y = \{0, 9\}$, the output domain. This is the set of valid classes for an MNIST digit.

2.2 Preprocessing

The MNIST dataset was deskewed and brought back to 8-bit data. This improved the performance of the classifier.

2.3 Optimization goal

Let $f : X \to Y$ a classifier mapping $x_i \in X$ to $y_i \in Y$.

We aim to solve the following optimization

$$\begin{aligned} &\text{minimize} && \|r\|^2 \\ &\text{subject to} && \\ & && x_i + r \in X \\ & && f(x_i) \neq f(x_i + r) \end{aligned} \tag{1}$$

This is quite similar to [**?**] but the newly generated images remain 8-bit to stay in the input domain of the MNIST dataset. Sadly, this also makes it a discrete optimization problem.

2.4 Optimization goal for the linear SVM

We denote $\widehat{y_i}$ the correct class for x_i. Given a correctly classified x_i, the objective is to find a vector of distortion r such that $f(x_i + r) \neq \widehat{y_i}$.

In a two-class setting, the classifier classifies the input based on the following decision function.

$$y_i = argmax(x_i \cdot W^T + b) \tag{2}$$

Suppose that $\widehat{y_i} = 1$, the correct class is 1.The difference between the class weights (coefficients) of the classifier is W_d. It is obtained by subtracting the coefficients of the goal class (2 in this case) by $\widehat{y_i}$'s.

$$W_d = W_2 - W_1 \tag{3}$$

The distance between the values of the two classes is d.

$$d = x_i \cdot W_d^T \tag{4}$$

To cause misclassification, r must respect the following constraint:

$$r \cdot W_d^T > d \tag{5}$$

What we need is to find the smallest $\|r\|^2$ that will cause misclassification.

Note that when there are more than two classes, we just apply the procedure to all other classes $y \neq \widehat{y_i}$ and choose the one with minimal squared euclidean norm.

The distorted set of images is generated by applying the following function over the input images.

$$distorted(x_i) = min(\{\|r_y\|^2 \mid y \neq \widehat{y_i}\}) \tag{6}$$

where $\widehat{y_i}$ is the true and predicted class of the ith image and $\|r_y\|^2$ the smallest squared euclidean norm needed for the classifier to classify it as y.

2.5 Knapsack problem and the greedy approach

The problem is similar to the bounded multiple-class binary Knapsack problem [**?**], with the difference that we are searching for the smallest knapsack holding a value superior to d (equation 4).

The exact algorithm would be too costly for our purpose so we chose to use a greedy heuristic inspired by Dantzig's [**?**].

The procedure is described in more details in section 4.1.

3 Experimental results

3.1 Precision of optimization procedure

Although the bounds on the optimization procedure could be arbitrary big, it is usually small for this problem, because d is usually quite big relative to the size of weight increments. This means tight bounds on the possible true value of the squared euclidean norm.

