Performance Prediction for Semantic Segmentation by a Self-Supervised Image Reconstruction Decoder

Andreas Bär[1] Marvin Klingner[1] Jonas Löhdefink[1] Fabian Hüger[2]
Peter Schlicht[2] Tim Fingscheidt[1]

{andreas.baer, m.klingner, j.loehdefink, t.fingscheidt}@tu-bs.de
{fabian.hueger, peter.schlicht}@volkswagen.de

[1]Technische Universität Braunschweig, Germany [2]CARIAD SE, Germany

Abstract

In supervised learning, a deep neural network's performance is measured using ground truth data. In semantic segmentation, ground truth data is sparse, requires an expensive annotation process, and, most importantly, it is not available during online operation. To tackle this problem, recent works propose various forms of performance prediction. However, they either rely on inference data histograms, additional sensors, or additional training data. In this paper, we propose a novel per-image performance prediction for semantic segmentation, with (i) no need for additional sensors (sensor efficiency), (ii) no need for additional training data (data efficiency), and (iii) no need for a dedicated retraining of the semantic segmentation (training efficiency). Specifically, we extend an already trained semantic segmentation network having fixed parameters with an image reconstruction decoder. After training and a subsequent regression, the image reconstruction quality is evaluated to predict the semantic segmentation performance. We demonstrate our method's effectiveness with a new state-of-the-art benchmark both on KITTI and Cityscapes for image-only input methods, on Cityscapes even excelling a LiDAR-supported benchmark.

1. Introduction

Ground truth data for deep neural networks (DNNs) is expensive, sparsely available for training and test, and typically not available during operation. In consequence, *the performance of a DNN in operation remains unknown*. This poses a problem, if we consider safety-critical tasks, such as semantic segmentation for highly automated driving [40], operating in (adversarially) distorted environments [1,4,20] or different domains, where DNNs often struggle to perform well. The latter issue is typically handled with domain adaptation techniques [5, 28, 45]. The former can be addressed by increasing the robustness to distorted inputs [23], detecting distorted data [3], or eliminating the distortion [21]. However, there cannot be any guarantees that DNN performance is always within the bounds for a safe operation.

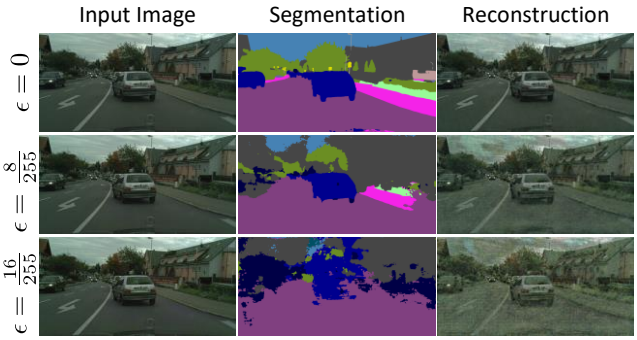


Figure 1. **General motivation** of our approach. Triplets of input image (left), segmentation output (middle) and reconstruction output (right) under clean conditions ($\epsilon = 0$) as well as perturbed conditions (FGSM [12], $\epsilon \in \{8, 16\} \cdot \frac{1}{255}$) (rows). Observation: Increasing the distortion strength ϵ leads to both, a decreased segmentation and reconstruction quality in a multi-task network.

A different perspective to this problem is to directly estimate a DNN's performance. In this paper, instead of assuming a certain performance from the validation in the lab, we propose to estimate the actual performance of a semantic segmentation network during inference. Competing works so far either do not provide a per-image performance estimate [34] or rely on additional sensors [24, 25]. In this paper, we propose a performance prediction for semantic segmentation on a per-image basis, with no need for additional training data or sensors. For this purpose, we combine semantic segmentation with image reconstruction (cf. Fig. 1), hypothesizing a correlation between both tasks. Specifi-

cally, we propose to attach an additional performance prediction (based on image reconstruction) to the encoder $\boldsymbol{E}^{\text{seg}}$ of a trained semantic segmentation network (cf. Fig. 2). The idea is to derive an $mIoU$ estimate $\widehat{mIoU}$ from the difference of the reconstructed image $\hat{\boldsymbol{x}}$ and the input image $\boldsymbol{x}$. The benefit of attaching the image reconstruction decoder *after* training the semantic segmentation network is an unchanged segmentation performance which is not influenced by multi-task learning. Moreover, this strategy provides an approach being highly transferable to other architectures and tasks, whenever a latent representation can be used to reconstruct the input.

Our contributions are: (i) a performance prediction for semantic segmentation based on a self-supervised image reconstruction decoder, that is (ii) sensor-efficient, as we only use the input image, (iii) training-efficient, as we only need to train the additional image reconstruction decoder, (iv) data-efficient, as the additional parts solely rely on data the semantic segmentation was trained on, and (v) setting a new state-of-the-art benchmark both on KITTI [11] and Cityscapes [8] regarding image-only input methods, on Cityscapes even excelling a LiDAR-supported benchmark.

2. Related Works

Robustness of Semantic Segmentation Models: Deep neural networks for semantic segmentation [32, 40] are vulnerable to a variety of input distortions [20] and adversarial examples [1, 15, 38]. This problem can be tackled from different perspectives, including preprocessing algorithms [4, 6, 7, 13, 21, 31], advanced learning techniques [9, 18, 19, 23, 36, 37, 44, 48], architectural changes [42], or detection mechanisms [3, 26, 30, 46, 47]. In this paper, we evaluate our method using clean and distorted input images [12,35] from Cityscapes [8] and KITTI [11]. However, we neither remove the distortion nor robustify the underlying network. Instead, we propose a performance prediction method that rather indicates distorted inputs by a low performance, and thus, indirectly contributes to detection mechanisms.

Performance Prediction: The performance of a deep neural network is evaluated using ground truth data, which is usually unavailable during inference. Estimating the network output's uncertainty is a solution to this problem. Two famous approaches are Monte-Carlo dropout [10] and deep ensembles [29], both shown to be scalable to semantic segmentation [14]. Nonetheless, both introduce noticable computational complexity. A different way of approaching ground truth unavailability during inference is to predict the current segmentation performance or error. The latter is addressed by Rottmann et al. [43], who propose a method for predicting the error of output segments. Our approach deals with the former, with the closest prior works in [24,25,34]. The authors of [34] use an autoencoder which runs *in parallel* to a trained semantic segmentation during inference. Their approach, however, is limited to domain shifts and *does not deliver per-image estimates*. The authors of [24, 25] propose a multi-task network with an auxiliary depth estimation decoder. While their approach delivers per-image estimates, they rely on *additional LiDAR data* during inference and *video data during training*.

In contrast, we provide a performance prediction which (a) yields per-image estimates, (b) relies only on camera data during inference, and (c) is efficient in terms of training data and training complexity.

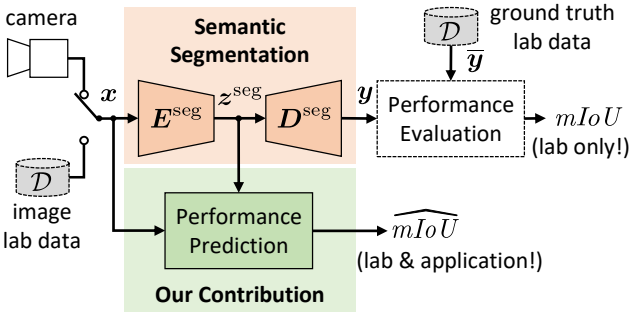


Figure 2. **Performance prediction setting**. A trained semantic segmentation network (with encoder $\boldsymbol{E}^{\text{seg}}$ and decoder $\boldsymbol{D}^{\text{seg}}$), is extended by a performance prediction module (cf. Fig. 3 for further details). The performance prediction module takes the latent representation $\boldsymbol{z}^{\text{seg}}$ and the camera image $\boldsymbol{x}$ as input and outputs an estimate $\widehat{mIoU}$ (9) w.r.t. the actual $mIoU$ (5). Note that $mIoU$ relies on ground truth data while $\widehat{mIoU}$ does not.

