A. Projection onto the intersection of the l_0-ball and the σ-map constraints

We here present the projection step which is used for the σ-PGD attack, where we allow perturbations on at most k pixels and respecting the σ-map constraints which are defined in (4) and (5).

A.1. Color images

Given a color image $x \in [0,1]^{d\times 3}$, we want to project a given point $y \in \mathbb{R}^{d\times 3}$ onto the set

$$C(x) = \big\{z \in \mathbb{R}^{d\times 3} \,\big|\, \sum_{i=1}^{d} \max_{j=1,\dots,3} \mathbb{1}_{|z_{ij}-x_{ij}|>0} \leq k, \\ (1-\kappa\sigma_{ij})x_{ij} \leq z_{ij} \leq (1+\kappa\sigma_{ij})x_{ij}, \\ 0 \leq z_{ij} \leq 1\big\},$$

where d is the number of pixels, σ_{ij} are the pixelwise, channel-specific bounds defined in Section 2 and $\kappa > 0$ a given parameter. We can write the projection problem as

$$\begin{aligned} \min_{\lambda\in\mathbb{R}^d} \quad & \sum_{i=1}^{d}\sum_{j=1}^{3}(y_{ij} - (1+\lambda_i\sigma_{ij})x_{ij})^2 \\ \text{s. th.} \quad & -\kappa \leq \lambda_i \leq \kappa, \quad i = 1,\dots,d \\ & 0 \leq (1+\lambda_i\sigma_{ij})x_{ij} \leq 1,\ i = 1,\dots,d,\ j = 1,\dots,3 \\ & \sum_{i=1}^{d} \mathbb{1}_{|\lambda_i|>0} \leq k \end{aligned}$$

Ignoring the combinatorial constraint, we first solve for each pixel the problem

$$\begin{aligned} \min_{\lambda_i\in\mathbb{R}} \quad & \sum_{j=1}^{3}(y_{ij} - (1+\lambda_i\sigma_{ij})x_{ij})^2 \\ \text{s. th.} \quad & -\kappa \leq \lambda_i \leq \kappa \\ & 0 \leq (1+\lambda_i\sigma_{ij})x_{ij} \leq 1, \quad j = 1,\dots,3 \end{aligned}$$

We first note that the last constraint is always fulfilled if $x_{ij} = 0$ or $\sigma_{ij} = 0$. In the other case we can rewrite the constraint as

$$-\frac{1}{\sigma_{ij}} \leq \lambda_i \leq \frac{1}{\sigma_{ij}}\Big(\frac{1}{x_{ij}} - 1\Big), \quad j = 1,\dots,3.$$

Combining all constraints yields

$$\lambda_i^{(l)} := \max\Big\{-\kappa, \max_{\substack{j \\ x_{ij}\neq 0, \sigma_{ij}\neq 0}} -\frac{1}{\sigma_{ij}}\Big\} \leq \lambda_i \\ \leq \min\Big\{\kappa, \min_{\substack{j \\ x_{ij}\neq 0, \sigma_{ij}\neq 0}} \frac{1}{\sigma_{ij}}\Big(\frac{1}{x_{ij}} - 1\Big)\Big\} := \lambda_i^{(u)}.$$

The unconstrained solution is given by

$$\lambda_i' = \frac{\sum_{j=1}^{3}\sigma_{ij}x_{ij}(y_{ij}-x_{ij})}{\sum_{j=1}^{3}\sigma_{ij}^2x_{ij}^2}.$$

Thus the optimal solution for each pixel i is given by

$$\lambda_i^* = \max\{\lambda_i^{(l)}, \min\{\lambda_i', \lambda_i^{(u)}\}\}.$$

The final solution of the original problem allows only to choose k pixels to be changed. For each pixel i the quantity

$$\phi_i := \sum_{j=1}^{3}(y_{ij}-x_{ij})^2 - \sum_{j=1}^{3}(y_{ij} - (1+\lambda_i^*\sigma_{ij})x_{ij})^2$$

represents the difference in how much the objective increases between the cases $\lambda_i = 0$ (that is y_i is projected to x_i) and $\lambda_i = \lambda_i^*$. Since we want to minimize the objective function, the optimal solution is obtained by sorting $(\phi_i)_{i=1}^d$ in decreasing order π and setting

$$\lambda_{\pi_i}^{(final)} = \begin{cases} \lambda_{\pi_i}^* & \text{if } i = 1,\dots,k, \\ 0 & \text{else .} \end{cases}$$

Finally, the point belonging to $C(x)$ onto which y is projected is $z \in \mathbb{R}^{d\times 3}$ defined componentwise by

$$z_{ij} = (1+\lambda_i^{(final)}\sigma_{ij})x_{ij},\ i = 1,\dots,d,\ j = 1,\dots,3.$$

A.2. Gray-scale images

Since gray-scale images have only one color channel and, to get imperceivable manipulations, we use additive modifications as defined in (5), we project onto the set, given the original image x,

$$C(x) = \big\{z \in \mathbb{R}^d \,\big|\, \sum_{i=1}^{d} \mathbb{1}_{|z_i-x_i|>0} \leq k, \\ x_i - \kappa\sigma_i \leq z_i \leq x_i + \kappa\sigma_i, \\ 0 \leq z_i \leq 1\big\}.$$

Defining

$$l_i := \max\{x_i - \kappa\sigma_i, 0\}, \quad u_i := \min\{x_i + \kappa\sigma_i, 1\},$$

we can see that in this case the problem is equivalent to the projection onto the intersection of an l_0-ball and box constraints and then solved as illustrated in Section 4.1.

