– Supplementary Material – Robust Regression for Monocular Depth Estimation

Julian Lienen JULIAN.LIENEN@UPB.DE
Paderborn University, Germany

Nils Nommensen NILS.NOMMENSEN@TIB.EU
Ralph Ewerth RALPH.EWERTH@TIB.EU
L3S Research Center, Leibniz University Hannover and TIB Hannover, Germany

Eyke Hüllermeier EYKE@IFI.LMU.DE
University of Munich (LMU), Germany

Editors: Vineeth N Balasubramanian and Ivor Tsang

1. Experimental Settings

To demonstrate the effectiveness of robust depth estimation methods, we present results for models trained with a variety of robust loss functions. As most of them involve loss-specific hyperparameters, as well as the optimization algorithm itself, we conducted a hyperparameter optimization for each induced model.

To this end, we employed a random search with 20 trials per run. In each trial, the model was trained using the regarded hyperparameter configuration for 25 epochs with a batch size of 16. When training on a subset of a given data set, we randomly sampled the desired number of instances from the original training split. We used Adam with default parameters as optimizer and optimized the initial learning rate $\eta \in [1e^{-4}, 1e^{-1}]$. As learning rate schedule, we applied cosine annealing (Loshchilov and Hutter, 2017). Moreover, we augmented the training and depth pairs by randomly flipping the images horizontally, augmenting the colors with varying hue, saturation, brightness and contrast, and randomly swapped the red and blue color channels. To select the model for the final assessment, the validation root mean squared error was calculated on *iBims-1* (Koch et al., 2018).

Table 1 enlists the hyperparameters along with their considered search spaces being optimized. We kept the notation the same as in the referred original publications.

Given the finally selected models, we evaluated the depth metrics as described in the paper and further extended in this supplementary material. As *NYUD-v2* (Silberman et al., 2012) and *SunRGBD* (Song et al., 2015) only provide depth values up to 10 m, we only assessed depth values in the ground truth data of the benchmark data sets with up to this value.

We ran all experiments on a modern high-performance cluster with several Nvidia RTX 1080 Ti, 2080 Ti and Titan RTX accelerators. In total, the experimental evaluation took about 4500 GPU hours.

Table 1: Loss-specific hyperparameters and their search spaces being considered within the random search.

Loss	Hyperparameter Space(s)
$\mathcal{L}_{\text{Huber}}$ (Laina et al., 2016)	$c \in [0.1, 0.9]$
$\mathcal{L}_{\text{BerHu}}$ (Laina et al., 2016)	$c \in [0.1, 0.9]$
$\mathcal{L}_{\text{Ruber}}$ (Irie et al., 2019)	$c \in [0.1, 0.9]$
$\mathcal{L}_{\text{Barron}}$ (Barron, 2019)	$\alpha \in [0, 2], c \in [0.1, 50]$
$\mathcal{L}_{\text{trim}}$ (Ranftl et al., 2020)	$U \in [0.1, 0.9]$ {Largest % of residuals being trimmed}
$\mathcal{L}_{\text{ScaledSIError}}$ (Lee et al., 2019)	$\lambda \in [0.1, 0.9], \alpha \in [0.1, 25]$
$\text{OSL}_{\mathcal{L}_1}$	$\epsilon \in [0, 0.25]$
$\text{OSL}_{\mathcal{L}_2}$	$\epsilon \in [0, 0.25]$
$\text{FOSL}_{\mathcal{L}_1}$	$\epsilon \in [0, 0.25], \delta \in [0, 2]$

2. Model Architecture

Fig. 1 shows the EfficientNet-based (Tan and Le, 2019) architecture as being used within our empirical studies. More precisely, we considered a EfficientNetB0 encoder pretrained on ImageNet and freezed the weights during training. While the encoder part comprises approx. 4 million parameters, we effectively maintained 11 million decoder weights, resulting in a total of 15 million parameters. As can be seen in the figure, the decoder part consists of stacked upsamling components applying convolutional, BatchNormalization, ReLU activation and bilinear upsampling layers.

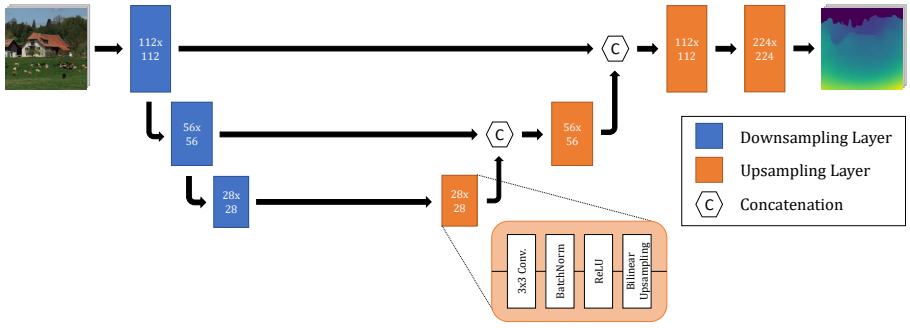


Figure 1: EfficientNet-based U-Net architecture as being used within the empirical evaluation. The blue downsampling layers are specified by the employed backbone. The layer captions denote the corresponding output dimensionality of the respective layers.