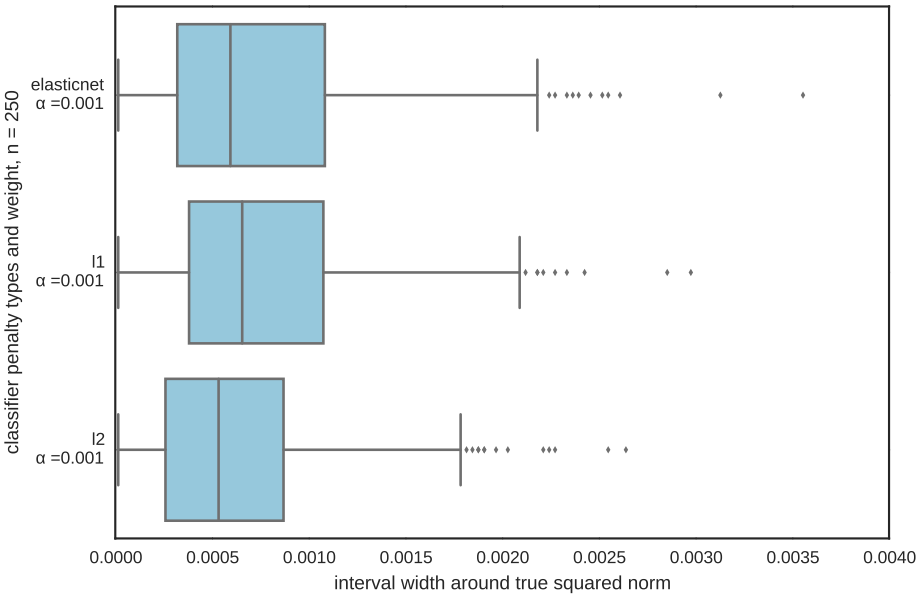

Figure 1: Size of the interval on the true value of the minimal squared norm for various classifier on the MNIST dataset. The bound is shown to be very tight, especially considering the usual size of d.

Since the bound on the true optimal value is always very tight, the use of the greedy optimization procedure is reasonable.

3.2 Regularization schemes

We wanted to observe the effects of different regularization methods and parameters on the examples. To do so, we started with a pilot run, to assess the potential of the various configurations.

The most promising regularization penalty was the L_2-norm (ridge regression) penalty and its results are shown here. The complete results can be viewed in figure 4.3 of the supplementary material.

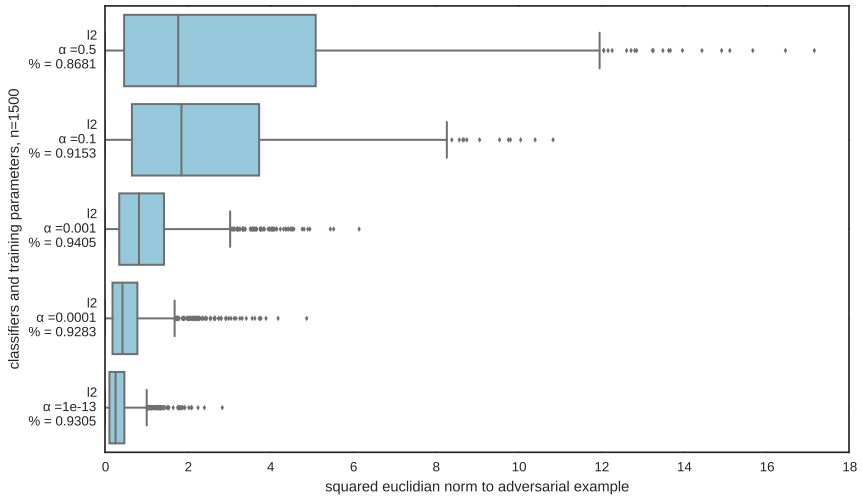

Figure 2: Improvement of robustness with regularization for the L_2-regularized SVM with hinge loss. The improvement in squared euclidean norm is very apparent as the value of α goes up.

4 Discussion

As expected, higher regularization penalty yields better robustness to distortion. This comes with the cost of decreasing generalization accuracy when the strength is too high. This got the L^1-regularized classifiers to fail early, as their accuracy decreased below the fixed threshold we chose (85% of generalization accuracy) very quickly, without much impact on robustness. This is show in figure 4.3.

From the observations in figure 3.2, we chose the L^2-regularized classifier with regularization term 0.5 for further experiments because it had the highest robustness to distortion.

We went on to explore a few aspects of the distortion. The first interesting property for our classifier was the non-uniformity of distribution of adversarial examples as shown in figure **??**. It is quite interesting to see the relation between original and closest class by distortion.

It is furthermore interesting to observe that some input classes are more robust (especially the ones, but it makes sense since the MNIST dataset was preprocessed to straighten them). The distribution of squared euclidean norms for adversarial examples is presented in figure 4.5. To further the point, selected examples of digits and their adversarial class counterpart we selected and are shown in figure 4.6. The smallest, median and largest required distortions were selected to give an idea of the range of distortion and the visual impact it has on the input images when using the optimization procedure. It must be remembered that the distortion can consist of adding or removing intensity by 1 bit increments and the resulting adversarial examples are valid 8-bit grayscale images.