B. Experiments

We here report the details about the attacks, the attacked models and the parameters used in Section 5. The test accuracy (validation accuracy for Restricted ImageNet) of every model introduced in the paper is reported in Table 1.

test accuracy of the attacked models

section	dataset	model	accuracy
Section 5.1	MNIST	NiN [2]	99.66%
	CIFAR-10	NiN [2]	90.62%
	R-ImageNet	ResNet-50 [6]	94.46%
Sections 5.2, 5.3, B.2	MNIST	*plain* [3]	99.17%
		l_∞-*at* [3]	98.53%
		l_2-*at*	98.95%
		l_0-*at*	96.38%
		$l_0 + \sigma$-*at*	99.29%
	CIFAR-10	*plain*	88.38%
		l_∞-*at*	79.90%
		l_2-*at*	80.44%
		l_0-*at*	82.31%
		$l_0 + \sigma$-*at*	76.24%
	R-ImageNet	ResNet-50 [6]	94.46%

Table 1: **Accuracy of the attacked models.** We here report the accuracy on the test (validation set for Restricted ImageNet) set of the models introduced in Section 5.

B.1. Evaluation of l_0-attacks

The architecture used for this experiment is the Network in Network from [2], which we trained according to the code available at https://github.com/BIGBALLON/cifar-10-cnn, adapting it also to the case of MNIST (which has different input dimension).
We run PGD$_0$ with ten thresholds k (that is the maximum number of pixels that can be modified), which are $k \in \{2, 3, 4, 5, 6, 8, 10, 12, 15, 20\}$ for MNIST and $k \in \{1, 2, 3, 4, 6, 8, 10, 12, 15, 20\}$ for CIFAR-10.

Running times We report the average running times for one image for the experiments in Section 5.1 (the times for SparseFool are those given by our reimplementation, which uses DeepFool as implemented in [5]). MNIST: LocSearchAdv 0.6s (from [4]), PA 21s, CW 300s, SparseFool 2.5s, CornerSearch 9.8s, PGD$_0$ 0.06s (one threshold). CIFAR-10: LocSearchAdv 0.7s [4], PA 22s, CW 283s, SparseFool 1.0s, CornerSearch 3.6s, PGD$_0$ 0.19s (one threshold). ImageNet: SparseFool 17s, CornerSearch 953s, PGD$_0$ 13s (one threshold).

Stability of CornerSearch Since Algorithm 1 involves a component of random sampling, we want to analyse here how the performance of CornerSearch depends on it. Then, we run CornerSearch for 10 times on the models used in Table 1 of Section 5 and get the following statistics: MNIST, success rate (%) 97.37 ± 0.13, *mean* 9.12 ± 0.05, *median* 7 ± 0. CIFAR-10, success rate (%) 99.33 ± 0.12, *mean* 2.71 ± 0.02, *median* 2 ± 0. This means that our attack is stable across different runs.

Figure 1: Left: original. Right: l_0-adversarial example with clearly visible changes along axis-aligned edges.

B.2. Sparse and Imperceivable manipulations

In Figure 1 we show an example of how changes along axis-aligned edges are evident and easy to detect, even if the color is similar to that of some of the neighboring pixels. This provides a further justification of the heuristic we use to decide where the images can be perturbed in an invisible way.

In Figures 2, 3 and 4 we illustrate how the $l_0 + \sigma$-map attack produces sparse and imperceivable adversarial perturbations, while both the l_0- and the $l_0 + l_\infty$-attack either introduce colors which are non-homogeneous with those of the neighbors or modify pixels in an uniform background, which makes them easily visible.

MNIST In Figure 2 we show the differences among the adversarial examples found by our attacks CornerSearch (l_0-attack), $l_0 + l_\infty$-attack and σ-CornerSearch. We see that our $l_0 + \sigma$-map attack does not modify pixels in the background, far from the digit or in areas of uniform color in the interior of the digit itself.
Moreover, while, in the example shown in the lower left quadrant of Figure 2, the original image is correctly classified as a "4", we notice that the adversarial example found by $l_0 + l_\infty$-attack shows features of a new class. The modified pixels bridge the gap between the vertical segments in the upper part of the digit, making it similar to a "9". This means that this image does not clearly belong to any class and thus cannot be considered as an obvious adversarial sample. Conversely, the $l_0 + \sigma$-attack does not change the shape of the digit, which can still be clearly recognized as belonging to the original class "4".
The attacked model is the *plain* model from Section 5.3 (more details on the architecture below). For the $l_0 + l_\infty$-attack we use a bound on the l_∞-norm of the perturbation of $\delta = 0.2$.