3. Additional Experimental Results

3.1. Metrics

Beyond the reported metrics in the paper, we further present results on the following additional metrics.

- Root mean squared error log (RMSLog): $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log y_i - \log \hat{y}_i)^2}$
- Squared relative difference (SQREL): $\frac{1}{n}\sum_{i=1}^{n}\frac{(y_i-\hat{y}_i)^2}{y_i}$

3.2. Homogeneous Depth Sensor: NYUD-v2

Tab. 2 extends Tab. 1 as presented in the main paper by providing results on *DIODE* for the additional metrics, as well as further errors on the Eigen test split of *NYUD-v2*. In accordance to the results presented in the paper, our loss proposals provide the best performance on *DIODE*. In the case of *NYUD-v2*, the scaled SI error turns out to perform best.

Table 2: Averaged results over three runs for additional metrics on *DIODE* and the Eigen test split of *NYUD-v2* for a varying number of instances with the corresponding standard deviations. The best model is indicated in bold per number of instances and metric.

# Insts.	Loss	*DIODE* RMSLog (↓)	*DIODE* SQREL (↓)	*NYUD-v2* $\log_{10}$ (↓)	*NYUD-v2* RMSLog (↓)	*NYUD-v2* SQREL (↓)	*NYUD-v2* δ_2 (↑)	*NYUD-v2* δ_3 (↑)
2k	$\mathcal{L}_2$	0.607 ± 0.013	1.144 ± 0.058	0.135 ± 0.012	0.393 ± 0.037	0.497 ± 0.101	0.757 ± 0.042	0.899 ± 0.025
	$\mathcal{L}_1$	0.598 ± 0.007	1.133 ± 0.076	0.124 ± 0.002	0.381 ± 0.036	0.407 ± 0.059	0.802 ± 0.015	0.926 ± 0.006
	$\mathcal{L}_{\text{Huber}}$	0.589 ± 0.037	1.084 ± 0.088	0.109 ± 0.001	0.316 ± 0.004	0.305 ± 0.014	0.843 ± 0.006	0.949 ± 0.002
	$\mathcal{L}_{\text{BerHu}}$	0.580 ± 0.004	1.074 ± 0.034	0.110 ± 0.004	0.320 ± 0.012	0.317 ± 0.034	0.836 ± 0.013	0.946 ± 0.006
	$\mathcal{L}_{\text{Ruber}}$	0.592 ± 0.020	1.085 ± 0.024	0.107 ± 0.005	0.313 ± 0.013	0.320 ± 0.050	0.847 ± 0.017	0.947 ± 0.010
	$\mathcal{L}_{\text{Barron}}$	0.585 ± 0.013	1.090 ± 0.022	0.115 ± 0.008	0.331 ± 0.020	0.360 ± 0.045	0.824 ± 0.026	0.938 ± 0.012
	$\mathcal{L}_{\text{trim}}$	0.664 ± 0.087	1.183 ± 0.069	0.129 ± 0.007	0.371 ± 0.023	0.575 ± 0.227	0.778 ± 0.024	0.912 ± 0.016
	$\mathcal{L}_{\text{ScaledSIError}}$	0.572 ± 0.016	**1.043** ± 0.008	**0.097** ± 0.007	**0.284** ± 0.018	**0.265** ± 0.043	**0.877** ± 0.019	**0.961** ± 0.008
	$\mathcal{L}_{\text{WeightedL2}}$	0.582 ± 0.005	1.132 ± 0.044	0.134 ± 0.009	0.387 ± 0.028	0.483 ± 0.076	0.761 ± 0.030	0.903 ± 0.019
	$\text{OSL}_{\mathcal{L}_1}$	**0.567** ± 0.024	1.091 ± 0.048	0.123 ± 0.012	0.359 ± 0.035	0.432 ± 0.093	0.794 ± 0.038	0.929 ± 0.021
	$\text{OSL}_{\mathcal{L}_2}$	0.592 ± 0.050	1.203 ± 0.115	0.137 ± 0.016	0.395 ± 0.044	0.561 ± 0.142	0.746 ± 0.055	0.889 ± 0.034
	$\text{FOSL}_{\mathcal{L}_1}$	0.590 ± 0.019	1.103 ± 0.035	0.112 ± 0.005	0.329 ± 0.018	0.336 ± 0.007	0.832 ± 0.018	0.948 ± 0.010
10k	$\mathcal{L}_2$	0.585 ± 0.010	1.088 ± 0.008	0.115 ± 0.001	0.336 ± 0.002	0.343 ± 0.032	0.824 ± 0.006	0.940 ± 0.005
	$\mathcal{L}_1$	0.586 ± 0.030	1.072 ± 0.032	0.095 ± 0.004	0.280 ± 0.011	0.251 ± 0.034	0.883 ± 0.014	0.964 ± 0.005
	$\mathcal{L}_{\text{Huber}}$	0.588 ± 0.027	1.123 ± 0.030	0.096 ± 0.001	0.284 ± 0.003	0.279 ± 0.019	0.878 ± 0.003	0.959 ± 0.002
	$\mathcal{L}_{\text{BerHu}}$	0.593 ± 0.008	1.093 ± 0.010	0.086 ± 0.001	0.258 ± 0.003	0.208 ± 0.009	0.904 ± 0.004	0.973 ± 0.000
	$\mathcal{L}_{\text{Ruber}}$	0.578 ± 0.003	1.068 ± 0.012	0.087 ± 0.004	0.261 ± 0.011	0.218 ± 0.021	0.901 ± 0.010	0.971 ± 0.004
	$\mathcal{L}_{\text{Barron}}$	0.579 ± 0.021	1.103 ± 0.013	0.106 ± 0.009	0.311 ± 0.026	0.315 ± 0.053	0.849 ± 0.029	0.951 ± 0.012
	$\mathcal{L}_{\text{trim}}$	0.602 ± 0.019	1.080 ± 0.022	0.098 ± 0.013	0.313 ± 0.070	0.244 ± 0.048	0.884 ± 0.024	0.964 ± 0.012
	$\mathcal{L}_{\text{ScaledSIError}}$	0.600 ± 0.025	1.096 ± 0.040	**0.079** ± 0.002	**0.237** ± 0.004	**0.171** ± 0.007	**0.926** ± 0.002	**0.981** ± 0.001
	$\mathcal{L}_{\text{WeightedL2}}$	0.580 ± 0.007	1.065 ± 0.021	0.107 ± 0.004	0.314 ± 0.011	0.295 ± 0.028	0.848 ± 0.012	0.950 ± 0.007
	$\text{OSL}_{\mathcal{L}_1}$	0.541 ± 0.004	1.053 ± 0.005	0.107 ± 0.002	0.311 ± 0.004	0.324 ± 0.019	0.844 ± 0.004	0.947 ± 0.004
	$\text{OSL}_{\mathcal{L}_2}$	**0.538** ± 0.006	**1.039** ± 0.023	0.115 ± 0.004	0.335 ± 0.012	0.379 ± 0.036	0.821 ± 0.017	0.932 ± 0.010
	$\text{FOSL}_{\mathcal{L}_1}$	0.597 ± 0.033	1.102 ± 0.045	0.091 ± 0.004	0.272 ± 0.013	0.232 ± 0.033	0.891 ± 0.015	0.966 ± 0.008