3. Method Description

In this section, we introduce the theoretical background, our proposed performance prediction framework, and its configuration.

3.1. Theoretical Background

Let $\boldsymbol{x} = (x_{i,c}) \in \mathbb{I}^{H \times W \times C}$ be a normalized image, with height H, width W, $C = 3$ color channels, pixel index $i \in \mathcal{I} = \{1, ..., H \cdot W\}$, color channel index $c \in \mathcal{C} = \{1, ..., C\}$, and $\mathbb{I} = [0, 1]$.

Semantic Segmentation: A semantic segmentation network maps the input $\boldsymbol{x}$ to the output $\boldsymbol{y} = (y_{i,s}) \in \mathbb{I}^{H \times W \times |\mathcal{S}|}$, with $\sum_{s \in \mathcal{S}} y_{i,s} = 1$, and class index $s \in \mathcal{S} = \{1, ..., |\mathcal{S}|\}$. Further, $y_{i,s} = P(s|i, \boldsymbol{x})$ is considered to be a posterior probability. We can also reformulate the mapping into two steps, see Fig. 2. We introduce the latent representation $\boldsymbol{z}^{\text{seg}} = \boldsymbol{E}^{\text{seg}}(\boldsymbol{x}), \boldsymbol{z}^{\text{seg}} \in \mathbb{R}^{\frac{H}{\delta} \times \frac{W}{\delta} \times C'}$, and the final output $\boldsymbol{y} = \boldsymbol{D}^{\text{seg}}(\boldsymbol{z}^{\text{seg}})$, with $\boldsymbol{E}^{\text{seg}}, \boldsymbol{D}^{\text{seg}}$ being the semantic segmentation network's encoder and decoder, respectively, and δ, $C' \in \mathbb{N}$ being the encoder's spatial downsampling factor and number of output feature maps, respectively. Note that lateral encoder-decoder connections

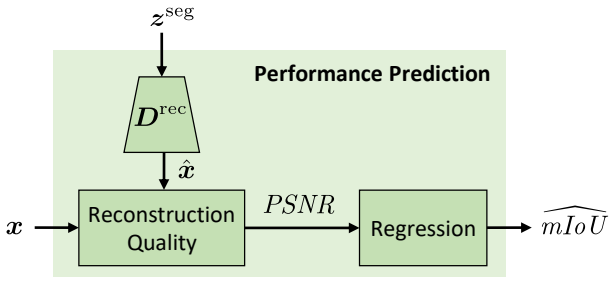


Figure 3. **Performance prediction module**. The semantic segmentation's latent representation $\boldsymbol{z}^{\text{seg}}$ is fed into an image reconstruction decoder $\boldsymbol{D}^{\text{rec}}$ yielding a reconstruction $\hat{\boldsymbol{x}}$ of the input $\boldsymbol{x}$. Next, the reconstruction quality is measured in terms of $PSNR$ (7). Finally, $mIoU$ (5) is estimated using a subsequent regression yielding $\widehat{mIoU}$ (9).

are also possible, but we neglect them here for simplicity. Further, we define $\overline{\boldsymbol{y}} = (\overline{y}_{i,s}) \in \{0,1\}^{H\times W\times|\mathcal{S}|}$, with $\sum_{s\in\mathcal{S}} \overline{y}_{i,s} = 1$, to be the one-hot-encoded ground truth semantic segmentation. Tensors $\boldsymbol{y}$ and $\overline{\boldsymbol{y}}$ are then compared via the cross-entropy loss

$$J^{\text{seg}} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\sum_{s\in\mathcal{S}} \overline{y}_{i,s}\cdot\log(y_{i,s}), \tag{1}$$

which is minimized during training.

Input Image Reconstruction: The general work flow of our image reconstruction is depicted in Fig. 3. An image reconstruction network maps the latent representation $\boldsymbol{z}^{\text{seg}}$ to the reconstructed image $\hat{\boldsymbol{x}} = \boldsymbol{D}^{\text{rec}}(\boldsymbol{z}^{\text{seg}}), \hat{\boldsymbol{x}} \in \mathbb{I}^{H\times W\times C}$. Here, $\boldsymbol{D}^{\text{rec}}$ represents a reconstruction decoder. The mean squared reconstruction error

$$J^{\text{rec}} = \frac{1}{HWC}\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2 \tag{2}$$

serves as a training objective and is minimized.

Input Image Distortion: Let $\boldsymbol{x}_\epsilon = (x_{\epsilon,i,c}) \in \mathbb{I}^{H\times W\times C}$ be a distorted input image and $\boldsymbol{r}_\epsilon = (r_{\epsilon,i,c}) \in [-1,1]^{H\times W\times C}$ its respective distortion which is given by $\boldsymbol{r}_\epsilon = \boldsymbol{x}_\epsilon - \boldsymbol{x}$. The subscript ϵ refers to the *effective* distortion strength defined as

$$\epsilon = \sqrt{\frac{1}{HWC}\mathbb{E}\left(||\boldsymbol{r}_\epsilon||_2^2\right)} \tag{3}$$

following Klingner et al. [25], with expectation value $\mathbb{E}(\cdot)$. We additionally introduce the *target* distortion strength $\overline{\epsilon}$. The exact difference between target and effective distortion strength is emphasized in the supplementary material. In short, $\overline{\epsilon}$ is set to generate a distortion, while ϵ is measured after distortion generation.

3.2. Performance Prediction Framework

In the following we introduce our performance prediction framework. We start with the additional image reconstruction decoder, continue by elaborating on the training process, and end with the explanation of the actual performance prediction scheme.

Attaching an Image Reconstruction Decoder: Our proposed method extends a semantic segmentation network by a decoder for image reconstruction (cf. Fig. 3). Given $\boldsymbol{E}^{\text{seg}}$ by the underlying semantic segmentation, we are free to decide on the decoder architecture $\boldsymbol{D}^{\text{rec}}$. Here, we consider two general architectural design options.

In the first design option for $\boldsymbol{D}^{\text{rec}}$, the semantic segmentation decoder's architecture $\boldsymbol{D}^{\text{seg}}$ serves as a basis for $\boldsymbol{D}^{\text{rec}}$ (e.g., as done in [24,25,27]). As semantic segmentation decoders usually output $|\mathcal{S}|$ output feature maps, each representing a distinct class, we change the decoder's number of output feature maps from $|\mathcal{S}|$ to $C = 3$ to deliver an RGB image in analogy to the input.

The second design option is mirroring the encoder $\boldsymbol{E}^{\text{seg}}$. For this purpose, we first rebuild the encoder architecture recursively in a simple fashion. Note that during this process, the input feature maps and output feature maps of all convolutions are flipped. Then, we replace all strided convolutions in the encoder (performing downsampling) by nearest neighbor upsampling with a subsequent convolution to avoid checkerboard artifacts [39].

Performing a Sequential Training: As a next step, both the segmentation and reconstruction tasks need to be trained. As already pointed out in [24,25], there are various training strategies for a multi-task network for performance prediction. Both tasks can be either trained sequentially, in parallel, or in a hybrid fashion. Parallel training as well as hybrid training update the encoder weights with gradient information from both tasks. Preliminary experiments with our setup showed that semantic segmentation quality does not benefit from parallel or hybrid reconstruction decoder training but rather worsens.