CIFAR-10 We show in Figure 3 more examples built as those in Figure 3 of Section 5. The attacked model is the *plain* classifier from Section 5.3 (more details on the architecture below). For the $l_0 + l_\infty$-attack we use a bound on

the l_∞-norm of the perturbation of $\delta = 0.1$.

Retricted ImageNet We show in Figure 4 more examples created as those in Figure 4 of Section 5. The attacked model is the ResNet-50 from [6] (both weights and code are available at https://github.com/MadryLab/robust-features-code) already introduced in Section 5.1. For the $l_0 + l_\infty$-attack we use $\delta = 0.05$ as bound on the l_∞-norm of the perturbation.

B.3. Adversarial training

MNIST The architecture used is the same as in [3] (available at https://github.com/MadryLab/mnist_challenge), consisting of 2 convolutional layers, each followed by a max-pooling operation, and 2 dense layers. We trained our classifiers for 100 epochs with Adam [1].
The *plain* and l_∞-*at* models are those provided by [3] at https://github.com/MadryLab/mnist_challenge, while we trained l_2-*at* using the plain gradient as direction of the update for PGD, in contrast to the sign of the gradient which is used for adversarial training wrt the l_∞-norm.
For adversarial training wrt l_0-norm we use $k = 20$ (maximal number of pixels to be changed), 40 iterations and step size $\eta = {}^{30000}/_{255}$. For $l_0 + \sigma$-*at* we set $k = 100$, $\kappa = 0.9$ (for the bounds given by the σ-map), 40 iterations of gradient descent with step size $\eta = {}^{30000}/_{255}$.

CIFAR-10 We use a CNN with 8 convolutional layers, consisting of 96, 96, 192, 192, 192, 192, 192 and 384 feature maps respectively, and 2 dense layers of 1200 and 10 units. ReLU activation function is applied on the output of each layer, apart from the last one. We perform the training with data augmentation (in particular, random crops and random mirroring are applied) for 100 epochs and with Adam optimizer [1].
For adversarial training wrt l_0-norm we use $k = 20$ (number of pixels to be changed), 10 iterations of PGD with step size $\eta = {}^{30000}/_{255}$. For $l_0 + \sigma$-*at*, we use $k = 120$, $\kappa = 0.6$, 10 iterations of PGD with step size $\eta = {}^{30000}/_{255}$.

C. Adversarial examples of σ-PGD

We want here to compare the adversarial examples generated by our two methods, σ-CornerSearch and σ-PGD. In Figures 5 and 6 we show the perturbed images crafted by the two attacks, as well as the original images and the modifications rescaled so that each component is in [0,1] and the largest one equals 1. Moreover, for σ-PGD we report the results obtained with a smaller κ. The gray images indicate unsuccessful cases.
It is clear that σ-CornerSearch produces sparser perturbations. Moreover, σ-PGD with the same κ used for σ-CS gives more visible manipulations. We think that this is due to two reasons: first, the whole budget of k pixels to modify is always used by σ-PGD, while this does not happen with σ-CS, and second, σ-PGD aims at maximizing the loss inside the space of the allowed perturbations. This is possibly achieved by modifying neighboring pixels, which sometimes have slightly different colors, in opposite directions (that is with λ_i with different signs for different i). Conversely, σ-CornerSearch does not consider spatial relations among pixels, and thus it does not show this behaviour. However, as showed in the Figures, it is possible to recover less visible changes also for σ-PGD by decreasing κ, at the cost of a smaller success rate.

D. Propagation of sparse perturbations

To visualize the effect of very sparse perturbations on the decision made by the classifier we can check how the output of each hidden layer is modified when an adversarial example is given as input of the network instead of the original image. We here consider the *plain* model used in the comparison of the different adversarial training schemes on CIFAR-10 in Section 5 and the adversarial examples generated by CornerSearch on it.
We perform a forward pass first with the original images as input and then with the adversarially manipulated ones. In Figure 7 we plot (each color represents an image of the test set) the difference between the output values, after the activation function, of each unit of the network obtained with the two forward passes. The vertical segments separate the layers, and the leftmost section shows the difference of the inputs. The horizontal lines represent no difference in the values between the two forward passes. We can see how, going deeper into the network (towards right in Figure 7), the sparsity of the modifications decreases (the perturbed components of the input are on average 0.21%, those of the last hidden layer 19.45%) while their magnitude becomes larger, so that changing even only one pixel (that is three entries of the original image) causes a wrong classification.

