# 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 robustness of a simple classifier to similar distortion.

To make this feasible on our limited computing power, we devised a simple greedy optimization procedure to generate adversarial examples efficiently. We then observed some interesting properties of the perturbations from the standpoint of input class accuracy. Finally, we confirmed that the perturbations affecting a classifier can affect another one, albeit in this case most perturbations are not all that imperceptible.

Further work is necessary to explain the observations, but for now they may be summed up by "well, that's funny".

## 1 Introduction

Recently, neural networks have been brought under questionning. The smoothness assumption, the idea that imperceptible distortion of input shouldn't change the output was shown not to hold [1]. 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 [2] and large-scale multi-character text recognition [3] 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" [4]. 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 [5], 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

(one against all for each class). The only other contender would have been Naive Bayes, but we happen to like sklearn's implementation of the linear SVM [1] [6]. 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.

## 2 Framework

### 2.1 Dataset

The experiments were performed on the MNIST dataset [7].

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

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

This is quite similar to [1] 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.

We will sometimes refer to $\|r\|$, the square euclidean norm as the distance. It must be stated that it is not a metric, as it does not satisfy the triangle inequality.

### 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 $d$.

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

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

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 distance.

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 $i$th image and $\|r_y\|^2$ the smallest distance 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 [8], 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 [9].

The procedure is described details in section 4.1, along with an experimental validation.

## 3 Experimental results

### 3.1 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 $\alpha$ term represents the weight on the regularization penalty (higher values mean harsher penalty).

The most promising regularization penalty was the $L_2$-norm (ridge regression) penalty and the distribution of are shown here. The complete results can be viewed in figure 4.2 of the supplementary material.

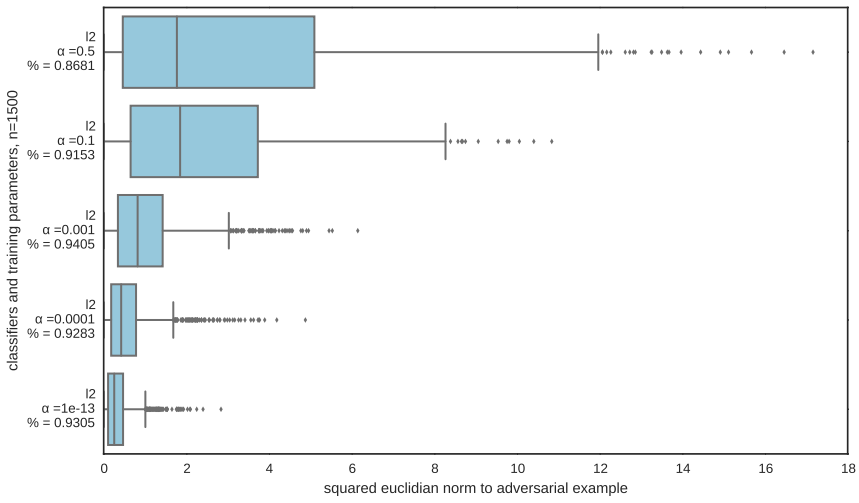


Figure 1: Improvement of robustness with regularization for the $L_2$-regularized SVM with hinge loss. The improvement in distance is very apparent as the value of $\alpha$ goes up.