Some additional work was done on cross-training-set generalization between two classifiers (L^2-regularized, $\alpha = 0.5$, same as before). The 60000 MNIST dataset was split in two disjoint sets, as in [**?**].

Acknowledgments

We would like to thank Red Bull, it gives you wings[2].

[2]No it doesn't. Don't sue them again.

Supplementary Materials

4.1 greedy optimization heuristic, the nitty–gritty details

Let $S = (o_1, ..., o_n)$ be objects sorted in decreasing unit-value order i.e. value(o_i)/volume(o_i) $\geq$ value(o_{i+1})/volume(o_{i+1}).

If K is a choice of the first k items of S and $V = volume(K)$, then a knapsack of size V would be optimal for the objects of S. This is intuitive since no other choice would have a better value/cost ratio. This is the crux of the matter for our optimization procedure.

Let (K_1, V_1) and (K_2, V_2), the volumes and values of two optimal knapsacks. We know that if $K_1 < K_2$ then $V_1 \leq V_2$, for we could choose the elements of K_1 to get an equal value or find better. This also means that given $V_1 < V_3 < V_2$ then $K_1 < K_3 < K_2$. This allows us to find bounds on K_3 if we know (K_1, V_1) and (K_2, V_2), as it must be squeezed between the two.

4.2 regularization methods and robustness

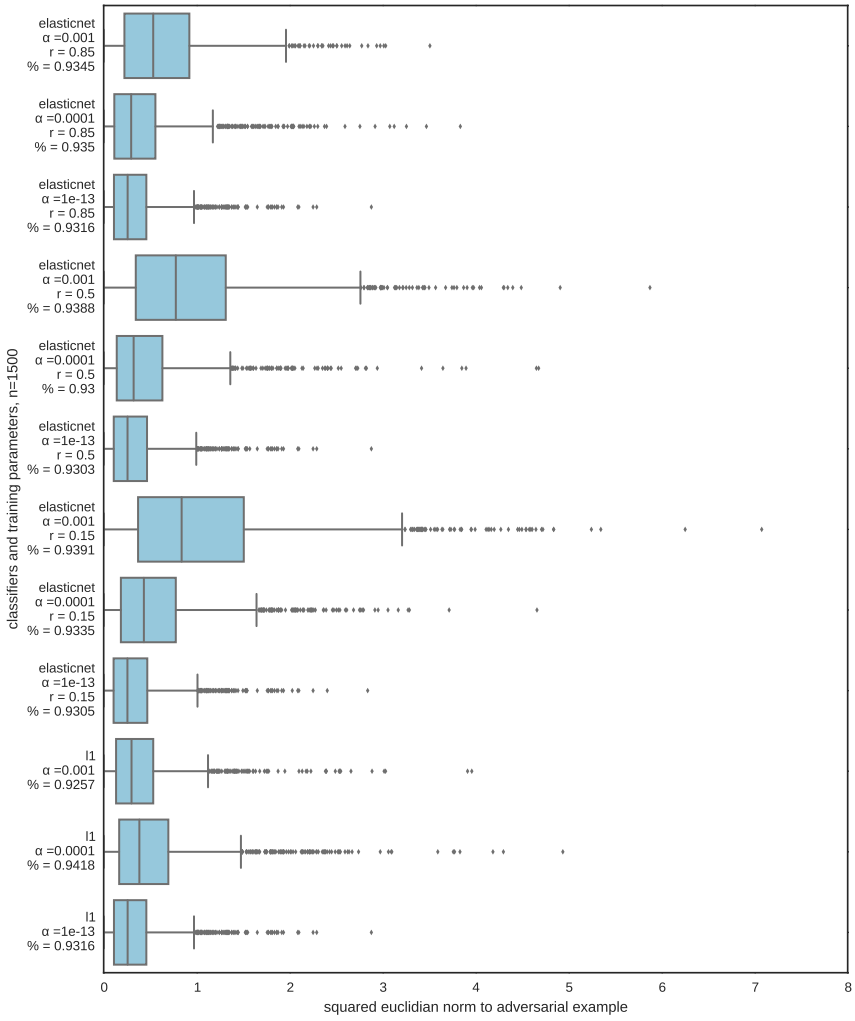

Figure 3: Less successful classifiers. Squared euclidian norm is relatively small in comparison to L_2 penalty. We see improvement as the coefficient of the penalty grows bigger.