References

[1] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. preprint, arXiv:1412.6980, 2014. 3

[2] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014. 2

[3] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 2, 3

[4] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. In *CVPR 2017 Workshops*, 2016. 2

[5] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *ICML Reliable Machine Learning in the Wild Workshop*, 2017. 2

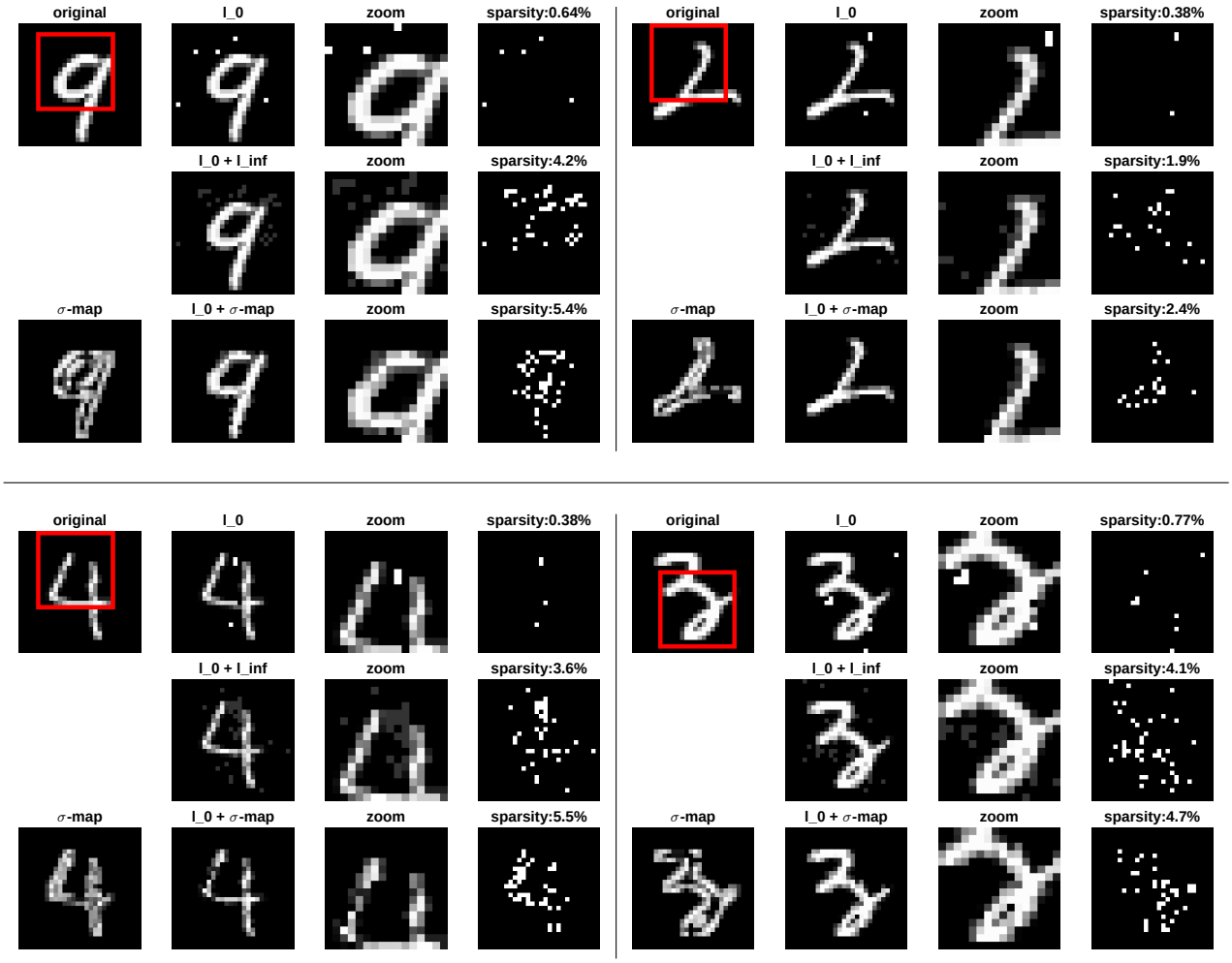

Figure 2: **Different attacks on MNIST.** We illustrate the differences of the adversarial examples (second column) found by CornerSearch (l_0), $l_0 + l_\infty$- and σ-CornerSearch, respectively first, second and third row. The third column shows the zoom of the area highlighted by the red box while the fourth column contains the map of the modified pixels (*sparsity* column). The original image can be found top left and the visualization of the σ-map (rescaled so that $\max_i \sigma_i = 1$) bottom left.