Tab. 3 further shows results on *DIODE* and *NYUD-v2* for the study involving artificial noise added to the training data. As can be seen, $\mathcal{L}_{\text{Ruber}}$ performs best on the noisy test data, whereas it provides inferior performance to most of the superset losses on the clear *DIODE* data. This suggests that it is more prone to reproduce the sensor errors.

Table 3: Further results on *DIODE* and the Eigen test split of *NYUD-v2* when trained on 2k instances from *NYUD-v2* with artificial noise (averaged results over three runs). The best model is indicated in bold per noise degree and metric.

Noise $\hat{\epsilon}$	Loss	*DIODE*		*NYUD-v2*							
		RMSLog (↓)	SQREL (↓)	REL (↓)	$\log_{10}$ (↓)	RMS (↓)	δ_1 (↑)	δ_2 (↑)	δ_3 (↑)	RMSLog (↓)	SQREL (↓)
0.5	$\mathcal{L}_2$	0.652 ± 0.031	2.074 ± 0.281	0.789 ± 0.055	0.226 ± 0.012	1.677 ± 0.098	0.221 ± 0.012	0.484 ± 0.026	0.700 ± 0.031	0.605 ± 0.031	1.669 ± 0.218
	$\mathcal{L}_1$	0.597 ± 0.013	1.126 ± 0.034	0.343 ± 0.026	0.130 ± 0.008	0.976 ± 0.061	0.474 ± 0.027	0.773 ± 0.030	0.912 ± 0.017	0.388 ± 0.021	0.419 ± 0.053
	$\mathcal{L}_{\text{Huber}}$	0.568 ± 0.018	1.301 ± 0.177	0.464 ± 0.090	0.150 ± 0.021	1.175 ± 0.175	0.427 ± 0.068	0.705 ± 0.065	0.862 ± 0.041	0.431 ± 0.047	0.732 ± 0.243
	$\mathcal{L}_{\text{BerHu}}$	0.556 ± 0.014	1.080 ± 0.027	0.306 ± 0.008	0.112 ± 0.003	0.860 ± 0.028	**0.545** ± 0.014	0.829 ± 0.008	0.941 ± 0.003	**0.326** ± 0.008	0.348 ± 0.022
	$\mathcal{L}_{\text{Ruber}}$	0.580 ± 0.008	1.079 ± 0.012	**0.286** ± 0.012	0.112 ± 0.004	**0.853** ± 0.024	0.535 ± 0.010	**0.833** ± 0.010	**0.946** ± 0.006	0.328 ± 0.014	**0.307** ± 0.023
	$\mathcal{L}_{\text{Barron}}$	0.591 ± 0.016	1.591 ± 0.130	0.667 ± 0.028	0.200 ± 0.005	1.473 ± 0.045	0.249 ± 0.006	0.551 ± 0.013	0.772 ± 0.023	0.538 ± 0.016	1.216 ± 0.105
	$\mathcal{L}_{\text{trim}}$	0.612 ± 0.039	1.417 ± 0.022	0.510 ± 0.002	0.174 ± 0.003	1.269 ± 0.011	0.360 ± 0.001	0.635 ± 0.002	0.818 ± 0.001	0.527 ± 0.034	0.803 ± 0.023
	$\mathcal{L}_{\text{ScaledSIError}}$	1.143 ± 0.491	1.796 ± 0.545	0.521 ± 0.175	0.369 ± 0.187	1.753 ± 0.472	0.154 ± 0.142	0.306 ± 0.269	0.437 ± 0.351	0.914 ± 0.412	0.993 ± 0.486
	$\mathcal{L}_{\text{WeightedL2}}$	0.613 ± 0.007	1.671 ± 0.028	0.618 ± 0.009	0.189 ± 0.002	1.362 ± 0.015	0.315 ± 0.006	0.590 ± 0.005	0.778 ± 0.005	0.522 ± 0.004	1.076 ± 0.025
	$\text{OSL}_{\mathcal{L}_1}$	**0.555** ± 0.012	1.117 ± 0.037	0.369 ± 0.057	0.129 ± 0.013	0.932 ± 0.078	0.491 ± 0.043	0.774 ± 0.045	0.907 ± 0.029	0.371 ± 0.037	0.462 ± 0.104