4.3 comparison of robustness of two classifiers

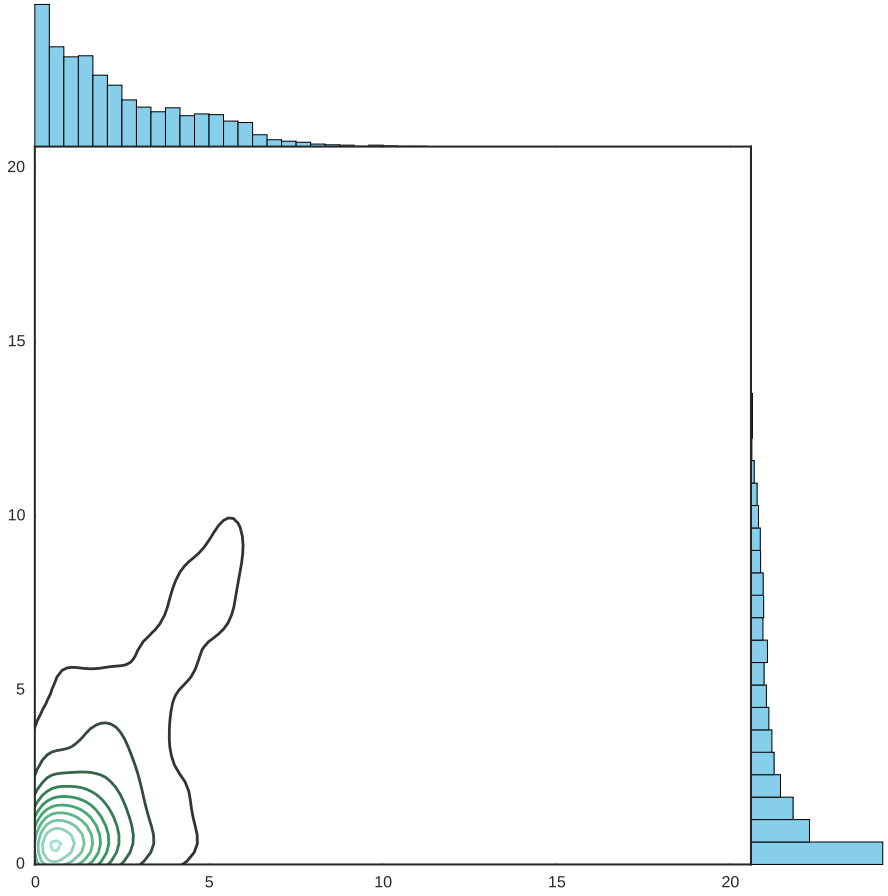

Figure 4: Comparison of the distribution of the squared norm for distorted images. The last same two classifiers as in figure 3.2 were compared. The classifier with bigger α is yielding bigger squared euclidean norm (the angle of the density allows us to tell). This means that the classifier is more robust to distortion.