[6] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *ICLR*, 2019. 2, 3

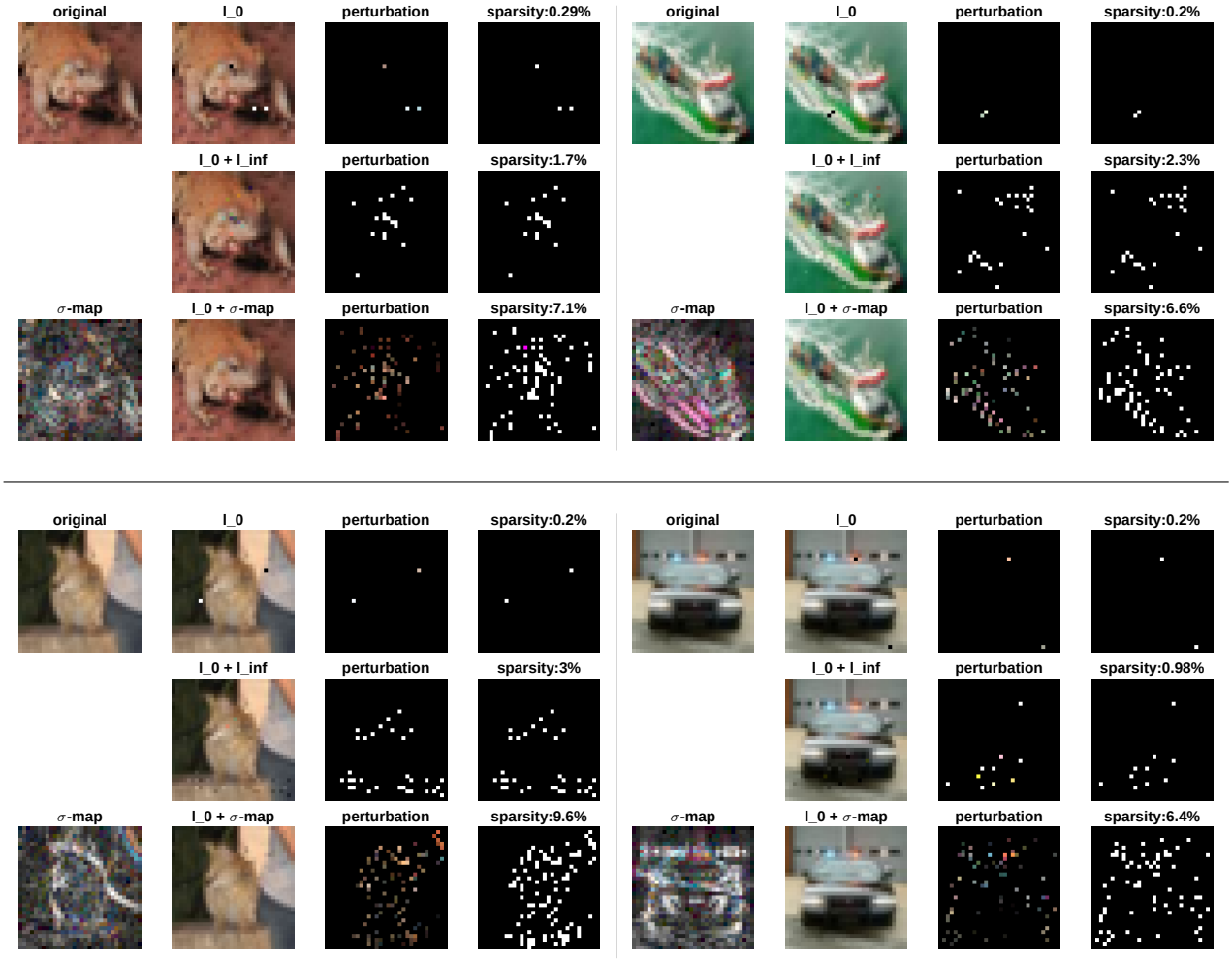

Figure 3: **Different attacks on CIFAR-10.** We illustrate the differences of the adversarial examples (second column) found by CornerSearch (l_0), $l_0 + l_\infty$-attack and σ-CornerSearch, respectively first, second and third row. The third column shows the adversarial perturbations rescaled to $[0, 1]$, the fourth the map of the modified pixels (*sparsity* column). The original image can be found top left and the RGB representation of the σ-map, rescaled so that $\max_{i,j} \sigma_{ij} = 1$, bottom left.