	$\text{OSL}_{\mathcal{L}_2}$	0.621 ± 0.026	1.884 ± 0.242	0.770 ± 0.079	0.223 ± 0.018	1.552 ± 0.149	0.208 ± 0.026	0.487 ± 0.048	0.717 ± 0.044	0.595 ± 0.047	1.581 ± 0.307
	$\text{FOSL}_{\mathcal{L}_1}$	0.583 ± 0.021	**1.078** ± 0.050	0.300 ± 0.012	**0.107** ± 0.006	0.857 ± 0.029	0.529 ± 0.021	0.825 ± 0.017	0.939 ± 0.009	0.343 ± 0.031	0.34 ± 0.02
1.0	$\mathcal{L}_2$	0.898 ± 0.093	5.929 ± 0.764	1.387 ± 0.387	0.333 ± 0.073	2.899 ± 0.714	0.130 ± 0.097	0.268 ± 0.161	0.438 ± 0.170	0.861 ± 0.151	5.079 ± 2.088
	$\mathcal{L}_1$	0.561 ± 0.006	1.117 ± 0.022	0.366 ± 0.014	0.133 ± 0.003	1.007 ± 0.031	0.460 ± 0.007	0.758 ± 0.013	0.904 ± 0.009	0.381 ± 0.008	0.464 ± 0.031
	$\mathcal{L}_{\text{Huber}}$	0.701 ± 0.010	2.834 ± 0.151	1.073 ± 0.102	0.286 ± 0.020	2.384 ± 0.229	0.112 ± 0.026	0.307 ± 0.055	0.556 ± 0.052	0.735 ± 0.043	3.017 ± 0.472
	$\mathcal{L}_{\text{BerHu}}$	0.566 ± 0.008	1.107 ± 0.036	0.375 ± 0.036	0.132 ± 0.008	0.993 ± 0.059	0.482 ± 0.022	0.772 ± 0.029	0.905 ± 0.018	0.389 ± 0.028	0.498 ± 0.079
	$\mathcal{L}_{\text{Ruber}}$	0.571 ± 0.001	1.096 ± 0.035	**0.295** ± 0.030	**0.113** ± 0.008	**0.880** ± 0.071	**0.532** ± 0.030	**0.833** ± 0.026	**0.945** ± 0.014	**0.325** ± 0.023	**0.354** ± 0.085
	$\mathcal{L}_{\text{Barron}}$	0.721 ± 0.041	3.004 ± 0.581	1.001 ± 0.134	0.270 ± 0.027	2.059 ± 0.254	0.152 ± 0.052	0.367 ± 0.075	0.592 ± 0.066	0.707 ± 0.061	2.541 ± 0.592
	$\mathcal{L}_{\text{trim}}$	0.575 ± 0.003	1.361 ± 0.036	0.488 ± 0.019	0.167 ± 0.003	1.231 ± 0.018	0.369 ± 0.009	0.648 ± 0.011	0.828 ± 0.008	0.470 ± 0.009	0.724 ± 0.045
	$\mathcal{L}_{\text{ScaledSIError}}$	1.124 ± 0.407	1.912 ± 0.352	0.593 ± 0.115	0.410 ± 0.188	1.920 ± 0.448	0.134 ± 0.144	0.265 ± 0.260	0.381 ± 0.320	1.025 ± 0.424	1.209 ± 0.411
	$\mathcal{L}_{\text{WeightedL2}}$	0.670 ± 0.033	1.861 ± 0.266	0.667 ± 0.100	0.201 ± 0.021	1.464 ± 0.144	0.291 ± 0.050	0.553 ± 0.058	0.750 ± 0.044	0.555 ± 0.050	1.249 ± 0.301
	$\text{OSL}_{\mathcal{L}_1}$	**0.559** ± 0.016	1.093 ± 0.018	0.351 ± 0.059	0.122 ± 0.014	0.933 ± 0.075	0.511 ± 0.044	0.796 ± 0.045	0.918 ± 0.028	0.354 ± 0.037	0.441 ± 0.110
	$\text{OSL}_{\mathcal{L}_2}$	0.783 ± 0.033	3.304 ± 0.811	1.211 ± 0.276	0.310 ± 0.051	2.546 ± 0.566	0.125 ± 0.073	0.288 ± 0.129	0.493 ± 0.132	0.818 ± 0.083	3.812 ± 1.484
	$\text{FOSL}_{\mathcal{L}_1}$	**0.559** ± 0.007	**1.089** ± 0.030	0.322 ± 0.014	0.118 ± 0.003	0.910 ± 0.029	0.516 ± 0.015	0.811 ± 0.010	0.931 ± 0.007	0.342 ± 0.009	0.384 ± 0.036