Sequential training leaves us with two strategies. In the first of two stages, we train the shared encoder and segmentation decoder, minimizing (1), and in the second stage we only train the reconstruction decoder, minimizing (2). The alternative would be doing it vice versa. As the representation learned by the encoder plays a crucial role for segmentation performance, we only consider the first strategy. *This training strategy not only ensures a high semantic segmentation quality but also allows adopting arbitrary trained semantic segmentation networks.*

Predicting Performance via Regression: The architecture modification and sequential training protocol are completed by a regression which is obtained after both tasks have been trained. The performance of the image reconstruction in terms of peak signal-to-noise ratio $PSNR$ is evaluated by comparing the image reconstruction $\hat{\boldsymbol{x}}$ with the input image $\boldsymbol{x}$ (cf. Fig. 3). Both are available at any time during inference. Note that this does not hold for seman-

tic segmentation as computing the mean intersection over union $mIoU$ involves comparing the segmentation output to its ground truth segmentation. However, if we assume a correlation between semantic segmentation and image reconstruction (cf. Fig. 1), a regression analysis between both tasks' performance metrics can be performed. Specifically, we want to apply the regression to estimate the semantic segmentation performance in terms of mean intersection over union, i.e., $\widehat{mIoU}$, from the peak signal-to-noise ratio $PSNR$ (cf. Fig. 3). The details of preparing and executing this regression are explained in the following.

3.3. Configuration of the Performance Prediction

After sequential training of both tasks, the performance prediction needs to be configured before inference. The process is split into three steps: semantic segmentation evaluation, image reconstruction evaluation, and regression calibration. In addition, we also elaborate on how the quality of the performance prediction will be reported later on.

Semantic Segmentation Evaluation: A semantic segmentation network is evaluated using the mean intersection over union $mIoU$, see Fig. 2. Class-wise true positives $TP_{s,\epsilon} = \sum_{n\in\mathcal{N}} TP_{n,s,\epsilon}$, false positives $FP_{s,\epsilon} = \sum_{n\in\mathcal{N}} FP_{n,s,\epsilon}$, and false negatives $FN_{s,\epsilon} = \sum_{n\in\mathcal{N}} FN_{n,s,\epsilon}$ are computed over a dataset $\mathcal{D}$ yielding

$$mIoU_{\epsilon} = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} \frac{TP_{s,\epsilon}}{TP_{s,\epsilon}+FP_{s,\epsilon}+FN_{s,\epsilon}}. \quad (4)$$

The subscript ϵ indicates an average distortion strength as defined in (3), $n \in \mathcal{N}$ is an image index from set $\mathcal{N} = \{1, ..., |\mathcal{D}|\}$, and $s \in \mathcal{S}$ is a class index from set $\mathcal{S} = \{1, ..., |\mathcal{S}|\}$. For the regression calibration, we compute the $mIoU$ *per image* rather than per dataset, yielding

$$mIoU_{n,\epsilon} = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} \frac{TP_{n,s,\epsilon}}{TP_{n,s,\epsilon}+FP_{n,s,\epsilon}+FN_{n,s,\epsilon}}. \quad (5)$$

Further, we define the average over images as

$$\overline{mIoU}_{\epsilon} = \frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}} mIoU_{n,\epsilon}. \quad (6)$$

Image Reconstruction Evaluation: An image reconstruction network is evaluated using the peak signal-to-noise ratio $PSNR$. In our chosen input space $\mathbb{I}$, this is defined (in dB) as

$$PSNR_{n,\epsilon} = -10\log\left(J^{\text{rec}}_{n,\epsilon}\right), \quad (7)$$

with its average over images

$$\overline{PSNR}_{\epsilon} = \frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}} PSNR_{n,\epsilon}. \quad (8)$$

Note that also here we refer to the distortion strength ϵ and the image index n in the subscripts. Further, $J^{\text{rec}}_{n,\epsilon}$ refers to (2) when a distorted image $\boldsymbol{x}_{n,\epsilon}$ is fed to the network.

Regression Calibration: After training, we can calibrate our proposed performance prediction by solving a regression problem [24, 25]. For this, we first generate $mIoU_{n,\epsilon}$ (5) and $PSNR_{n,\epsilon}$ (7) for all images with indices $n \in \mathcal{N}$ of dataset $\mathcal{D}$. Next, we perform a polynomial regression of order 2 to obtain an $mIoU$ estimate

$$\widehat{mIoU}_{n,\epsilon} = \sum_{k\in\mathcal{K}} \theta_k \cdot PSNR^k_{n,\epsilon}, \quad (9)$$

with regression parameters $\theta_k, k \in \mathcal{K} = \{0, 1, 2\}$. All θ_k are optimized using the mean squared error.

Performance Prediction Evaluation: We evaluate our performance prediction using mainly three metrics: Pearson correlation, mean absolute error, and mean root squared error [24, 25]. Considering a dataset $\mathcal{D}$, we generate values $a_{n,\epsilon} = mIoU_{n,\epsilon}$ and $b_{n,\epsilon} = PSNR_{n,\epsilon}$, $n \in \mathcal{N}$. Then, the Pearson correlation is computed via

$$\rho = \frac{\sum_{n,\epsilon}(a_{n,\epsilon}-\mu_a)(b_{n,\epsilon}-\mu_b)}{\sqrt{\sum_{n,\epsilon}(a_{n,\epsilon}-\mu_a)^2}\sqrt{\sum_{n,\epsilon}(b_{n,\epsilon}-\mu_b)^2}} \quad (10)$$

with $\mu_a = \frac{1}{|\mathcal{N}||\mathcal{E}|}\sum_{n,\epsilon} a_{n,\epsilon}$, $\mu_b = \frac{1}{|\mathcal{N}||\mathcal{E}|}\sum_{n,\epsilon} b_{n,\epsilon}$, $\epsilon \in \mathcal{E}$, set of distortion strengths $\mathcal{E}$, and $\rho \in [-1, 1]$. The special cases $\rho \in \{0, -1, 1\}$ indicate 'no correlation', 'perfect negative correlation', and 'perfect positive correlation', respectively. In addition, we define the Pearson correlation *under distortion strength* ϵ as

$$\rho_{\epsilon} = \frac{\sum_{n}(a_{n,\epsilon}-\mu_{a,\epsilon})(b_{n,\epsilon}-\mu_{b,\epsilon})}{\sqrt{\sum_{n}(a_{n,\epsilon}-\mu_{a,\epsilon})^2}\sqrt{\sum_{n}(b_{n,\epsilon}-\mu_{b,\epsilon})^2}}, \quad (11)$$

with $\mu_{a,\epsilon} = \frac{1}{|\mathcal{N}|}\sum_{n} a_{n,\epsilon}$, $\mu_{b,\epsilon} = \frac{1}{|\mathcal{N}|}\sum_{n} b_{n,\epsilon}$.

For both error metrics, we first measure $\Delta mIoU_{n,\epsilon} = \widehat{mIoU}_{n,\epsilon} - mIoU_{n,\epsilon}$ over dataset $\mathcal{D}$. Afterwards, we compute the mean absolute error

$$\Delta mIoU^{\text{M}} = \frac{1}{|\mathcal{N}||\mathcal{E}|}\sum_{n\in\mathcal{N}}\sum_{\epsilon\in\mathcal{E}} \left|\Delta mIoU_{n,\epsilon}\right| \quad (12)$$

and the root mean squared error

$$\Delta mIoU^{\text{R}} = \sqrt{\frac{1}{|\mathcal{N}||\mathcal{E}|}\sum_{n\in\mathcal{N}}\sum_{\epsilon\in\mathcal{E}} \left(\Delta mIoU_{n,\epsilon}\right)^2} \quad (13)$$

of the predicted performance.

4. Experimental Setup

In the following, we introduce our employed datasets and network architectures, as well as training and evaluation details. All experiments are performed using `PyTorch` [41] and a single `NVIDIA GTX 1080Ti`.

Table 1. Datasets and subsets we used in our experiments.