### 3.2 class-based distribution of distances

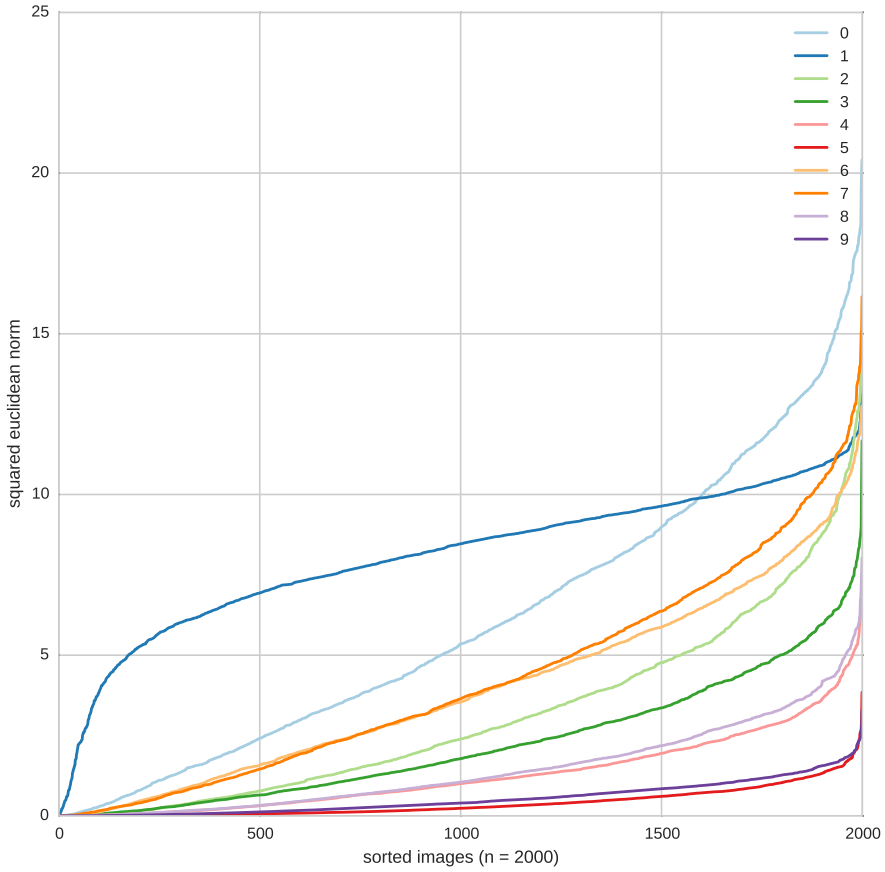


Figure 2: Comparison of the distribution of the squared norm for distorted images for each class of the $\alpha = 0.5$ $L_2$-regularized. Some classes are much easier to generate adversarial examples from.

The distribution of distances for adversarial examples is quite variable and some classes are very different for each class, and it makes sense since some of them are quite similar at the pixel level. The relation between input class and closest adversarial class was also investigated and the results are shown in figure 4.4.

### 3.3 cross-training generalization

By training two classifiers on disjoint sets of the MNIST, generating adversarial examples for each (on which they have zero accuracy) and feeding them to the other classifier, we obtain this class-based accuracy (normalized on rows). Obviously, only examples that are well classified by both classifiers are pertinent to use so that limits the possible inputs. It is quite interesting to observe that they complement each other on the diagonal in that if one has a high value, the other doesn't. We are not quite sure why that is.

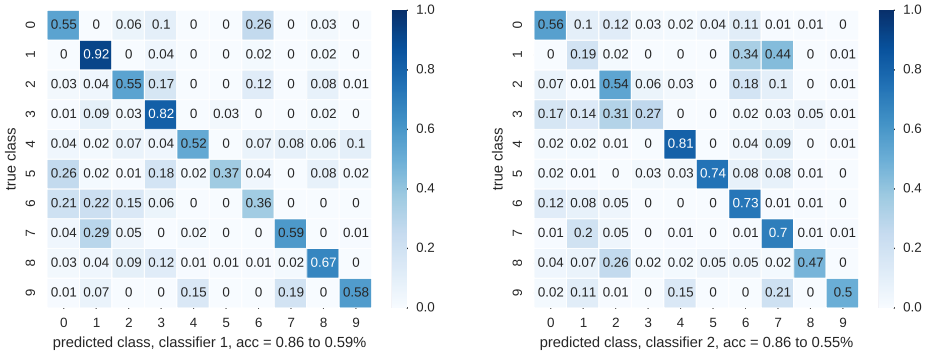


Figure 3: Class-based accuracy of cross-training generalization for two similar classifiers ($L_2$-regularized, $alpha = 0.5$) trained on disjoint sets.

## 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. The $L_2$-regularized classifiers performed quite well and have a good robustness to distortion as show in figure 3.1. We chose this mode of regularization with $\alpha = 0.5$ for further experiments because it had the highest robustness to distortion. The $L_1$-regularized classifiers (including the $elasticnet$) failed early, as their accuracy decreased below the fixed threshold we chose (85% of generalization accuracy) without much impact on robustness. This is shown in figure 4.2.

We went on 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 distances for adversarial examples is presented in figure 3.2. To get a better intuition, selected examples of digits and their adversarial class counterpart are shown in figure 4.3. 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. We believe the squared euclidean norm correlates well with visual impact.