3.3. Heterogeneous Depth Sensors: SunRGBD

In Tab. 4, we provide more results on *DIODE* and the official test split of *SunRGBD* when trained on the corresponding training part. While the results on *DIODE* are matching the observations made with regard to the other metrics, $\mathcal{L}_{\text{WeightedL2}}$ performs remarkably well on the noisy *SunRGBD* test data when trained on the full data. It is worth to note that the performance on the latter test data gets worse the more data is observed. This could potentially be due to a domain-shift between *iBims-1* and *SunRGBD*.

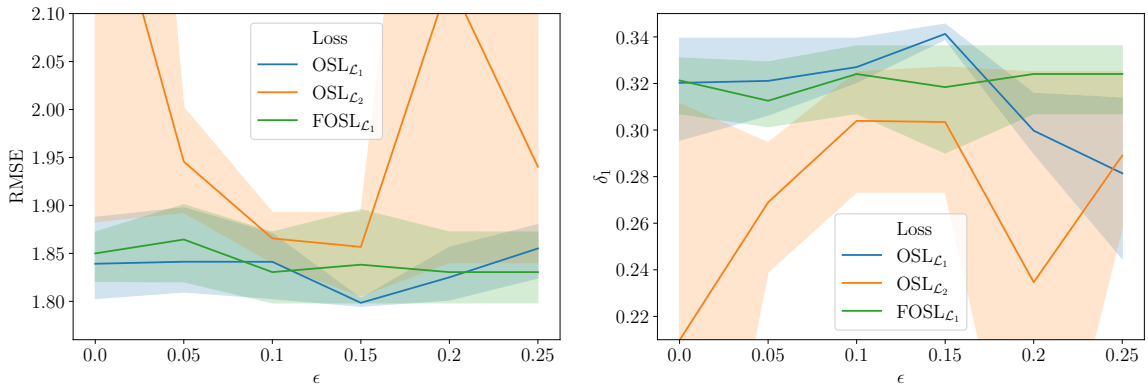
3.4. Hyperparameter Sensitivity

All superset losses employ a hyperparameter ϵ that determines the degree of imprecisiation involved in the training. Here, we assess the sensitivity of this parameter when training on *NYUD-v2* with either 2k or 10k subsamples in the same experimental setting as considered before. To this end, we fix ϵ to values in $[0, 0.25]$ and solely optimize the initial learning rate η on *iBims-1* as we did in the main experiments, and report the RMSE and δ_1 accuracy on *DIODE*. For $\text{FOSL}_{\mathcal{L}_1}$, we set the fuzzy set support parameter δ to 1. For statistical significance, we conducted each experiment five times.