Dataset	Official Subset	# Images	Math. Symbol
Cityscapes [8] (CS)	train	2,975	$\mathcal{D}^{\text{CS}}_{\text{train}}$
	val	59	$\mathcal{D}^{\text{CS}}_{\text{val}}$
		441	$\mathcal{D}^{\text{CS}}_{\text{test}}$
Kitti [11] (KIT)	train	50	$\mathcal{D}^{\text{KIT}}_{\text{val}}$
		150	$\mathcal{D}^{\text{KIT}}_{\text{test}}$

Employed Datasets: As shown in Tab. 1, we train our models on the Cityscapes [8] training set ($\mathcal{D}^{\text{CS}}_{\text{train}}$) and report results on both Cityscapes validation set and KITTI 2015 Stereo [11] training set. We further split the Cityscapes validation set into a validation ($\mathcal{D}^{\text{CS}}_{\text{val}}$, road scenes captured in Lindau, Germany) and test ($\mathcal{D}^{\text{CS}}_{\text{test}}$, road scenes captured in Frankurt/Münster, Germany) subsplit [2,3]. We do the same for the KITTI training set, this time in an alpha-numerical fashion, yielding $\mathcal{D}^{\text{KIT}}_{\text{val}}$ and $\mathcal{D}^{\text{KIT}}_{\text{test}}$ (as used in [24, 25]).

Network Architecture: We deploy `SwiftNet18` [40] as semantic segmentation architecture, using a `ResNet18` [17] encoder with the `SwiftNet` segmentation decoder [40], consisting of a spatial pyramid pooling module [16] and subsequent upsampling modules with lateral encoder-decoder connections. This results in 11.8M parameters in total. Further, we use nearest neighbor upsampling during training and bilinear upsampling during inference supporting reproducibility. For more details on the `SwiftNet18`, we point the interested reader to [40].

For the image reconstruction decoder, we follow our two design options introduced in Section 3.2. For the first design option, we deploy `SwiftNet` segmentation decoder variants [40] with or without lateral encoder-decoder connections or spatial pyramid pooling. For the second design option, we deploy a mirrored `ResNet18` attached to the encoder output. We experiment with reducing and increasing the number of residual units (RUs) per block, yielding a `ResNet10` decoder (one RU per block), and a `ResNet26` (three RUs per block) decoder. We also deploy a mirrored `ResNet18` with lateral encoder-decoder connections, which we dub `ResNet18L`. Here, we simply add the pre-activated outputs of the encoder residual blocks [40] to the inputs of respective decoder residual blocks.

Training Details: We train the `SwiftNet18` for 200 epochs on $\mathcal{D}^{\text{CS}}_{\text{train}}$ with a batch size of 12. For the ImageNet-pretrained `ResNet18` and the randomly initialized `SwiftNet` segmentation decoder, the model parameters are optimized using the Adam optimizer [22] with a weight decay of $2.5 \cdot 10^{-5}$ and 10^{-4}, respectively, and a cosine annealing learning rate [33], starting at 10^{-4} and $4 \cdot 10^{-4}$, respectively, and finishing at 10^{-7} and 10^{-6}, respectively. After training the `SwiftNet18`, all parameters are fixed and the image reconstruction decoder with randomly initialized weights is trained for further 10 epochs with a batch size of 8. Here, we apply the same optimizer and learning rate settings as for the segmentation decoder.

Table 2. $mIoU_{\epsilon=0}$ [%] (4), $\overline{mIoU}_{\epsilon=0}$ [%] (6), and $\overline{PSNR}_{\epsilon=0}$ [dB] (8) of `SwiftNet18` and attached `SwiftNet`-based, `ResNet18`-based, or `ResNet18L`-based reconstruction decoder reported under clean conditions ($\epsilon = 0$) on $\mathcal{D}^{\text{CS}}_{\text{val}}$, $\mathcal{D}^{\text{CS}}_{\text{test}}$, $\mathcal{D}^{\text{KIT}}_{\text{val}}$, and $\mathcal{D}^{\text{KIT}}_{\text{test}}$. Note that only $\mathcal{D}^{\text{CS}}_{\text{train}}$ is used as training set. Best image reconstruction results in boldface.

Metric	Rec. Dec.	$\mathcal{D}^{\text{CS}}_{\text{val}}$	$\mathcal{D}^{\text{CS}}_{\text{test}}$	$\mathcal{D}^{\text{KIT}}_{\text{val}}$	$\mathcal{D}^{\text{KIT}}_{\text{test}}$
$mIoU_{\epsilon=0}$	-	65.02	72.95	43.12	39.46
$\overline{mIoU}_{\epsilon=0}$	-	49.37	61.41	36.04	34.18
$\overline{PSNR}_{\epsilon=0}$	`SwiftNet`	29.86	29.39	20.31	20.60
	`ResNet18`	21.52	20.64	13.90	13.87
	`ResNet18L`	**31.80**	**31.42**	**20.69**	**21.26**

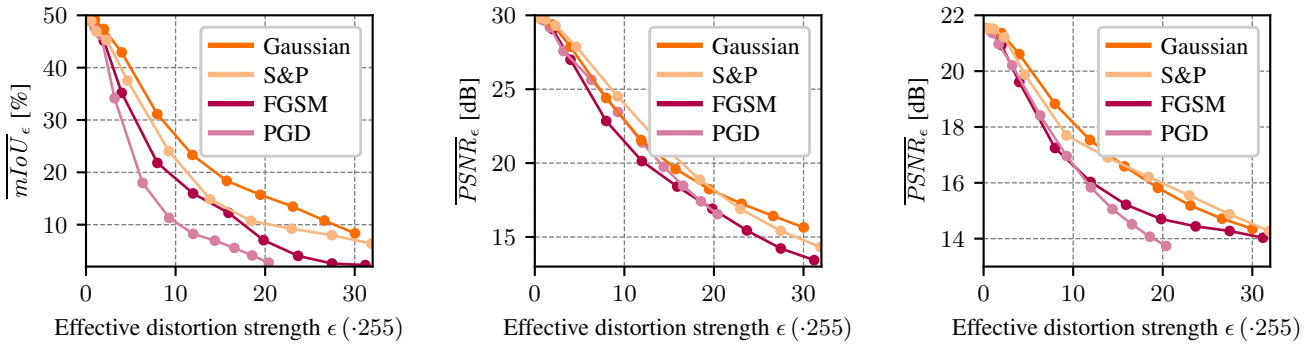
Evaluation & Regression Details: We evaluate our method on clean and distorted versions of $\mathcal{D}^{\text{CS}}_{\text{val}}$, $\mathcal{D}^{\text{CS}}_{\text{test}}$, $\mathcal{D}^{\text{KIT}}_{\text{val}}$, and $\mathcal{D}^{\text{KIT}}_{\text{test}}$, mainly reporting the Pearson correlation (10) of $mIoU_{n,\epsilon}$ (5) and $PSNR_{n,\epsilon}$ (7). The regression is calibrated with clean and distorted versions of $\mathcal{D}^{\text{CS}}_{\text{val}}$ or $\mathcal{D}^{\text{KIT}}_{\text{val}}$, using (9). The clean conditions are simulated using the original data. The distorted conditions are simulated using either Gaussian noise, salt-and-pepper noise, FGSM [12], or PGD [35] (40 iterations with step size of $\frac{2}{255}$) applied with various target distortion strengths $\bar{\epsilon} \in \overline{\mathcal{E}} = \{0.25, 0.5, 1, 2, 4, 8, 12, 16, 20, 24, 28, 32\} \cdot \frac{1}{255}$ following [24,25]. FGSM and PGD are optimized to maximize (1). In total we create $4 \cdot |\overline{\mathcal{E}}| = 48$ distorted datasets in addition to their clean origin. More details about the distortion settings can be found in the supplementary material.

5. Experimental Results

A high correlation ρ (10) is an indicator for a reliable regression and thus performance prediction. Therefore, in the following experimental evaluations, we will concentrate on ρ (10) rather than on predicting $\widehat{mIoU}_{n,\epsilon}$ (9) in the first place. Finally, in the last paragraph, we will also discuss the quality of $\widehat{mIoU}_{n,\epsilon}$ in terms of $\Delta mIoU^{\text{M}}$ (12) and $\Delta mIoU^{\text{R}}$ (13) and compare our approach to the closest prior art [24, 25].