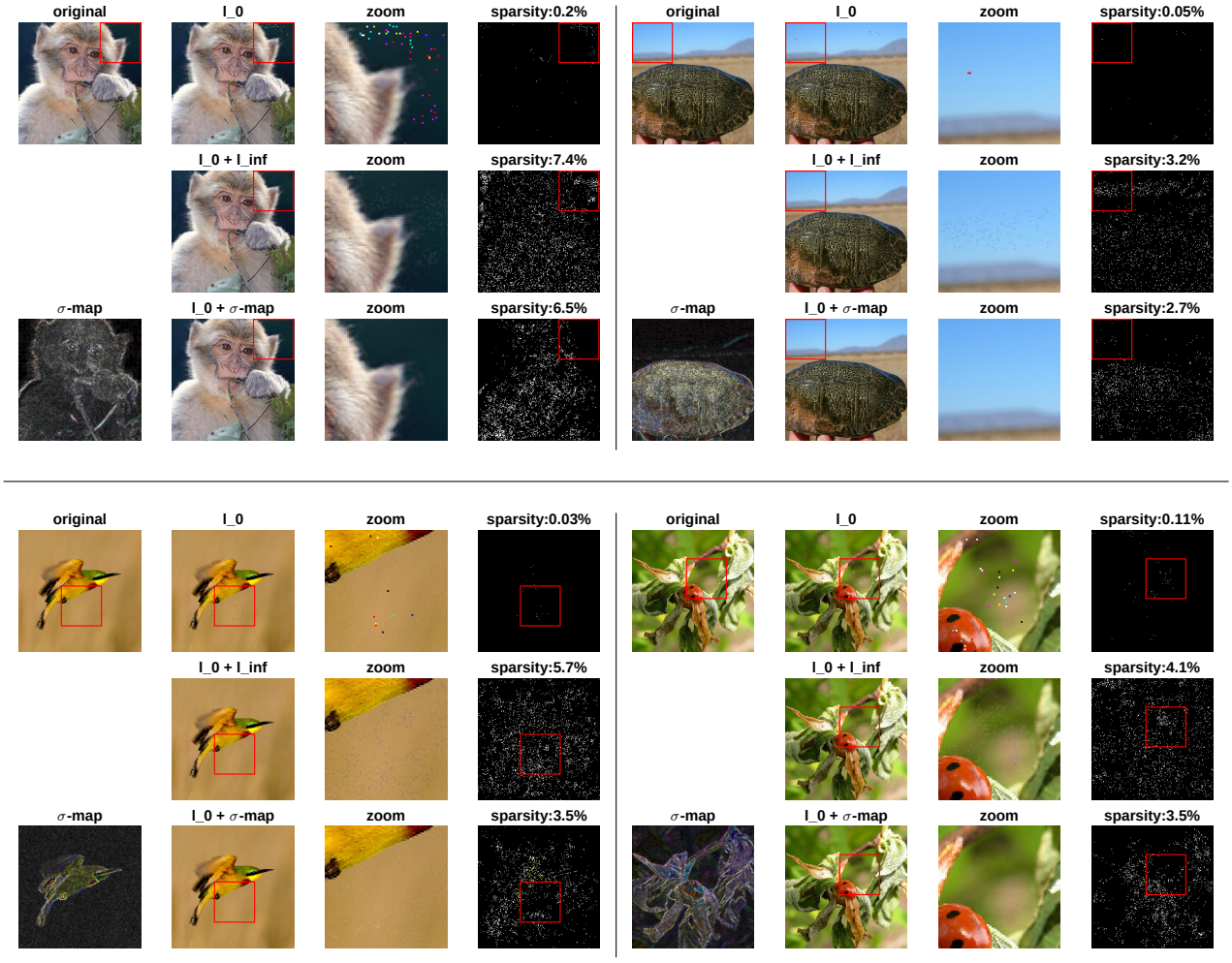

Figure 4: **Different attacks on Restricted ImageNet.** We illustrate the differences of the adversarial examples (second column, zoom in third column) found by CornerSearch (l_0), $l_0 + l_\infty$-attack and σ-CornerSearch, respectively first, second and third row. The fourth column shows the map of the modified pixels (*sparsity* column). The original image is in the top left and the RGB representation of the σ-map, rescaled so that $\max_{i,j} \sigma_{ij} = 1$, bottom left.