4.4 example of transitions

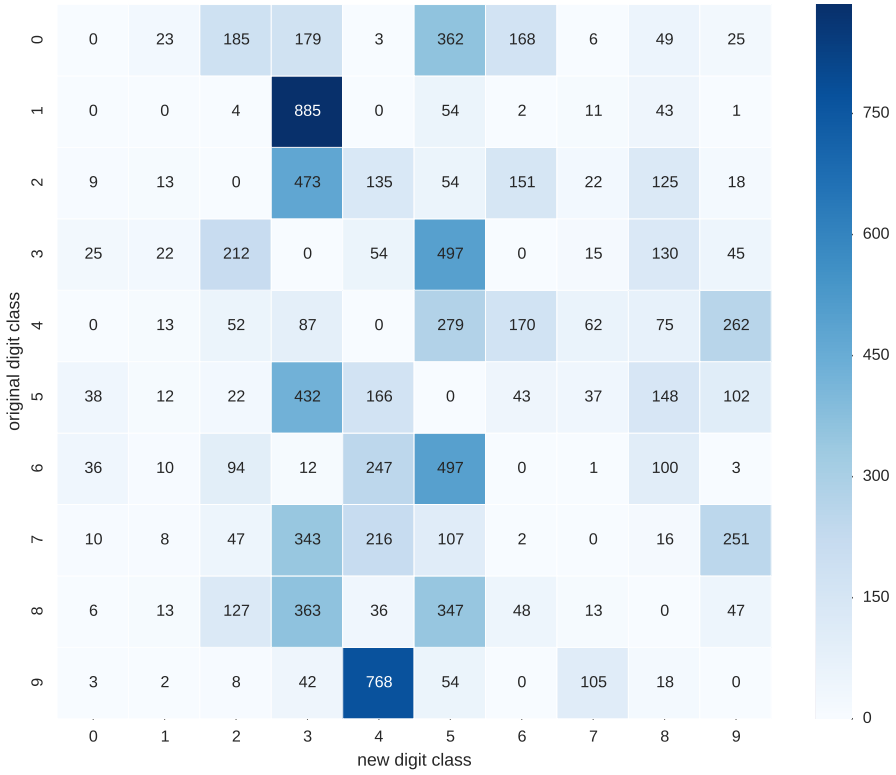

Figure 5: Classes of the adversarial examples generated. For each original class, 1000 images were chosen (rows sum to 1000). We can see quite a strong bias to some classes, especially for class 1 and 9.