5.1. First Analysis on Task Performances

Clean Data: The focus of this work is performance prediction. However, for completeness, we also provide results regarding actual performance and discuss them briefly. We first take a look at the clean performance on $\mathcal{D}^{\text{CS}}_{\text{val}}$, $\mathcal{D}^{\text{CS}}_{\text{test}}$, $\mathcal{D}^{\text{KIT}}_{\text{val}}$, and $\mathcal{D}^{\text{KIT}}_{\text{test}}$ of a `SwiftNet18` with either a plain `SwiftNet`-based or `ResNet18`-based reconstruction de-



(a) `SwiftNet18`-based segmentation (b) `SwiftNet`-based reconstruction (c) `ResNet18`-based reconstruction

Figure 4. $\overline{mIoU}_\epsilon$ (6) [%] and $\overline{PSNR}_\epsilon$ (8) [dB] under various ϵ (3) on $\mathcal{D}_{\text{val}}^{\text{CS}}$. Note, each curve point also represents a distinct $\bar{\epsilon} \in \bar{\mathcal{E}}$. Results are reported for (a) `SwiftNet18`-based segmentation, (b) `SwiftNet`-based reconstruction, and (c) `ResNet18`-based reconstruction.

Table 3. Correlation ρ (10) of `SwiftNet18` segmentation and various attached `SwiftNet`-based reconstruction decoder variants. Results are reported under various input conditions on $\mathcal{D}_{\text{val}}^{\text{CS}}$ and on $\mathcal{D}_{\text{val}}^{\text{KIT}}$. Note that only $\mathcal{D}_{\text{train}}^{\text{CS}}$ is used as training set. 'Lat.' refers to lateral skip connections from encoder to decoder [40], 'SPP' refers to spatial pyramid pooling [16, 40], and 'all' refers to all data used for ρ computation. Best variant on each dataset in bold, second best underlined.

Eval on	Lat.	SPP	Clean	FGSM	PGD	Gaussian	S&P	all
$\mathcal{D}_{\text{val}}^{\text{CS}}$			0.19	0.77	0.78	0.70	0.78	0.76
		✓	**0.20**	0.77	0.79	0.70	0.77	0.76
	✓		-0.01	**0.88**	**0.85**	**0.83**	**0.86**	**0.85**
	✓	✓	-0.01	**0.88**	0.84	**0.83**	0.85	0.84
$\mathcal{D}_{\text{val}}^{\text{KIT}}$			-0.29	0.68	0.72	0.46	0.64	0.65
		✓	-0.38	0.62	0.65	0.35	0.56	0.58
	✓		-0.42	**0.79**	**0.77**	**0.67**	0.79	**0.75**
	✓	✓	**-0.43**	**0.79**	0.76	0.66	**0.81**	**0.75**

coder. Results in terms of $mIoU_{\epsilon=0}$ (4), $\overline{mIoU}_{\epsilon=0}$ (6), and $\overline{PSNR}_{\epsilon=0}$ (8) are listed in Tab. 2. In terms of $\overline{PSNR}_{\epsilon=0}$, the `SwiftNet`-based reconstruction decoder (design option 1) is superior to the `ResNet18`-based reconstruction decoder (design option 2). As a first hypothesis, we explain this by the additional lateral encoder-decoder connections in a `SwiftNet`-based reconstruction decoder, which can be expected to support the task of image reconstruction. This is confirmed by showing improved results with `ResNet18L`—a `ResNet18`-based reconstruction decoder with additional lateral encoder-decoder connections.

Distorted Data: Next, we take a closer look at $\overline{mIoU}_\epsilon$ (6) and $\overline{PSNR}_\epsilon$ (8) on $\mathcal{D}_{\text{val}}^{\text{CS}}$ under various input distortions and (mean) effective distortion strengths ϵ (3). We again use a `SwiftNet18` with either a `SwiftNet`-based or `ResNet18`-based reconstruction decoder. The results are depicted in Fig. 4, with $\overline{mIoU}_\epsilon$ being plotted in Fig. 4a and $\overline{PSNR}_\epsilon$ being plotted in Figs. 4b and 4c for `SwiftNet`-based and `ResNet18`-based reconstruction, respectively. We observe that both $\overline{mIoU}_\epsilon$ (Fig. 4a) and $\overline{PSNR}_\epsilon$ (Figs. 4b and 4c) drop with increasing ϵ. This indicates an existing correlation between $\overline{mIoU}_\epsilon$, $mIoU_{n,\epsilon}$ and $\overline{PSNR}_\epsilon$, $PSNR_{n,\epsilon}$ and that this correlation is present regardless of the underlying image reconstruction architecture. Further, the $\overline{mIoU}_\epsilon$ curves (cf. Fig. 4a) are more spread than the $\overline{PSNR}_\epsilon$ ones (cf. Figs. 4b and 4c), the latter even showing intersections. Accordingly, we conclude $\overline{mIoU}_\epsilon$ heavily depends on both distortion type *and* ϵ, while $\overline{PSNR}_\epsilon$ more or less only depends on ϵ. Next, we will analyze, if one of our two design options is superior in terms of ρ (10).

5.2. Pearson Correlation, Architectural Aspects

As concerns Pearson correlation ρ (10), we first investigate different architectural aspects of the reconstruction decoder. Remember that we aim at a high ρ such that we can estimate $mIoU_{n,\epsilon}$ from $PSNR_{n,\epsilon}$. For this analysis, we deploy `SwiftNet` decoder variants and `ResNet` decoder variants. We first take a look at the `SwiftNet` decoder on $\mathcal{D}_{\text{val}}^{\text{CS}}$ (cf. Tab. 3, upper half). It can be observed that incorporating lateral encoder-decoder connections improves the ρ (cf. Tab. 3, Lat., rows 1 & 2 vs. rows 3 & 4), while the effect of spatial pyramid pooling is negligible (cf. Tab. 3, SPP, rows 2 & 4 vs. rows 1 & 3). Further, while lateral encoder-decoder connections improve ρ on distorted images (cf. Tab. 3, e.g., FGSM, rows 1 & 2 vs. rows 3 & 4), they completely eliminate a correlation on clean images (cf. Tab. 3, Clean, rows 1 & 2 vs. rows 3 & 4). This, however, does not pose a problem as in a practical use case, we could think of a high threshold θ and discard $\widehat{mIoU}_{n,\epsilon}$ for $PSNR_{n,\epsilon} > \theta$ as on average with high $PSNR_{n,\epsilon}$ we expect high $mIoU_{n,\epsilon}$ (cf. Figs. 4a and 4b). In numbers, our best `SwiftNet` decoder model achieves

Table 4. Correlation ρ (10) of SwiftNet18 segmentation and various attached ResNet-based (RN) and the best SwiftNet-based (SN) reconstruction decoders from Tab. 3. Results are reported under various input conditions on $\mathcal{D}_{\text{val}}^{\text{CS}}$ and on $\mathcal{D}_{\text{val}}^{\text{KIT}}$. Note that only $\mathcal{D}_{\text{train}}^{\text{CS}}$ is used as training set. 'Block c.' refers to the number of residual units in the first up to the fourth ResNet block [17] and 'all' refers to all data used for ρ computation. Best variant on each dataset in bold, second best underlined.

