Supplementary Material for the Paper: Performance Prediction for Semantic Segmentation by a Self-Supervised Image Reconstruction Decoder

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 $\boldsymbol{x} = (x_{i,c}) \in \mathbb{I}^{H \times W \times C}$, distorted input $\boldsymbol{x}_\epsilon = (x_{\epsilon,i,c}) \in \mathbb{I}^{H \times W \times C}$, (constrained) distortion $\boldsymbol{r}_\epsilon = (r_{\epsilon,i,c}) \in [-1,1]^{H \times W \times C}$, and (effective) distortion strength

$$\epsilon = \sqrt{\frac{1}{HWC}\mathbb{E}\left(||\boldsymbol{r}_\epsilon||_2^2\right)}, \tag{1}$$

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 $\mathcal{I} = \{1, ..., H \cdot W\}$ and color channel indices $\mathcal{C} = \{1, ..., C\}$, respectively. Further, $\boldsymbol{x}$, $\boldsymbol{x}_\epsilon$, and $\boldsymbol{r}_\epsilon$ are related via

$$\boldsymbol{r}_\epsilon = \boldsymbol{x}_\epsilon - \boldsymbol{x}. \tag{2}$$

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 $\tilde{\boldsymbol{r}}_\epsilon = (\tilde{r}_{\epsilon,i,c}) \in \mathbb{R}^{H \times W \times C}$, which is typically the output of any distortion algorithm, e.g., a Gaussian noise generator or an FGSM attack. As $\tilde{\boldsymbol{r}}_\epsilon$ is not constrained, a clipping operation has to be performed to guarantee $x_{\epsilon,i,c} \in \mathbb{I}$, with

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

If we consider $\boldsymbol{x} = (x_{i,c})$ to be given, then (3) can be understood as a constraint applied to distortion $\tilde{\boldsymbol{r}}_\epsilon$. Thus, our unconstrained distortion $\tilde{\boldsymbol{r}}_\epsilon$ is indirectly subject to the constraint (3), resulting in the constrained distortion $\boldsymbol{r}_\epsilon$ (2). With increasing ϵ, more and more pixels will be affected by (3), resulting in differences between $\tilde{\boldsymbol{r}}_\epsilon$ and $\boldsymbol{r}_\epsilon$. 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., $\tilde{\boldsymbol{r}}_\epsilon \approx \boldsymbol{r}_\epsilon$.

1.3. On Defining a Target Distortion Strength

In (1) we define the distortion strength ϵ, 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 $\boldsymbol{r}_\epsilon$. 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* $\bar{\epsilon}$, which, potentially jointly with the clean input $\boldsymbol{x}$, determines the strength of the resulting unconstrained distortion $\tilde{\boldsymbol{r}}_\epsilon$. Typically, but not necessarily, we have $\bar{\epsilon}$ being close to ϵ. Next, we will explain the role of $\bar{\epsilon}$ for each of our investigated distortions and also the relation between ϵ and $\bar{\epsilon}$.

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 $\bar{\epsilon}$. For simplicity, we neglect the effect of clipping introduced in (3). Thus, we can assume $r_{\epsilon,i,c} \approx \tilde{r}_{\epsilon,i,c}$, 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 $\bar{\epsilon}^2$ for each element of $\tilde{\boldsymbol{r}}_\epsilon$. Accordingly, we have $\mathbb{E}\left(||\tilde{\boldsymbol{r}}_\epsilon||_2^2\right) = HWC \cdot \bar{\epsilon}^2$. When inserting this expression into (1), we obtain $\epsilon = \bar{\epsilon}$ as our effective distortion strength ϵ.

Salt-and-Pepper Noise: We randomly set some elements $x_{\epsilon,i,c}$ of $\boldsymbol{x}_\epsilon$ to 0 or 1, both with equal probability, i.e.,

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

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

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

with a small value δ. 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

$$\frac{1}{D}\mathbb{E}\left(||\boldsymbol{r}_\epsilon||_2^2\right) = \frac{n_{\mathrm{sp}}}{2D} \cdot \left[(0.5-\delta)^2 + (0.5+\delta)^2\right] \quad (6)$$

$$= \frac{n_{\mathrm{sp}}}{D} \cdot \left[\frac{1}{4} + \delta^2\right] = \epsilon^2, \quad (7)$$

with $D = HWC$. Inserting $\mathbb{E}(\|\boldsymbol{r}_\epsilon\|_2^2) = \epsilon^2 \cdot HWC$ (cf. (1)) into (6), we obtain the amount of salt-and-pepper pixels

$$n_{\mathrm{sp}} = \frac{\epsilon^2}{\frac{1}{4} + \delta^2} \cdot HWC. \quad (8)$$

Now, we can freely choose $\overline{\epsilon} = \epsilon$, and accordingly we can generate the salt-and-pepper noise on a ratio of

$$\frac{n_{\mathrm{sp}}}{HWC} = \frac{\overline{\epsilon}^2}{\frac{1}{4} + \delta^2} \quad (9)$$

of the pixels. Note that δ 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 $\overline{\epsilon}$ to be the upper bound for the *unconstrained* distortion $\tilde{\boldsymbol{r}}_\epsilon$ yielding

$$\|\tilde{\boldsymbol{r}}_\epsilon\|_\infty \leq \overline{\epsilon}. \quad (10)$$

For FGSM, we can actually further tighten (10) to $|\tilde{r}_{\epsilon,i,c}| = \overline{\epsilon}$ as we have

$$\tilde{\boldsymbol{r}}_\epsilon = \overline{\epsilon} \cdot \mathrm{sign}(\boldsymbol{\nabla}_{\boldsymbol{x}} J), \quad (11)$$

with the sign operator $\mathrm{sign}() \in \{-1, 1\}$ and with $\boldsymbol{\nabla}_{\boldsymbol{x}} J$ as the input gradient with respect to loss J. Under the low distortion strength assumption, we have $\boldsymbol{r}_\epsilon = \tilde{\boldsymbol{r}}_\epsilon$, and therefore we obtain $\epsilon = \tilde{\epsilon}$ for FGSM.

For PGD, (10) is usually ensured by a clipping operation similar to (3). As PGD follows an iterative optimization process for obtaining $\tilde{\boldsymbol{r}}_\epsilon$ (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 $\tilde{\boldsymbol{r}}_\epsilon$ actually converge to the upper bound of (10), as with FGSM, for instance. Thus, we usually obtain $\epsilon \leq \overline{\epsilon}$ in practice.

References

[1] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *Proc. of ICLR*, pages 1–10, San Diego, CA, USA, May 2015. 2

[2] Marvin Klingner and Tim Fingscheidt. Online Performance Prediction of Perception DNNs by Multi-Task Learning with Depth Estimation. *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, 22(7):4670–4683, July 2021. 1

[3] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proc. of ICLR*, pages 1–28, Vancouver, BC, Canada, Apr. 2018. 2