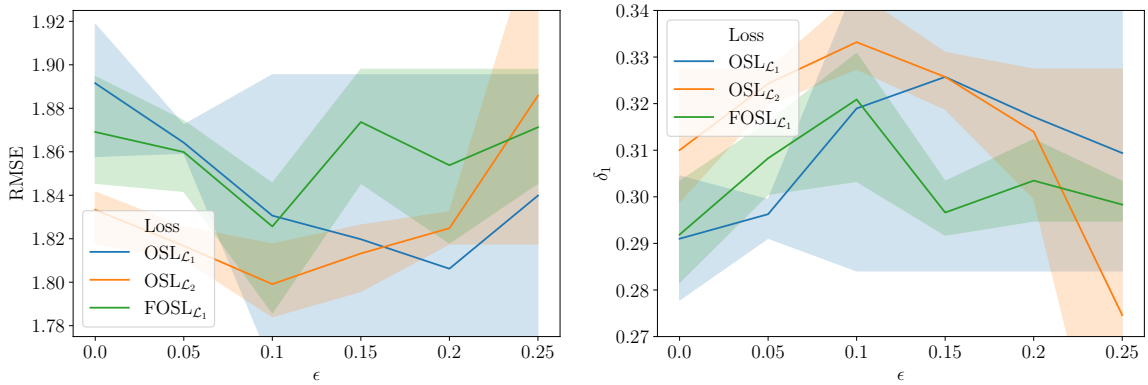
As can be seen in Figure 2(*a*) for 2k instances, $\text{OSL}_{\mathcal{L}_1}$ benefits from higher degrees of imprecisiation towards $\epsilon = 0.15$ compared to simple $\mathcal{L}_1$ ($\epsilon = 0$). On the contrary, least squares optimization using $\text{OSL}_{\mathcal{L}_2}$ leads to worse performance with higher variance. In case of 10k training instances (Figure 2(*b*)), $\text{OSL}_{\mathcal{L}_2}$ provides the best performance for $\epsilon = 0.1$, whereas the variance of $\text{OSL}_{\mathcal{L}_1}$ increases with higher degrees of imprecisiation. In any case, $\text{FOSL}_{\mathcal{L}_1}$ shows relatively low sensitivity to the degree of imprecisiation, while providing competitive performance at the same time.

Table 4: Additional results on *DIODE* and the official *SunRGBD* test split (average over three runs). The best model is indicated in bold per number of instances and metric.

# Insts.	Loss	*DIODE*		*SunRGBD*				
		RMSLog (↓)	SQREL (↓)	$\log_{10}$ (↓)	δ_2 (↑)	δ_3 (↑)	RMSLog (↓)	SQREL (↓)
2k	$\mathcal{L}_2$	0.797 ± 0.195	1.221 ± 0.117	0.148 ± 0.007	**0.747** ± 0.002	**0.911** ± 0.005	0.472 ± 0.045	0.708 ± 0.055
	$\mathcal{L}_1$	0.577 ± 0.010	1.057 ± 0.056	0.146 ± 0.007	0.725 ± 0.026	0.896 ± 0.016	0.446 ± 0.020	0.718 ± 0.089
	$\mathcal{L}_{\text{Huber}}$	0.561 ± 0.010	1.064 ± 0.039	0.145 ± 0.007	0.730 ± 0.026	0.901 ± 0.016	0.442 ± 0.018	0.782 ± 0.089
	$\mathcal{L}_{\text{BerHu}}$	0.581 ± 0.026	1.030 ± 0.053	0.154 ± 0.002	0.712 ± 0.012	0.887 ± 0.009	0.499 ± 0.047	0.782 ± 0.054
	$\mathcal{L}_{\text{Ruber}}$	0.562 ± 0.013	**1.020** ± 0.036	0.150 ± 0.003	0.710 ± 0.010	0.888 ± 0.007	0.454 ± 0.007	0.798 ± 0.039
	$\mathcal{L}_{\text{Barron}}$	0.568 ± 0.013	1.101 ± 0.022	0.151 ± 0.006	0.709 ± 0.021	0.886 ± 0.015	0.457 ± 0.016	0.896 ± 0.110
	$\mathcal{L}_{\text{trim}}$	0.600 ± 0.026	1.215 ± 0.060	0.150 ± 0.011	0.746 ± 0.009	0.910 ± 0.007	0.477 ± 0.063	**0.680** ± 0.101
	$\mathcal{L}_{\text{ScaledSIError}}$	0.579 ± 0.009	1.053 ± 0.015	0.152 ± 0.003	0.706 ± 0.010	0.883 ± 0.008	0.458 ± 0.008	0.784 ± 0.036
	$\mathcal{L}_{\text{WeightedL2}}$	0.582 ± 0.018	1.107 ± 0.054	**0.143** ± 0.005	0.740 ± 0.018	0.905 ± 0.010	**0.441** ± 0.016	0.703 ± 0.078
	$\text{OSL}_{\mathcal{L}_1}$	**0.538** ± 0.018	1.054 ± 0.019	0.145 ± 0.004	0.723 ± 0.014	0.894 ± 0.010	0.453 ± 0.011	0.795 ± 0.040
	$\text{OSL}_{\mathcal{L}_2}$	0.541 ± 0.007	1.073 ± 0.063	0.144 ± 0.007	0.729 ± 0.026	0.902 ± 0.017	0.452 ± 0.019	0.796 ± 0.102
	$\text{FOSL}_{\mathcal{L}_1}$	0.565 ± 0.007	**1.020** ± 0.018	0.151 ± 0.003	0.708 ± 0.009	0.884 ± 0.005	0.459 ± 0.008	0.788 ± 0.041
Full	$\mathcal{L}_2$	0.535 ± 0.006	0.972 ± 0.035	0.152 ± 0.001	0.706 ± 0.003	0.885 ± 0.003	0.458 ± 0.003	0.862 ± 0.006
	$\mathcal{L}_1$	0.561 ± 0.010	0.998 ± 0.036	0.154 ± 0.002	0.698 ± 0.007	0.877 ± 0.004	0.464 ± 0.005	0.845 ± 0.044
	$\mathcal{L}_{\text{Huber}}$	0.533 ± 0.022	0.960 ± 0.023	0.156 ± 0.003	0.691 ± 0.011	0.873 ± 0.009	0.469 ± 0.008	0.906 ± 0.021
	$\mathcal{L}_{\text{BerHu}}$	0.551 ± 0.027	0.944 ± 0.045	0.158 ± 0.001	0.687 ± 0.004	0.869 ± 0.003	0.484 ± 0.015	0.927 ± 0.053
	$\mathcal{L}_{\text{Ruber}}$	0.521 ± 0.005	0.900 ± 0.022	0.158 ± 0.001	0.688 ± 0.003	0.868 ± 0.003	0.477 ± 0.003	0.940 ± 0.047
	$\mathcal{L}_{\text{Barron}}$	0.535 ± 0.006	0.979 ± 0.016	0.155 ± 0.001	0.696 ± 0.005	0.877 ± 0.003	0.466 ± 0.004	0.906 ± 0.038
	$\mathcal{L}_{\text{trim}}$	0.563 ± 0.021	1.032 ± 0.048	0.159 ± 0.008	0.693 ± 0.014	0.872 ± 0.011	0.505 ± 0.053	0.963 ± 0.125
	$\mathcal{L}_{\text{ScaledSIError}}$	0.519 ± 0.016	0.913 ± 0.027	0.158 ± 0.002	0.686 ± 0.006	0.867 ± 0.005	0.473 ± 0.004	0.868 ± 0.039
	$\mathcal{L}_{\text{WeightedL2}}$	0.540 ± 0.017	0.983 ± 0.047	**0.150** ± 0.001	**0.711** ± 0.005	0.888 ± 0.003	**0.455** ± 0.004	**0.839** ± 0.017
	$\text{OSL}_{\mathcal{L}_1}$	**0.486** ± 0.015	**0.826** ± 0.069	0.155 ± 0.002	0.702 ± 0.008	0.875 ± 0.005	0.460 ± 0.006	0.929 ± 0.033
	$\text{OSL}_{\mathcal{L}_2}$	0.507 ± 0.013	0.976 ± 0.114	0.152 ± 0.001	0.706 ± 0.005	**0.889** ± 0.004	0.456 ± 0.002	0.912 ± 0.041
	$\text{FOSL}_{\mathcal{L}_1}$	0.550 ± 0.023	0.934 ± 0.023	0.157 ± 0.001	0.694 ± 0.007	0.878 ± 0.003	0.466 ± 0.004	0.943 ± 0.056