Eval on	Dec.	Block c.	Clean	FGSM	PGD	Gauss.	S&P	all
$\mathcal{D}_{\text{val}}^{\text{CS}}$	SN	-	-0.01	**0.88**	**0.85**	**0.83**	**0.86**	**0.85**
	RN10	1-1-1-1	**0.24**	0.82	0.82	0.76	0.84	0.81
	RN18	2-2-2-2	0.19	0.83	0.82	0.77	0.82	0.81
	RN26	3-3-3-3	0.21	0.84	0.83	0.77	0.82	0.82
	RN18L	2-2-2-2	0.08	**0.88**	**0.85**	**0.83**	**0.86**	**0.85**
$\mathcal{D}_{\text{val}}^{\text{KIT}}$	SN	-	**-0.42**	**0.79**	**0.77**	**0.67**	**0.79**	**0.75**
	RN10	1-1-1-1	-0.35	0.67	0.72	0.48	0.64	0.65
	RN18	2-2-2-2	-0.35	0.69	0.74	0.50	0.64	0.67
	RN26	3-3-3-3	-0.37	0.70	0.74	0.50	0.64	0.67
	RN18L	2-2-2-2	**-0.42**	0.77	**0.77**	0.65	**0.79**	**0.75**

$\rho = 0.85$ under all distortions and various ϵ by only using lateral encoder-decoder connections (cf. Tab. 3, row 3). Next, we take a look at the ResNet (RN) decoder variants on $\mathcal{D}_{\text{val}}^{\text{CS}}$ (cf. Tab. 4, upper half) and compare them to the best SwiftNet (SN) decoder model from Tab. 3. First, one can see that the SwiftNet decoder model is superior to the plain ResNet18 (RN18) decoder model in terms of ρ on distorted images. Second, reducing (cf. Tab. 4, RN10) or increasing (cf. Tab. 4, RN26) the ResNet decoder model parameters does not really have an effect on ρ. Third, the correlation on clean images of the plain ResNet decoder models (RN10, RN18, RN26) is better. We hypothesize that this comes from missing lateral encoder-decoder connections as we observe comparable effects with similar SwiftNet decoder variants (cf. Tab. 3, rows 1 & 2 vs. rows 3 & 4). This is confirmed by the results shown for ResNet18L (RN18L), where we improve upon correlation on distorted data, while worsen upon correlation on clean data. Further, this variant performs on par with the best SwiftNet-based reconstruction decoder. We finally conclude that a SwiftNet18 equipped with either a SwiftNet-based or a ResNet18-based reconstruction decoder, *both with lateral encoder-decoder connections*, yield the best results. However, while the ResNet18-based reconstruction decoder has about 11.2M parameters, the SwiftNet-based one is more efficient with only about 0.95M parameters. Therefore, we will focus the regression analysis on the SwiftNet-based reconstruction decoder.

5.3. Dataset Transferability

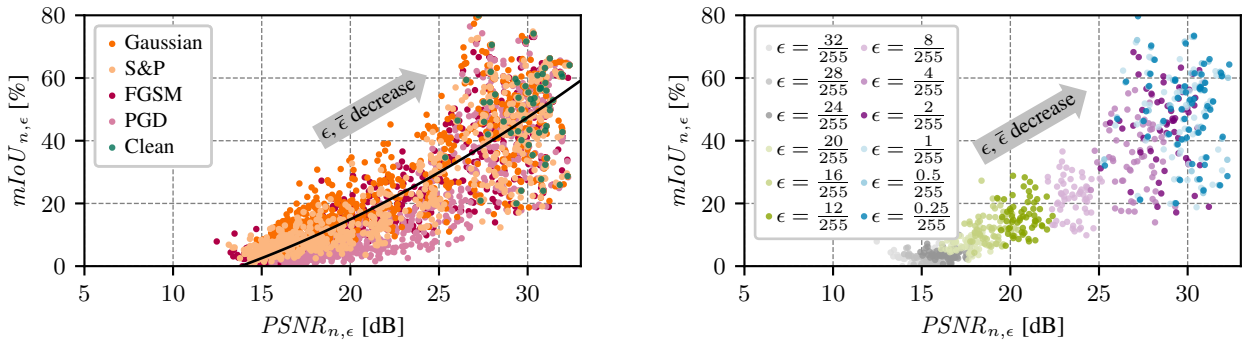
The following experiment series elaborates on the dataset transferability of our approach. For this, we test the models trained on Cityscapes on the KITTI dataset. We first take a look at experiments with the SwiftNet-based decoder on $\mathcal{D}_{\text{val}}^{\text{KIT}}$ (Tab. 3, lower half), where we make several observations: First, ρ drops noticeably. Second, models with no lateral encoder-decoder connections have significantly lower ρ for Gaussian noise. The strongest drop is observable with the model which only incorporates spatial pyramid pooling. Last, we observe higher (absolute) but negative ρ on clean data. Similar observations are made for the ResNet decoder experiments $\mathcal{D}_{\text{val}}^{\text{KIT}}$ (Tab. 4, lower half).

5.4. Regression Analysis

To predict the mean intersection over union we need a regression model. For this purpose, we take one of our best models, i.e, SwiftNet18 with a SwiftNet-based reconstruction decoder with lateral encoder-decoder connections but without spatial pyramid pooling, and perform a regression analysis on $\mathcal{D}_{\text{val}}^{\text{CS}}$ according to Section 3.3. We choose the best SwiftNet-based reconstruction decoder as it performs on par with the best ResNet-based one (cf. Tab. 3, SN vs. RN18L), while being less complex in terms of model parameters (0.95M vs. 11.17M). Respective plots can be seen in Fig. 5. Here, Fig. 5a depicts scatter plots of all distortions and various ϵ (3), $\bar{\epsilon} \in \bar{\mathcal{E}}$. In addition a respective polynomial regression curve of order 2 is drawn. Fig. 5b shows scatter plots for FGSM with a closer look at ϵ. Note that for FGSM we can assume $\epsilon \mathrel{\widehat{=}} \bar{\epsilon}$. Considering Fig. 5a, one can observe that the regression error is visibly higher with high model performance (upper right) and decreases with low model performance (lower left). In addition, the low correlation on clean images from Tabs. 3 and 4 can be qualitatively confirmed. Similar observations were made in [25]. Taking a closer look at the FGSM scatter plot in Fig. 5b, we observe the following: The correlation ρ_ϵ (11) is rather low for individual distortion strengths, e.g., $\epsilon = \frac{2}{255}$ or $\epsilon = \frac{12}{255}$. We observed similar effects for the other distortions. We conclude, the high ρ (10) we observe in Tabs. 3 and 4 is caused only by the effect that an increasing ϵ (3) leads to decreasing $mIoU_{n,\epsilon}$ (5), $PSNR_{n,\epsilon}$ (7). This complements the observations and conclusion derived from Fig. 4, where we looked at mean values of $mIoU_{n,\epsilon}$ and $PSNR_{n,\epsilon}$ that motivated our further analysis.

5.5. State of the Art Comparison

Next, we use the calibrated regression on $\mathcal{D}_{\text{val}}^{\text{CS}}$ to predict $mIoU_{n,\epsilon}$ on $\mathcal{D}_{\text{test}}^{\text{KIT}}$ as well as $\mathcal{D}_{\text{test}}^{\text{CS}}$. We further use $\mathcal{D}_{\text{val}}^{\text{KIT}}$ for calibration to see its effect on testing against $\mathcal{D}_{\text{test}}^{\text{KIT}}$. Results and comparisons to state of the art [24, 25] are reported in Tab. 5 in terms of ρ (9), $\Delta mIoU^{\text{M}}$ (12), and



(a) All distortions

(b) FGSM only

Figure 5. Scatter plots of $mIoU_{n,\epsilon}$ (5) and $PSNR_{n,\epsilon}$ (7) on clean and distorted images from $\mathcal{D}^{\text{CS}}_{\text{val}}$, with $\bar{\epsilon} \in \bar{\mathcal{E}}$ as well as various ϵ (3). (a) FGSM, PGD, Gaussian noise, salt-and-pepper (S&P) noise, and clean, with a respective polynomial regression curve of second order. (b) Isolated look onto FGSM. Note that for FGSM we can assume $\epsilon \mathrel{\widehat{=}} \bar{\epsilon}$ (see supplementary material).

Table 5. Metrics ρ (10), $\Delta mIoU^{\text{M}}$ (12), and $\Delta mIoU^{\text{R}}$ (13) for our method and comparable variants from [24, 25]. Our approaches use $\mathcal{D}^{\text{CS}}_{\text{train}}$ as training set, while [24, 25] also use video datasets $\mathcal{D}^{\text{CS}}_{\text{vid}}$, $\mathcal{D}^{\text{KIT}}_{\text{vid}}$. Results are reported under a mix of input conditions either on $\mathcal{D}^{\text{CS}}_{\text{test}}$ (as [24] uses a variant of $\mathcal{D}^{\text{CS}}_{\text{test}}$, we mark the entries with *) or $\mathcal{D}^{\text{KIT}}_{\text{test}}$. 'Cal.' refers to regression calibration.