Finally, an experiment 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 [1]. The results are shown in figure 3.3. The accuracy decreased by about 30%, for both classifiers, but the interesting thing is that there is strong class bias, i.e. when one classifier correctly classified adversarial examples of its counterpart, the other one usually failed to do the same.

In conclusion, we are quite pleased with the robustness and overall performance of the classifier. Its simplicity allowed us to devise a simple procedure to generate adversarial examples and in general, it held up quite well against distortion.

### Acknowledgments

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

## References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. URL `http://arxiv.org/abs/1312.6199`.

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

[2] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014. URL http://arxiv.org/abs/1411.4555.

[3] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Sacha Vinay. Multi-digit number recognition from street view. 2013.

[4] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data mining and knowledge discovery*, 3(4):409–425, 1999. URL http://link.springer.com/article/10.1023/A:1009868929893.

[5] David J. Hand. Classifier technology and the illusion of progress. *Statistical Science*, 21(1): 1–14, February 2006. ISSN 0883-4237. doi: 10.1214/088342306000000060. URL http://projecteuclid.org/Dienst/getRecord?id=euclid.ss/1149600839/.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] Yann Lecun and Corinna Cortes. The mnist database of handwritten digits. 1998.

[8] Francois Vanderbeck. Extending dantzig's bound to the bounded multiple-class binary knapsack problem. *Mathematical Programming*, 94(1):125–136, December 2002. ISSN 0025-5610, 1436-4646. doi: 10.1007/s10107-002-0300-7. URL http://link.springer.com/10.1007/s10107-002-0300-7.

[9] George B Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2), 1957. doi: 10.1287/opre.5.2.266.

# 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.

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

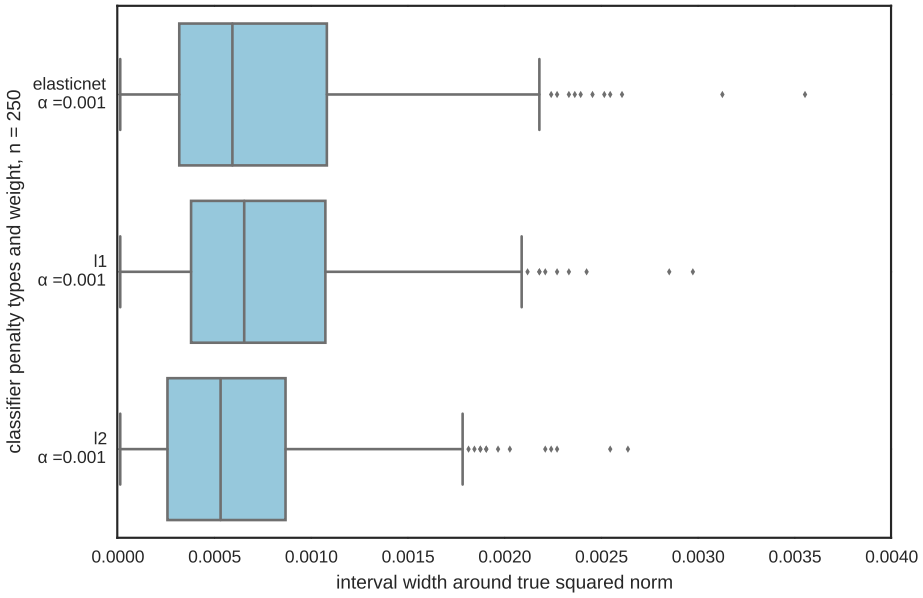


Figure 4: Size of the interval on the true value of the minimal squared norm for various classifier on the MNIST dataset. The bound is shown to be very tight, especially considering the usual size of $d$.