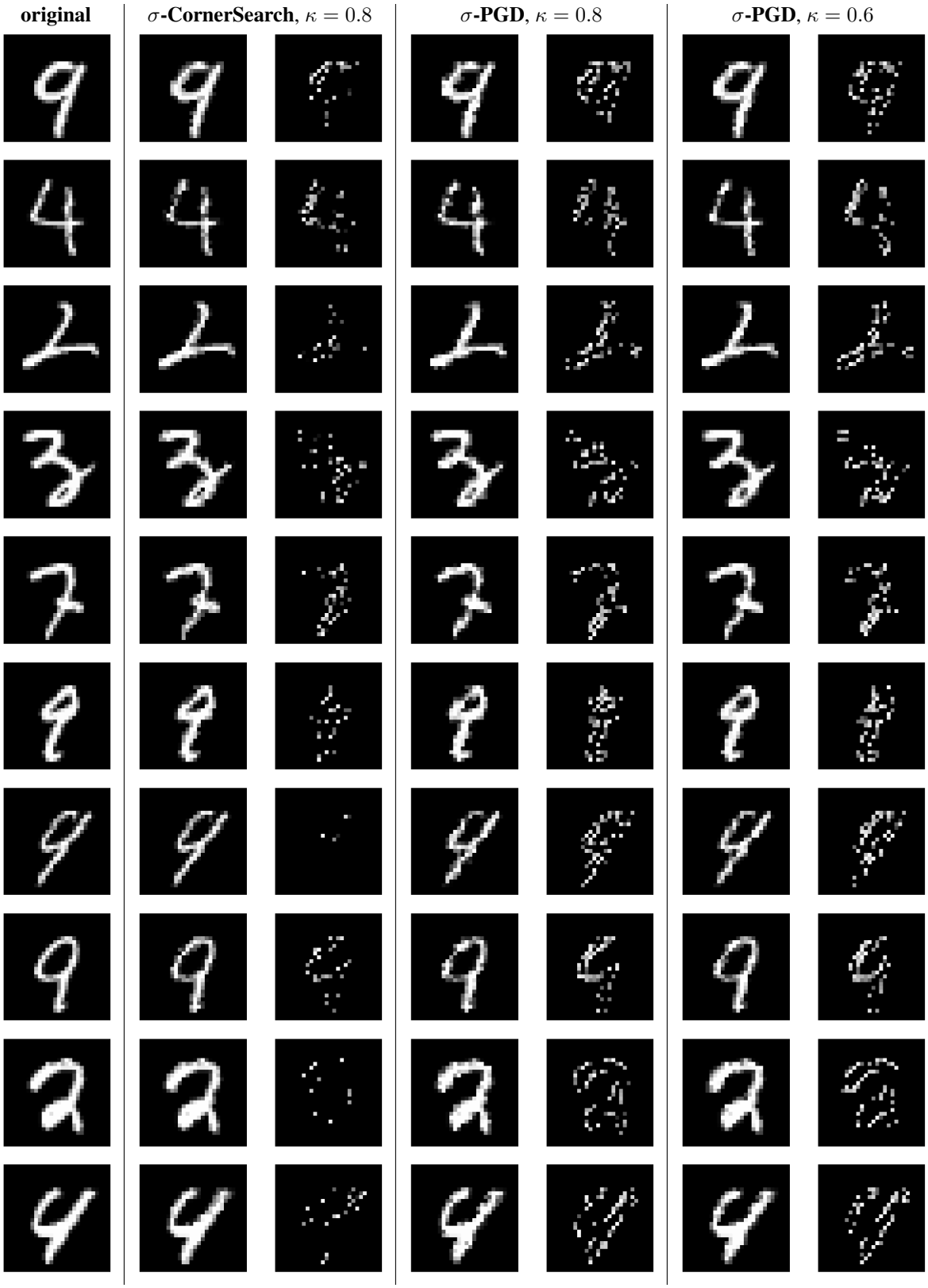

Figure 5: **Comparison σ-CornerSearch and σ-PGD on MNIST**. We show the adversarial examples generated by σ-CornerSearch with $\kappa = 0.8$, σ-PGD with $\kappa = 0.8$ and σ-PGD with $\kappa = 0.6$, together with the respective perturbations rescaled to [0,1]. The sparsity level used for σ-PGD is $k = 50$.