(a) Models trained with 2k instances.



(b) Models trained with 10k instances.

Figure 2: RMSE and δ_1 with the respective standard deviations on the *DIODE* test data for models trained on either 2k or 10k instances from *NYUD-v2*.

3.5. Discussion

In our experiments, we consider various scenarios that may apply to real-life use cases. In most scenarios, either facing few training instances in a relatively uniform domain (*NYUD-v2* with 2k instances) or training data captured by heterogeneous depth sensors (*SunRGBD*), $\mathrm{OSL}_{\mathcal{L}_1}$ represents an easy to optimize yet well performing method for robust depth regression. Nevertheless, $\mathrm{OSL}_{\mathcal{L}_2}$ becomes more effective when observing more instances in the homogeneous sensor setting, whereas $\mathrm{FOSL}_{\mathcal{L}_1}$ has shown promising performance in the high noise scenario. However, the increased expressiveness of complex fuzzy set-based superset modeling comes with a larger hyperparameter search space, making its optimization more resource demanding. This method could potentially benefit from more than 20 trials in the random search as conducted within our experiments, as well as from more sophisticated yet domain tailored fuzzy set modeling.

4. Exemplary Predictions

In Fig. 3, we provide sample predictions of selected baselines and our superset learning-based methods on *DIODE*. Here, we consider models trained on a subset of 2k instances from *NYUD-v2* with different degrees of noise injection. As can be seen, the robust superset learning-based losses keep providing reliable predictions even for high degrees of noise, for which $\mathcal{L}_2$-related losses fail.