4.5 class-based distribution of squared euclidean norm

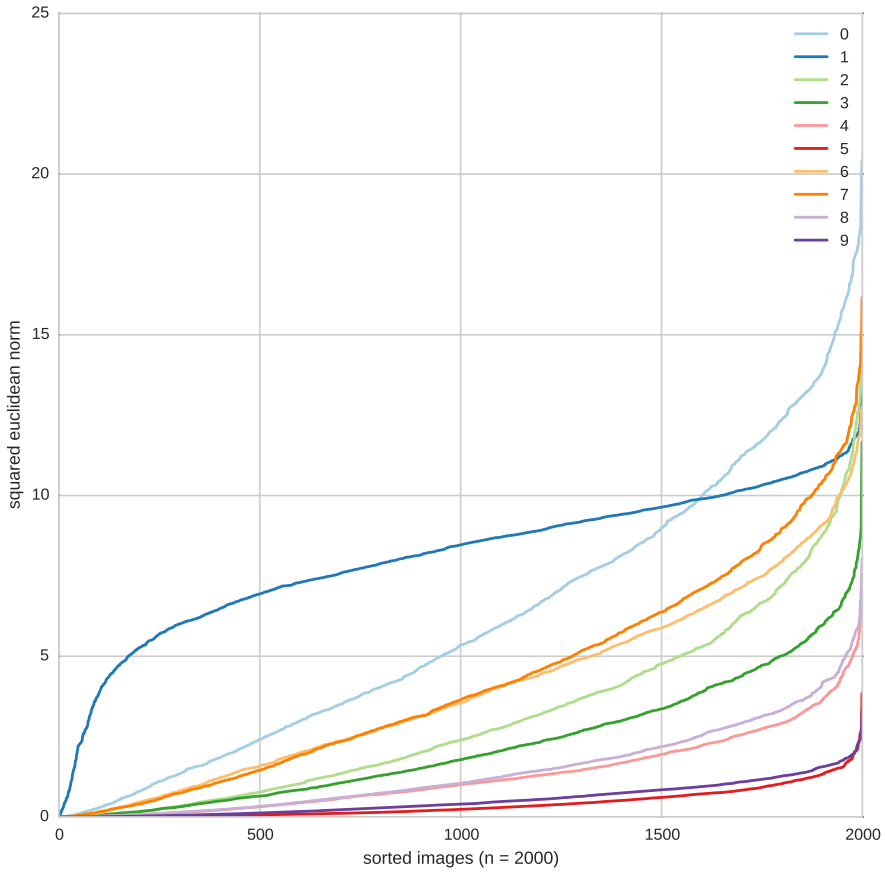

Figure 6: Comparison of the distribution of the squared norm for distorted images for each class of the $\alpha = 0.5$ L^2-regularized. Some classes are much easier to generate adversarial examples from.

4.6 example of input and adversarial images

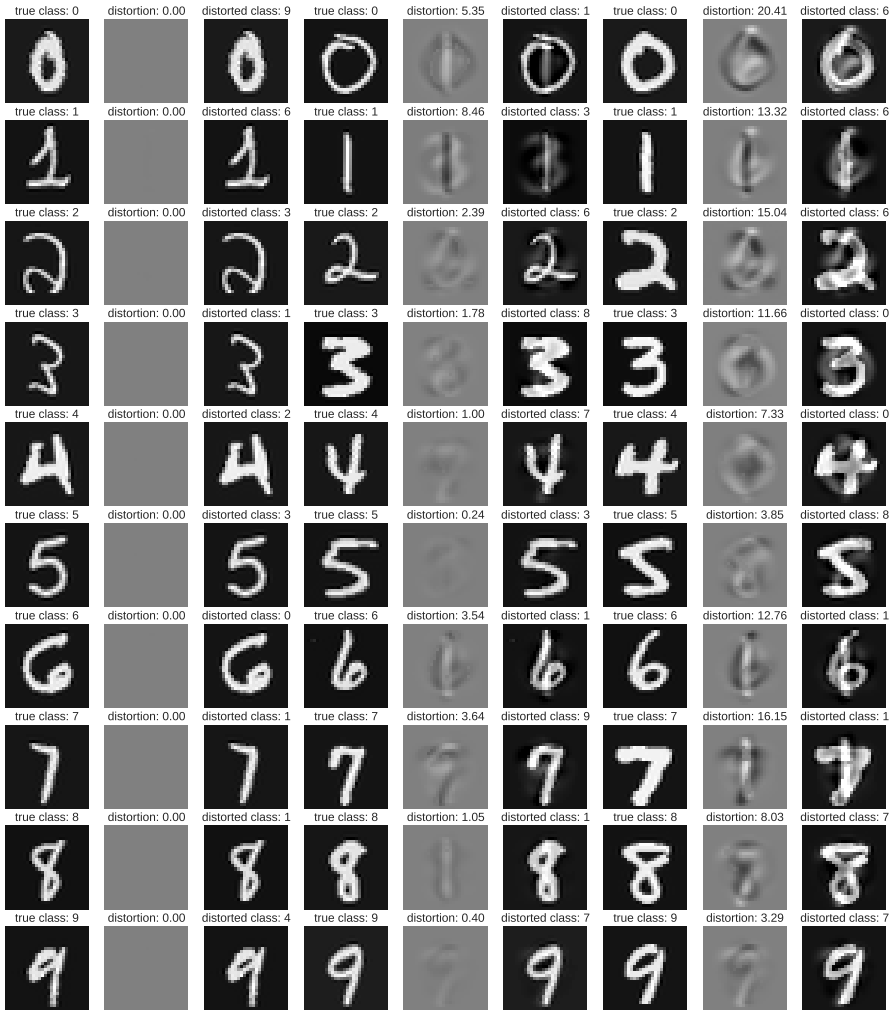

Figure 7: For each digit class, adversarial examples were generated. On each row, the 0, 50 and 100th percentiles of squared euclidean norms were chosen. The images are going from small to big from left to right. As can be observed, the smallest distortion is imperceptible but on the other hand, the median (50th percentile) is generally subtle. This is a good sign, meaning that the classifier is pretty robust.