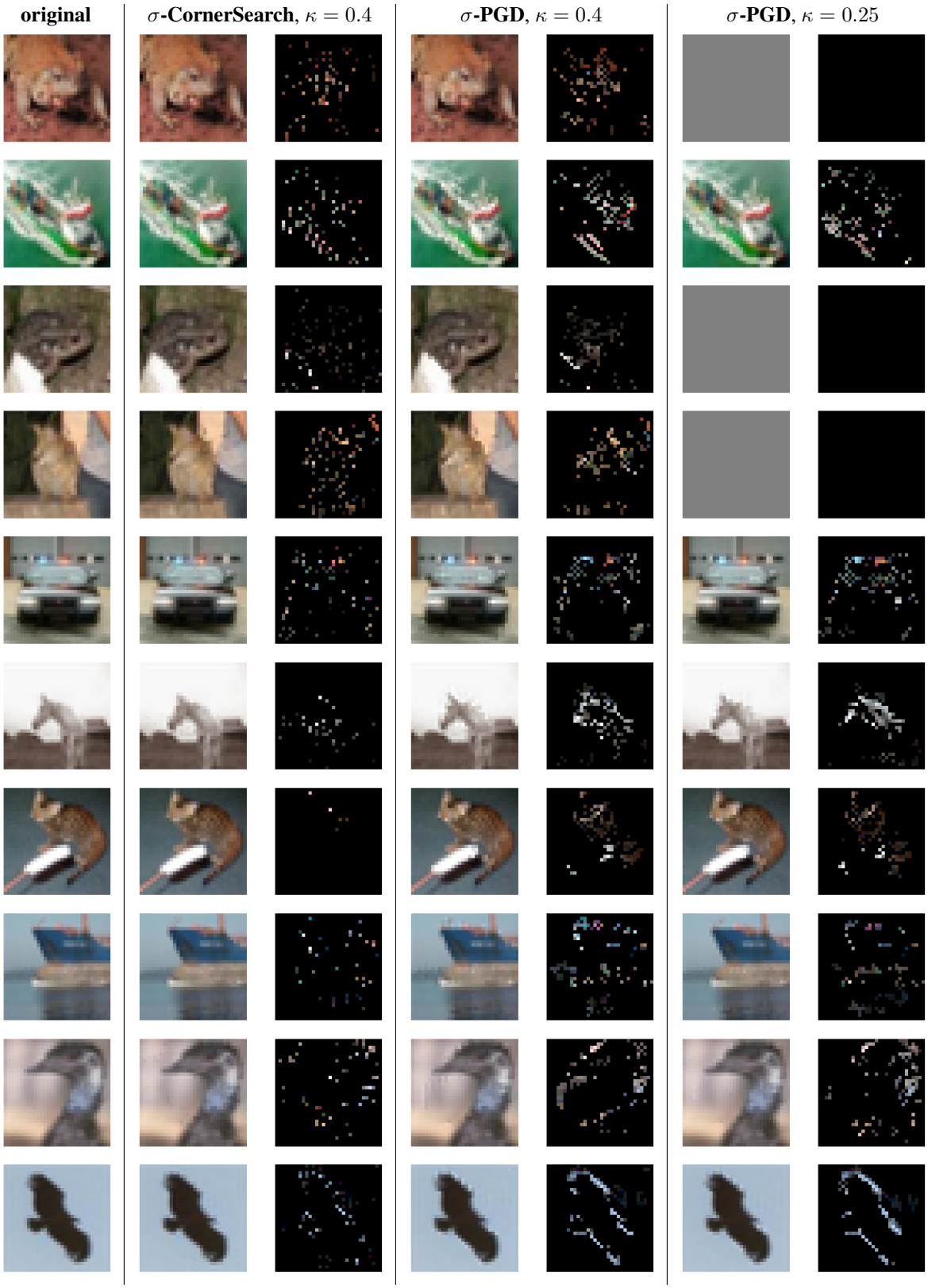

Figure 6: **Comparison σ-CornerSearch and σ-PGD on CIFAR-10**. We show adversarial examples generated by σ-CornerSearch with $\kappa = 0.4$, σ-PGD with $\kappa = 0.4$ and σ-PGD with $\kappa = 0.25$, together with the respective perturbations rescaled to [0,1]. The sparsity level used is $k = 100$. The gray images means the method could not find an adversarial manipulation.

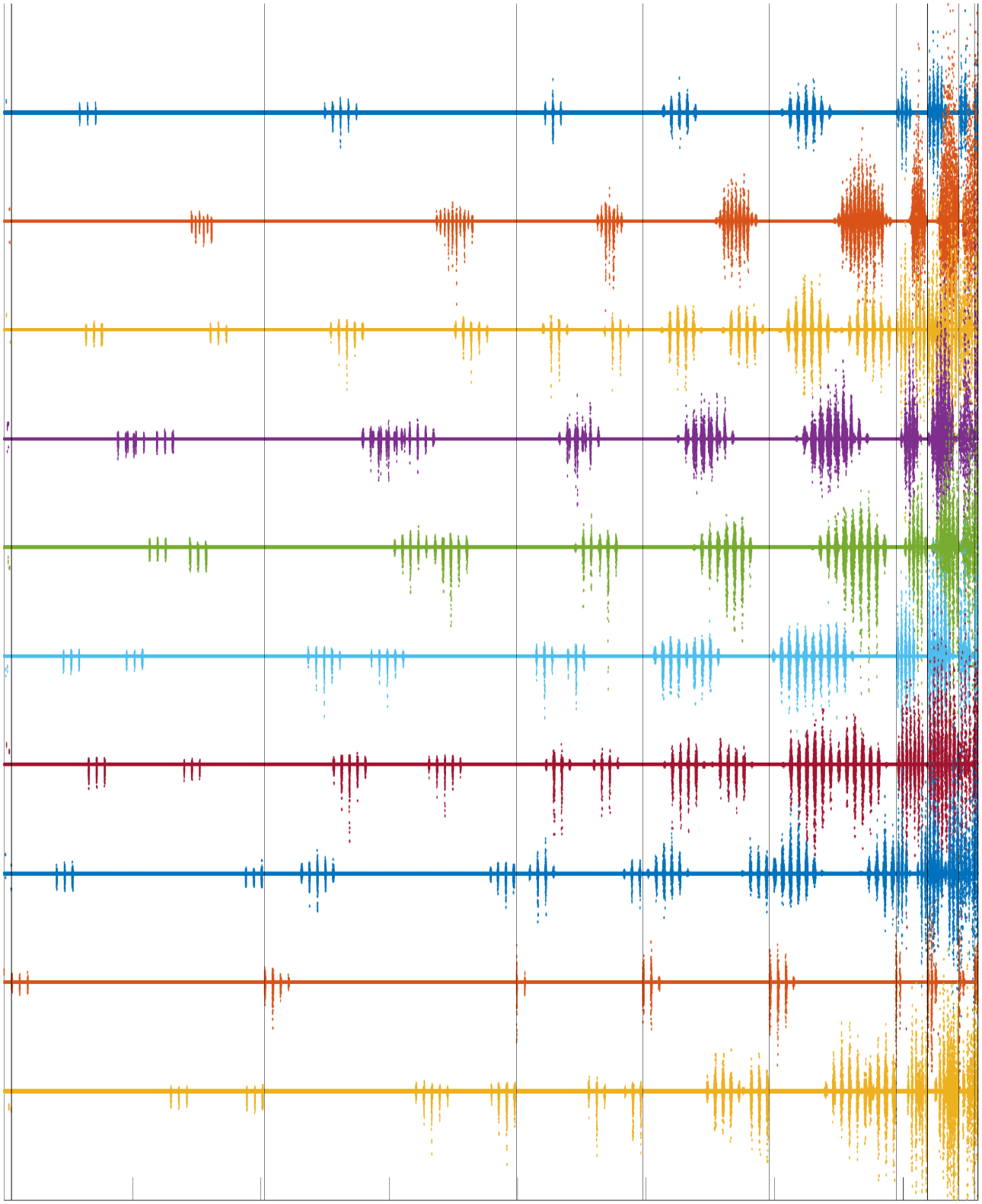

Figure 7: **Propagation of perturbations**. Difference in the values of each unit of the network obtained when propagating images of the test set and adversarial examples associated to them. The vertical segments distinguish the units of different layers, so that the input space is shown on left and the output on the right. Each color represents an image.