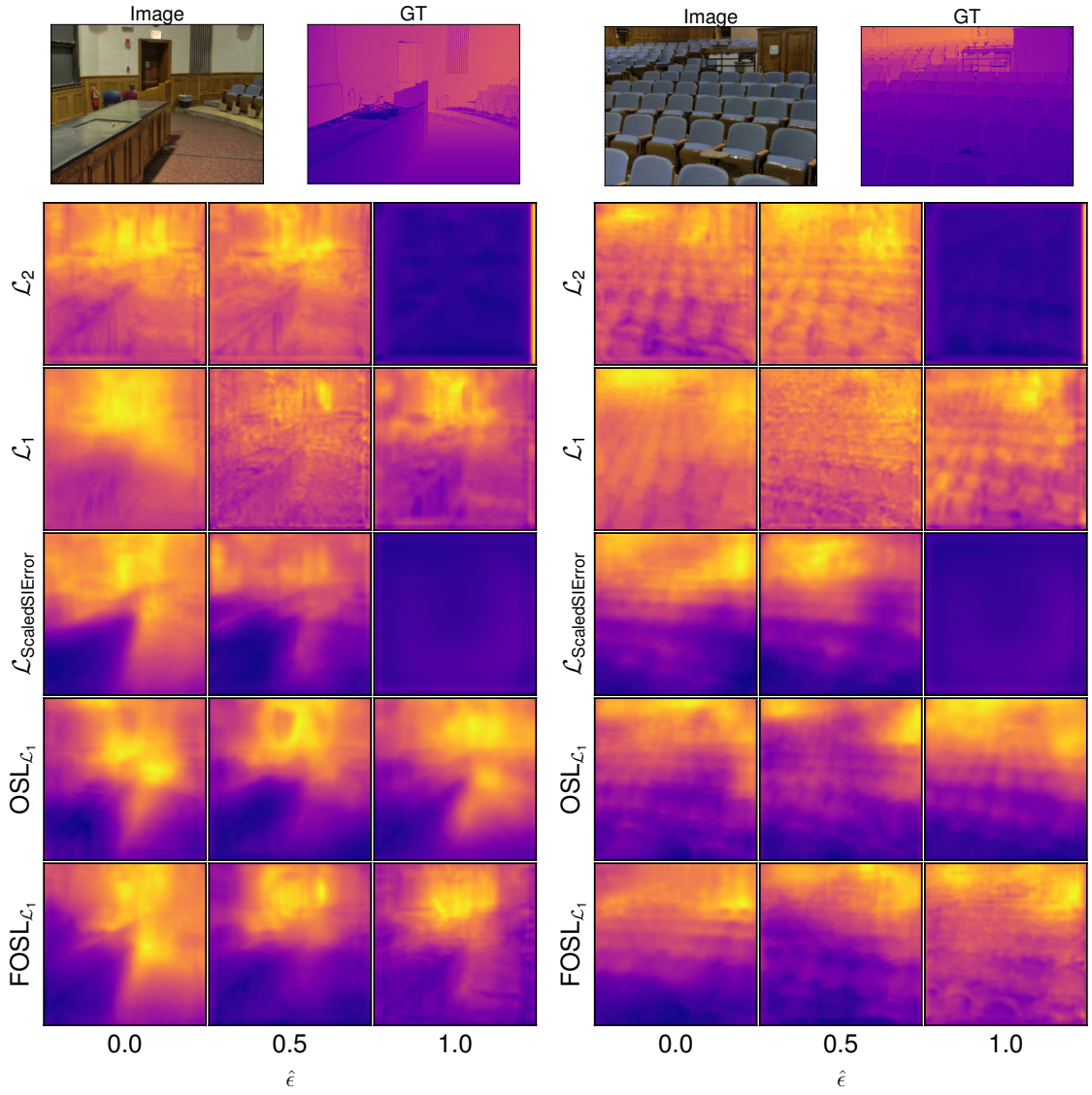


Figure 3: Exemplary predictions for two *DIODE* images of models trained on a subset of discussed loss functions. Here, we compare models trained with different noise levels $\hat{\epsilon}$ injected into the training data as discussed in the paper.

References

Jonathan T. Barron. A general and adaptive robust loss function. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 4331–4339. Computer Vision Foundation / IEEE, 2019.

Go Irie, Takahito Kawanishi, and Kunio Kashino. Robust learning for deep monocular depth estimation. In *Proc. of the IEEE International Conference on Image Processing, ICIP*, pages 964–968. IEEE, 2019.

Tobias Koch, Lukas Liebel, Friedrich Fraundorfer, and Marco Körner. Evaluation of CNN-based single-image depth estimation methods. In *Proc. of the European Conference on Computer Vision, ECCV, Workshops Part III*, volume 11131 of *LNCS*, pages 331–348. Springer, 2018.

Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *Proc. of the 4th International Conference on 3D Vision, 3DV*, pages 239–248. IEEE Computer Society, 2016.

Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *CoRR*, abs/1907.10326, 2019.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *Proc. of the 5th International Conference on Learning Representations, ICLR*. OpenReview.net, 2017.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *Proc. of the 12th European Conference on Computer Vision, ECCV, Part V*, volume 7576 of *LNCS*, pages 746–760. Springer, 2012.

Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 567–576. IEEE Computer Society, 2015.

Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.