Eval on	Video	Cal. on	Method	ρ	$\Delta mIoU^{\text{M}}$	$\Delta mIoU^{\text{R}}$
$\mathcal{D}^{\text{CS}}_{\text{test}}$	-	$\mathcal{D}^{\text{CS}}_{\text{val}}$	Ours	**0.90**	**10.12**	**13.18**
	$\mathcal{D}^{\text{CS}}_{\text{vid}}$	$\mathcal{D}^{\text{CS}}_{\text{val}}$	[24]	0.58*	12.19*	15.71*
	$\mathcal{D}^{\text{KIT}}_{\text{vid}}$	$\mathcal{D}^{\text{CS}}_{\text{val}}$	[24]	0.43*	13.38*	16.12*
$\mathcal{D}^{\text{KIT}}_{\text{test}}$	-	$\mathcal{D}^{\text{CS}}_{\text{val}}$	Ours	0.73	11.62	14.26
	-	$\mathcal{D}^{\text{KIT}}_{\text{val}}$	Ours	0.73	8.00	10.24
	$\mathcal{D}^{\text{CS}}_{\text{vid}}$	$\mathcal{D}^{\text{KIT}}_{\text{val}}$	[24]	0.54	7.81	9.79
	$\mathcal{D}^{\text{KIT}}_{\text{vid}}$	$\mathcal{D}^{\text{KIT}}_{\text{val}}$	[24]	0.77	6.01	7.70
	$\mathcal{D}^{\text{KIT}}_{\text{vid}}$	$\mathcal{D}^{\text{KIT}}_{\text{val}}$	[25]	**0.86**	**4.45**	**6.16**

$\Delta mIoU^{\text{M}}$ (13). Note that our approach is solely trained on $\mathcal{D}^{\text{CS}}_{\text{train}}$ while [24, 25] need additional video data from either Cityscapes $\mathcal{D}^{\text{CS}}_{\text{vid}}$ or KITTI $\mathcal{D}^{\text{KIT}}_{\text{vid}}$. When it comes to Cityscapes $\mathcal{D}^{\text{CS}}_{\text{test}}$ (Tab. 5, upper half), we clearly excel state of the art in all categories. However, we are not able to excel state of the art on KITTI $\mathcal{D}^{\text{KIT}}_{\text{test}}$ (Tab. 5, lower half). We hypothesize that this is due to the domain shift from Cityscapes to KITTI. This claim is supported by [24], where we see that through the inclusion of KITTI, the performance is improved significantly (entry [24] ($\mathcal{D}^{\text{CS}}_{\text{vid}}$) vs. [24] ($\mathcal{D}^{\text{KIT}}_{\text{vid}}$) in the lower half of Tab. 5).

Through our experiments we could not only provide a proof of concept, but also set a new state-of-the-art benchmark both on Cityscapes and KITTI for image-only input methods for segmentation performance prediction. Moreover, we even excel a LiDAR-supported benchmark [24] on Cityscapes. All this is achieved without additional video data for training ($\mathcal{D}^{\text{CS}}_{\text{vid}}$, $\mathcal{D}^{\text{KIT}}_{\text{vid}}$) or LiDAR data during inference, both, however, needed in [24, 25]. We train our image reconstruction solely on $\mathcal{D}^{\text{CS}}_{\text{train}}$ and only need the input image to provide a performance estimate $\widehat{mIoU}_{n,\epsilon}$ (9).

6. Conclusions

In this paper, we proposed a performance prediction for semantic segmentation by using a self-supervised image reconstruction decoder. Our proposed method is efficient as it does not rely on additional sensors, additional training data, or retraining of the semantic segmentation. We set a new state-of-the-art benchmark both on KITTI and Cityscapes for image-only input methods, excelling even a LiDAR-supported benchmark on Cityscapes. We believe that our proposed method facilitates further research and increases the awareness for safety in neural network-based highly automated driving.

Acknowledgment

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Climate Action" within the project "Methoden und Maßnahmen zur Absicherung von KI basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI-Absicherung)". The authors would like to thank the consortium for the successful cooperation. Further, the authors gratefully acknowledge support of this work by Johannes Meyer, Vijesh Vidhani, and CARIAD SE, Wolfsburg, Germany.

References

[1] Anurag Arnab, Ondrej Miksik, and Philip H. S. Torr. On the Robustness of Semantic Segmentation Models to Adversarial Attacks. In *Proc. of CVPR*, pages 888–897, Salt Lake City, UT, USA, June 2018. 1, 2

[2] Andreas Bär, Fabian Hüger, Peter Schlicht, and Tim Fingscheidt. On the Robustness of Redundant Teacher-Student Frameworks for Semantic Segmentation. In *Proc. of CVPR - Workshops*, pages 1380–1388, Long Beach, CA, USA, June 2019. 5

[3] Andreas Bär, Marvin Klingner, Serin Varghese, Fabian Hüger, Peter Schlicht, and Tim Fingscheidt. Robust Semantic Segmentation by Redundant Networks With a Layer-Specific Loss Contribution and Majority Vote. In *Proc. of CVPR - Workshops*, pages 1348–1358, Seattle, WA, USA, June 2020. 1, 2, 5

[4] Andreas Bär, Jonas Löhdefink, Nikhil Kapoor, Serin John Varghese, Fabian Hüger, Peter Schlicht, and Tim Fingscheidt. The Vulnerability of Semantic Segmentation Networks to Adversarial Attacks in Autonomous Driving: Enhancing Extensive Environment Sensing. *IEEE Signal Processing Magazine*, 38(1):42–52, Jan. 2021. 1, 2

[5] Jan-Aike Bolte, Markus Kamp, Antonia Breuer, Silviu Homoceanu, Peter Schlicht, Fabian Hüger, Daniel Lipinski, and Tim Fingscheidt. Unsupervised Domain Adaptation to Improve Image Segmentation Quality Both in the Source and Target Domain. In *Proc. of CVPR - Workshops*, pages 1404–1413, Long Beach, CA, USA, June 2019. 1

[6] Alvin Chan, Yi Tay, and Yew-Soon Ong. What It Thinks Is Important Is Important: Robustness Transfers Through Input Gradients. In *Proc. of CVPR*, pages 332–341, Seattle, WA, USA, June 2020. 2

[7] Seungju Cho, Tae Joon Jun, Byungsoo Oh, and Daeyoung Kim. DAPAS : Denoising Autoencoder to Prevent Adversarial attack in Semantic Segmentation. In *Proc. of IJCNN*, pages 1–8, Glasgow, UK, July 2020. 2

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of CVPR*, pages 3213–3223, Las Vegas, NV, USA, June 2016. 2, 5

[9] Gianni Franchi, Nacim Belkhir, Mai Lan Ha, Yufei Hu, Andrei Bursuc, Volker Blanz, and Angela Yao. Robust Semantic Segmentation with Superpixel-Mix. In *Proc. of BMVC*, pages 1–16, Virtual Conference, Nov. 2021. 2

[10] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proc. of ICML*, pages 1050–1059, New York, NY, USA, June 2016. 2

[11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision Meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, Aug. 2013. 2, 5

[12] 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. 1, 2, 5

[13] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering Adversarial Images using Input Transformations. In *Proc. of ICLR*, pages 1–12, Vancouver, BC, Canada, Apr. 2018. 2

[14] Frederik K. Gustafsson, Martin Danelljan, and Thomas B. Schön. Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision. In *Proc. of CVPR - Workshops*, pages 1289–1298, Seattle, WA, USA, June 2020. 2

[15] Atiye Sadat Hashemi, Andreas Bär, Saeed Mozaffari, and Tim Fingscheidt. Transferable Universal Adversarial Perturbations Using Generative Models. *arXiv*, 2010.14919:1–9, Oct. 2020. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(9):1904–1916, Sept. 2015. 5, 6