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

### 4.2 regularization methods and robustness

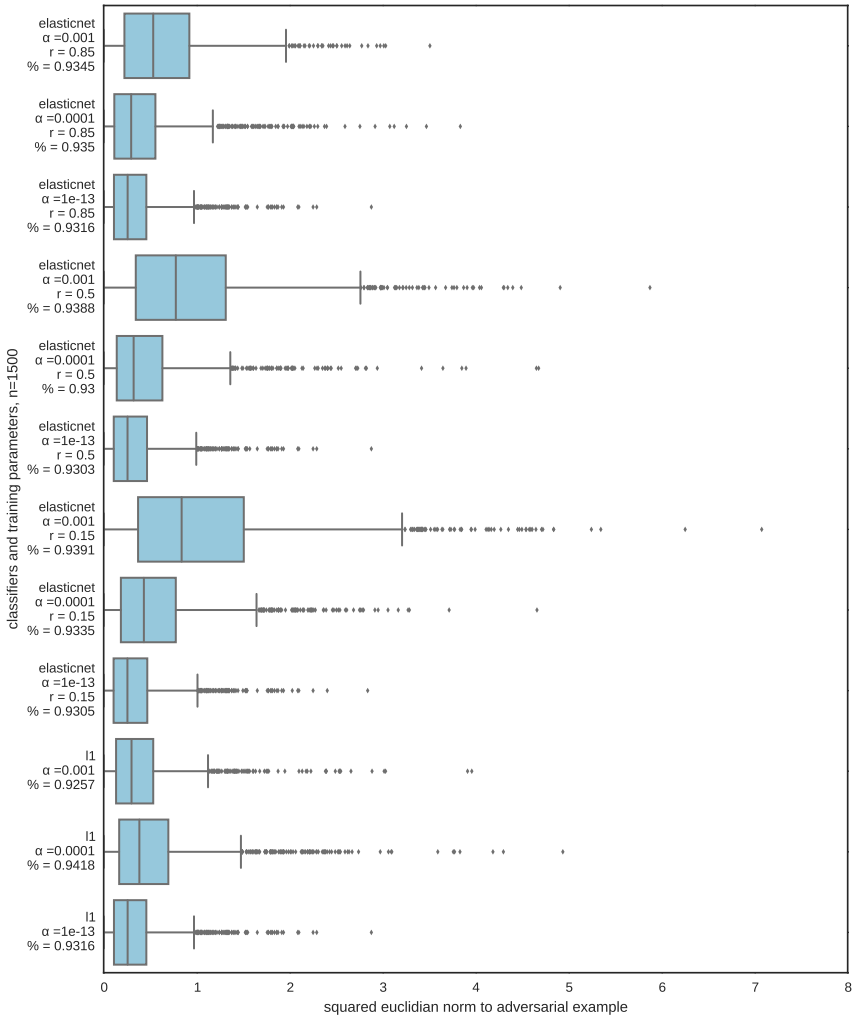


Figure 5: Less successful classifiers. Squared euclidian norm is relatively small in comparison to $L_2$ penalty. We see improvement as the coefficient of the penalty grows bigger.

The observation of these values prompted us to choose $L_2$-regularization for further experiments.

