1. Our Distortion Settings

In the following, we provide additional information about our distortion settings.

1.1. Theoretical Background

For better readability, we repeat a few definitions from the main paper. We define clean input \( x = (x_{i,c}) \in \mathbb{I}^{H \times W \times C} \), distorted input \( x_r = (x_{r_{i,c}}) \in \mathbb{I}^{H \times W \times C} \), (constrained) distortion \( r = (r_{r_{i,c}}) \in [-1,1]^{H \times W \times C} \), and (effective) distortion strength

\[
\epsilon = \sqrt{\frac{1}{HWC} \mathbb{E} (||r||_2^2)}, \tag{1}
\]

following [2]. Further, \( H, W, \) and \( C \) refer to height, width, and number of color channels, respectively, \( \mathbb{I} \) is the input space defined as \( \mathbb{I} = [0,1] \), and indices \( i, c \) refer to pixel indices \( I = \{1,\ldots,H \cdot W\} \) and color channel indices \( C = \{1,\ldots,C\} \), respectively. Further, \( x, x_r, \) and \( r \) are related via

\[
r_r = x_r - x. \tag{2}
\]

1.2. On Defining an Unconstrained Distortion

We extend our formulation in Section 1.1 and respective related parts of our main paper with an unconstrained distortion \( \tilde{r} = (\tilde{r}_{\tilde{r}_{i,c}}) \in \mathbb{R}^{H \times W \times C} \), which is typically the output of any distortion algorithm, e.g., a Gaussian noise generator or an FGSM attack. As \( \tilde{r} \) is not constrained, a clipping operation has to be performed to guarantee \( x_{\tilde{r}_{i,c}} \in \mathbb{I} \), with

\[
x_{\tilde{r}_{i,c}} = \min(\max(x_{i,c} + \tilde{r}_{\tilde{r}_{i,c}}, 0), 1). \tag{3}
\]

If we consider \( x = (x_{i,c}) \) to be given, then (3) can be understood as a constraint applied to distortion \( \tilde{r} \). Thus, our unconstrained distortion \( \tilde{r} \) is indirectly subject to the constraint (3), resulting in the constrained distortion \( r \).

With increasing \( \epsilon \), more and more pixels will be affected by (3), resulting in differences between \( \tilde{r} \) and \( r \). However, for \( \epsilon \ll 1 \) as is the case in our paper, this effect can typically be neglected, which we assume in the following mathematical description, i.e., \( \tilde{r} \approx r \).

1.3. On Defining a Target Distortion Strength

In (1) we define the distortion strength \( \epsilon \), following [2]. However, note that this can be considered as an effective distortion strength, i.e., the distortion strength that is measured after having generated the distortion \( r_{r_{i,c}} \). During the generation process, however, we may already need to set a desired distortion strength—a classical chicken-and-egg situation. Thus, we define the target distortion strength \( \tau \), which, potentially jointly with the clean input \( x \), determines the strength of the resulting unconstrained distortion \( \tilde{r} \).

Typically, but not necessarily, we have \( \tau \) being close to \( \epsilon \). Next, we will explain the role of \( \tau \) for each of our investigated distortions and also the relation between \( \epsilon \) and \( \tau \).

1.4. On the Configuration of Our Distortions

In the following, we reintroduce our distortion types and emphasize on their configuration using the target distortion strength \( \tau \). For simplicity, we neglect the effect of clipping introduced in (3). Thus, we can assume \( r_{r_{i,c}} \approx \tilde{r}_{r_{i,c}} \), which is effectively a low distortion strength assumption.

**Gaussian Noise**: With Gaussian noise, it is fairly simple. We set the mean to 0 and the variance to \( \tau^2 \) for each element of \( \tilde{r} \). Accordingly, we have \( \mathbb{E} (||\tilde{r}_{\tilde{r}}||_2^2) = HWC \cdot \tau^2 \).

When inserting this expression into (1), we obtain \( \epsilon = \tau \) as our effective distortion strength \( \epsilon \).

**Salt-and-Pepper Noise**: We randomly set some elements \( x_{r_{i,c}} \) of \( x_r \) to 0 or 1, both with equal probability, i.e.,

\[
x_{r_{i,c}} = (x_{i,c} + r_{r_{i,c}}) \in \{0,1\}, \text{ for some } i, c. \tag{4}
\]

Thus, by design, there will be no difference between constrained distortion \( r_{r_{i,c}} \) and unconstrained distortion \( \tilde{r}_{r_{i,c}} \), instead, we truly have \( r_r = \tilde{r} \). Further, in practice, we observe the pixel expectation to be

\[
\mathbb{E}(x_{i,c}) = \frac{1}{HWC} \sum_{i \in \mathbb{I}} \sum_{c \in C} x_{i,c} = 0.5 + \delta, \tag{5}
\]

with a small value \( \delta \). If we now define \( n_{sp} \) to be the amount of pixels for which (4) holds (salt-and-pepper pixels), our
assumption in (5) yields
\[
\frac{1}{D} \mathbb{E} \left( \| r_\epsilon \|_2^2 \right) = \frac{n_{sp}}{2D} \cdot \left[ (0.5 - \delta)^2 + (0.5 + \delta)^2 \right] = \frac{n_{sp}}{D} \cdot \left[ \frac{1}{4} + \delta^2 \right] = \epsilon^2, \tag{6}
\]
with \( D = HWC \). Inserting \( \mathbb{E}(\| r_\epsilon \|_2^2) = \epsilon^2 \cdot HWC \) (cf. (1)) into (6), we obtain the amount of salt-and-pepper pixels
\[
n_{sp} = \frac{\epsilon^2}{\frac{1}{4} + \delta^2} \cdot HWC. \tag{7}
\]
Now, we can freely choose \( \epsilon = \epsilon \), and accordingly we can generate the salt-and-pepper noise on a ratio of
\[
\frac{n_{sp}}{HWC} = \frac{\frac{\epsilon^2}{\frac{1}{4} + \delta^2}}{HWC} = \epsilon \tag{8}
\]
of the pixels. Note that \( \delta \) is to be measured upfront from the data according to (5), or one can simply assume \( \delta = 0 \) (our choice).

**FGSM and PGD:** Considering FGSM and PGD, we follow [1, 3] and set \( \tau \) to be the upper bound for the *unconstrained* distortion \( \tilde{r}_\epsilon \) yielding
\[
\| \tilde{r}_\epsilon \|_\infty \leq \tau. \tag{10}
\]
For FGSM, we can actually further tighten (10) to \( |\tilde{r}_{\epsilon,c}^i| = \tau \) as we have
\[
\tilde{r}_\epsilon = \epsilon \cdot \text{sign}(\nabla_x J), \tag{11}
\]
with the sign operator \( \text{sign}() \in \{-1, 1\} \) and with \( \nabla_x J \) as the input gradient with respect to loss \( J \). Under the low distortion strength assumption, we have \( r_\epsilon = \tilde{r}_\epsilon \), and therefore we obtain \( \epsilon = \epsilon \) for FGSM.

For PGD, (10) is usually ensured by a clipping operation similar to (3). As PGD follows an iterative optimization process for obtaining \( \tilde{r}_\epsilon \) (we set the number of iterations to 40 and choose a step size of \( \frac{2}{255} \)), we cannot ensure that all pixel values of \( \tilde{r}_\epsilon \) actually converge to the upper bound of (10), as with FGSM, for instance. Thus, we usually obtain \( \epsilon \leq \tau \) in practice.

**References**