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*, pages 770–778, Las Vegas, NV, USA, June 2016. 5, 7

[18] Md Amirul Islam, Matthew Kowal, Konstantinos G. Derpanis, and Neil D. B. Bruce. Feature Binding with Category-Dependant MixUp for Semantic Segmentation and Adversarial Robustness. In *Proc. of BMVC*, pages 1–13, Virtual Conference, Sept. 2020. 2

[19] Christoph Kamann, Burkhard Güssefeld, Robin Hutmacher, Jan Hendrik Metzen, and Carsten Rother. Increasing the Robustness of Semantic Segmentation Models with Painting-by-Numbers. In *Proc. of ECCV*, pages 369–387, Glasgow, UK, Aug. 2020. 2

[20] Christoph Kamann and Carsten Rother. Benchmarking the Robustness of Semantic Segmentation Models. In *Proc. of CVPR*, pages 8828–8838, Virtual Conference, June 2020. 1, 2

[21] Nikhil Kapoor, Andreas Bär, Serin Varghese, Jan David Schneider, Fabian Hüger, Peter Schlicht, and Tim Fingscheidt. From a Fourier-Domain Perspective on Adversarial Examples to a Wiener Filter Defense for Semantic Segmentation. In *Proc. of IJCNN*, pages 1–8, Virtual Conference, July 2021. 1, 2

[22] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*, pages 1–15, San Diego, CA, USA, May 2015. 5

[23] Marvin Klingner, Andreas Bär, and Tim Fingscheidt. Improved Noise and Attack Robustness for Semantic Segmentation by Using Multi-Task Training with Self-Supervised Depth Estimation. In *Proc. of CVPR - Workshops*, pages 1299–1309, Seattle, WA, USA, June 2020. 1, 2

[24] Marvin Klingner, Andreas Bär, Marcel Mross, and Tim Fingscheidt. Improving Online Performance Prediction for Semantic Segmentation. In *Proc. of CVPR - Workshops*, pages 1–11, Virtual Conference, June 2021. 1, 2, 3, 4, 5, 7, 8

[25] 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, 2, 3, 4, 5, 7, 8

[26] Marvin Klingner, Varun Ravi Kumar, Senthil Yogamani, Andreas Bär, and Tim Fingscheidt. Detecting Adversarial Perturbations in Multi-Task Perception. *arXiv*, 2203.01177:1–9, Mar. 2022. 2

[27] Marvin Klingner, Jan-Aike Termöhlen, Jonas Mikolajczyk, and Tim Fingscheidt. Self-Supervised Monocular Depth Estimation: Solving the Dynamic Object Problem by Semantic Guidance. In *Proc. of ECCV*, pages 582–600, Glasgow, UK, Aug. 2020. 3

[28] Marvin Klingner, Jan-Aike Termöhlen, Jacob Ritterbach, and Tim Fingscheidt. Unsupervised BatchNorm Adaptation (UBNA): A Domain Adaptation Method for Semantic Segmentation Without Using Source Domain Representations. In *Proc. of WACV - Workshops*, pages 210–220, Waikoloa, HI, USA, Jan. 2022. 1

[29] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Proc. of NIPS*, pages 6402–6413, Long Beach, CA, USA, Dec. 2017. 2

[30] Jiayang Liu, Weiming Zhang, Yiwei Zhang, Dongdong Hou, Yujia Liu, Hongyue Zha, and Nenghai Yu. Detection Based Defense Against Adversarial Examples from the Steganalysis Point of View. In *Proc. of CVPR*, pages 4825–4834, Long Beach, CA, USA, June 2019. 2

[31] Zihao Liu, Qi Liu, Tao Liu, Nuo Xu, Xue Lin, Yanzhi Wang, and Wujie Wen. Feature Distillation: DNN-Oriented JPEG Compression Against Adversarial Examples. In *Proc. of CVPR*, pages 860–868, Long Beach, CA, USA, June 2019. 2

[32] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proc. of CVPR*, pages 3431–3440, Boston, MA, USA, June 2015. 2

[33] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *Proc. of ICLR*, pages 1–16, Toulon, France, Apr. 2017. 5

[34] Jonas Löhdefink, Justin Fehrling, Marvin Klingner, Fabian Hüger, Peter Schlicht, Nico M. Schmidt, and Tim Fingscheidt. Self-Supervised Domain Mismatch Estimation for Autonomous Perception. In *Proc. of CVPR - Workshops*, pages 1359–1368, Seattle, WA, USA, June 2020. 1, 2

[35] 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, 5

[36] Chengzhi Mao, Amogh Gupta, Vikram Nitin, Baishakhi Ray, Shuran Song, Junfeng Yang, and Carl Vondrick. Multitask Learning Strengthens Adversarial Robustness. In *Proc. of ECCV*, pages 158–174, Virtual Conference, Aug. 2020. 2

[37] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. A Self-Supervised Approach for Adversarial Robustness. In *Proc. of CVPR*, pages 262–271, Seattle, WA, USA, June 2020. 2

[38] Federico Nesti, Giulio Rossolini, Saasha Nair, Alessandro Biondi, and Giorgio Buttazzo. Evaluating the Robustness of Semantic Segmentation for Autonomous Driving Against Real-World Adversarial Patch Attacks. In *Proc. of WACV*, pages 2280–2289, Waikoloa, HI, USA, Jan. 2022. 2

[39] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and Checkerboard Artifacts. *Distill*, pages 1–1, Oct. 2016. 3

[40] Marin Oršić, Ivan Krešo, Petra Bevandić, and Siniša Šegvić. In Defense of Pre-Trained ImageNet Architectures for Real-Time Semantic Segmentation of Road-Driving Images. In *Proc. of CVPR*, pages 12607–12616, Long Beach, CA, USA, June 2019. 1, 2, 5, 6

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proc. of NeurIPS*, pages 8024–8035, Vancouver, BC, Canada, Dec. 2019. 4

[42] Julien Rebut, Andrei Bursuc, and Patrick Pérez. StyleLess Layer: Improving Robustness for Real-World Driving. In *Proc. of IROS*, pages 8992–8999, Prague, Czech Republic, Oct. 2021. 2

[43] Matthias Rottmann, Pascal Colling, Thomas Paul Hack, Robin Chan, Fabian Hüger, Peter Schlicht, and Hanno Gottschalk. Prediction Error Meta Classification in Semantic Segmentation: Detection via Aggregated Dispersion Measures of Softmax Probabilities. In *Proc. of IJCNN*, pages 1–9, Glasgow, UK, July 2020. 2

[44] Evgenia Rusak, Lukas Schott, Roland S. Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. A Simple Way to Make Neural NetworksRobust Against Diverse ImageCorruptions. In *Proc. of ECCV*, pages 53–69, Glasgow, UK, Aug. 2020. 2

[45] Jan-Aike Termöhlen, Marvin Klingner, Leon J. Brettin, Nico M. Schmidt, and Tim Fingscheidt. Continual Unsupervised Domain Adaptation for Semantic Segmentation by Online Frequency Domain Style Transfer. In *Proc. of ITSC*, pages 2881–2888, Virtual Conference, Sept. 2021. 1

[46] Jinyu Tian, Jiantao Zhou, Yuanman Li, and Jia Duan. Detecting Adversarial Examples from Sensitivity Inconsistency of Spatial-Transform Domain. In *Proc. of AAAI*, pages 9877–9885, Virtual Conference, Feb. 2021. 2

[47] Yingda Xia, Yi Zhang, Fengze Liu, Wei Shen, and Alan Yuille. Synthesize Then Compare: Detecting Failures and Anomalies for Semantic Segmentation. In *Proc. of ECCV*, pages 145–161, Glasgow, UK, Aug. 2020. 2

[48] Xiaogang Xu, Hengshuang Zhao, and Jiaya Jia. Dynamic Divide-and-Conquer Adversarial Training for Robust Semantic Segmentation. In *Proc. of ICCV*, pages 7486–7495, Virtual Conference, Oct. 2021. 2