### 4.3 example of input and adversarial images

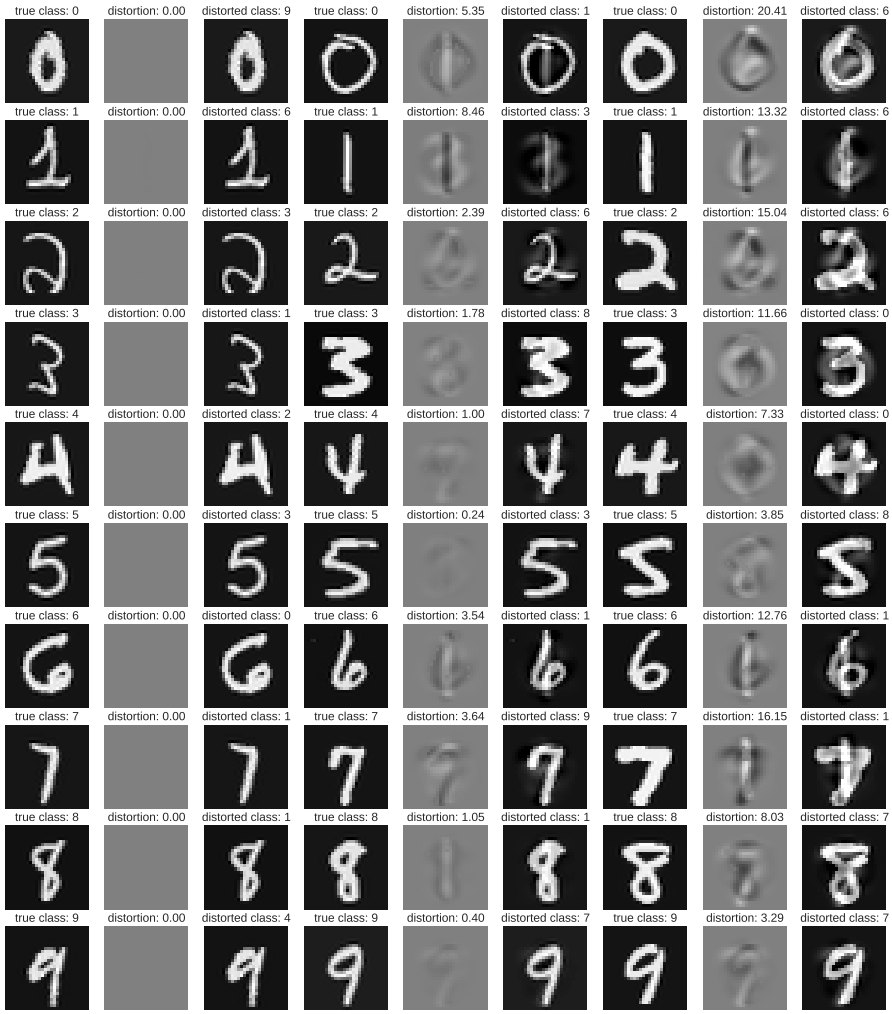


Figure 6: For each digit class, adversarial examples were generated. On each row, the 0, 50 and 100th percentiles of distances were chosen. The images are going from small to big from left to right. As can be observed, the smallest distortion is imperceptible but on the other hand, the median (50th percentile) is generally subtle. This is a good sign that the classifier is pretty robust to distortion.

### 4.4 relationship between input class and closest adversarial class

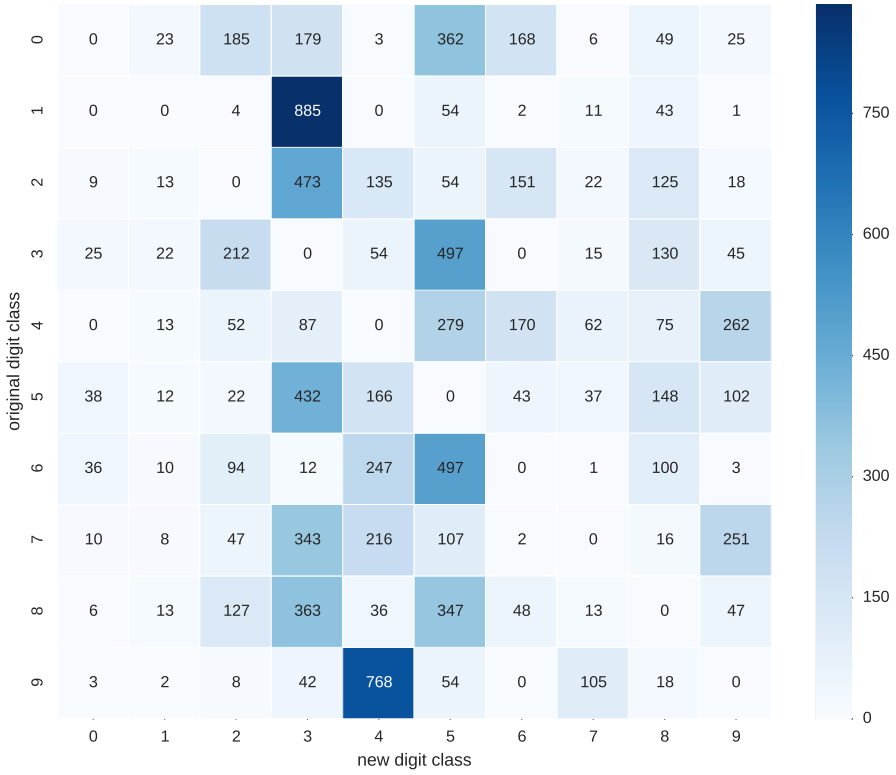


Figure 7: Classes of the adversarial examples generated. For each original class, 1000 images were chosen (rows sum to 1000). We can see quite a strong bias to some classes, especially for class 1 and 9.