Supplementary Material: Compensation Learning in Semantic Segmentation

Timo Kaiser, Christoph Reinders, Bodo Rosenhahn
Institute for Information Processing (tnt)
L3S - Leibniz Universität Hannover, Germany
{kaiser, reinders, rosenhahn}@tnt.uni-hannover.de

We present supplemental material for our paper *Compensation Learning in Semantic Segmentation* to disclose more details about the method and experiments. In the first section, we present detail of the implementation of our and the referenced methods. The second section shows mathematical details of the metrics used in our experiments. The third section elaborates the process of noise induction used in the main paper. Then, more fine-grained results of the bias induced inference experiments are presented in section four. Finally, the fifth and sixth section provide the full compensation and confusion matrices, mentioned in the main paper.

A. Experimental Setup

To simplify the reproduction process, we integrate our method as well as the referenced methods into the well-known framework *MMSegmentation* [3], if no suitable code was available. *MMSegmentation* provides code for multiple state-of-the-art semantic segmentation frameworks. Also our baseline methods *DeepLabv3+* and *SegFormer* are implemented in *MMSegmentation* with different hyperparameter settings for different datasets. Reference links to the exact configurations for our baseline experiments can be found in Tab. 1. The next section elaborates on how the referenced methods and datasets used in the main paper are integrated and configured.

The code for our method and some mentioned competitors are maintained in a local github fork[1]. We want to note that we endeavour to merge the code into the widely-used public repository[2] in the future to enlarge the visibility.

A.1. Model Architectures

To compare our method, reference methods are evaluated in the main paper. Except *Hyperbolic Image Segmentation*, the methods *s-model*, *c-model*, and a *Bayesian Neural Network* methods are integrated on top of the baseline frameworks *DeepLabv3+* and *SegFormer*. The following sections elaborates the implementation of all reference methods.

[1] https://github.com/tnt-LUH/compensation_learning
[2] https://github.com/open-mmlab/mmsegmentation

s-model and c-model

The methods *s-model* and *c-model* are referenced in the robust learning experiments in Sec. 4.4 in the main paper. Both methods are introduced in [5] and implement a trainable transition matrix into a neural network. The transition matrix T encodes the global noise distribution and can be used to estimate the clean probability $P(Y = i|x)$ for class i from a noisy probability prediction $\overline{P}(Y = i|x)$ of a pixel x. For the pixel x and a set of classes C, this estimation can be formulated as

$$\forall x \in I: \quad \vec{\overline{p}}_x = T\vec{p}_x, \quad T \in [0,1]^{C\times C} \tag{1}$$

with the additional condition

$$\forall j \in C: \quad \sum_i T_{ij} = 1 \tag{2}$$

to conserve a probability distribution. During training, the matrix T is optimized alongside the model parameters to fit on the noisy labels, and during inference of unseen data, Eq. (1) is removed from the framework to obtain the clean probabilities $\vec{p}_x$.

To implement the simple *s-model* from [5], we transfer Eq. (1) into the two-dimensional case by modeling T as a convolutional layer with kernel size 1 (*a.k.a.* pointwise CNN), and input/output-channels of C. This convolution is applied to the final two-dimensional probability map of the underlying baseline framework. To satisfy Eq. (2), the weights of the convolutional layer ($= T$) are normalized using the softmax function (see Eq. 1 in the main paper) on each column before convolving the two-dimensional probability map containing $\vec{p}_x$ for all pixels x.

To implement the complex *c-model* from [5], we extend *s-model*. In *s-model*, a shared transition matrix T is used for all pixels in all images globally. Compared to the global approach, *c-model* estimates the transition matrix individually for every pixel based on the high-level image features. To implement this into the segmentation framework,

Table 1. Model configurations references to reproduce the training in *MMSegmentation*. Note that learning rate decay as in [2] is applied.

	Dataset	Training Crop-Size	Steps	Learning Rate	Optimizer	Reference Configuration File
DeepLabv3+	Cityscapes	512×1024	80 000	$1 \cdot 10^{-2}$	SGD	deeplabv3plus_r50-d8_512x1024_80k_cityscapes.py
	KITTI-STEP	368×368	80 000	$1 \cdot 10^{-2}$	SGD	deeplabv3plus_r50-d8_368x368_80k_kittistep.py
	COCO-stuff10k	512×512	80 000	$1 \cdot 10^{-2}$	SGD	deeplabv3plus_r50-d8_512x512_80k_coco-stuff10k.py
	ADE20k	512×512	80 000	$1 \cdot 10^{-2}$	SGD	deeplabv3plus_r50-d8_512x512_80k_ade20k.py
SegFormer	Cityscapes	1024×1024	160 000	$6 \cdot 10^{-5}$	Adam	segformer_mit-b5_8x1_1024x1024_160k_cityscapes.py
	KITTI-STEP	368×368	160 000	$6 \cdot 10^{-5}$	Adam	segformer_mit-b5_368x368_160k_kittistep.py
	COCO-stuff10k	512×512	160 000	$6 \cdot 10^{-5}$	Adam	segformer_mit-b5_512x512_160k_coco-stuff10k.py
	ADE20k	512×512	160 000	$6 \cdot 10^{-5}$	Adam	segformer_mit-b5_512x512_160k_ade20k.py

we employed an additional branch parallel to the segmentation head similar to the implementation of our local uncertainty branch. The branch contains two pointwise two-dimensional convolutional layers. The first layer has input channels depending on the dimensions of the high-level image features and 32 output channels. This layer is used to reduce the complexity to a tractable size. The second layer has an input size of 32 and an output size of $|C|^2$. The $|C|^2$-dimensional output for pixel x is reshaped to a $C \times C$ matrix, normalized with softmax, and used as individual weight for the final convolution described above for *s-model*. We want to note that the computational effort for large-scale datasets with a large amount of classes like ADE20k or COCO-stuff10k is not computable with current state-of-the-art graphic cards and segmentation frameworks. Therefore the experiments for *c-model* and the large-scale datasets cannot be evaluated.

The code integration of *s-model* and *c-model* into *MMSegmentation* can be found here[3].

Bayesian Neural Network

Mukhoti *et al.* [9] present a *Bayesian Neural Network* (BNN) that implements *Monte-Carlo Dropout* into *DeepLabv3+* with a comparable backbone as used in our experiment. *Monte-Carlo Dropout* consists of basically two modifications of the baseline framework: Dropout layers with high dropout probability d are applied to high-level feature maps during training and during test time, the inference is repeated multiple times with active dropout layers to generate sample predictions from a Bernoulli-distributed weight distribution. The mean of the generated samples is the most likely correct prediction and the variance can be interpreted as the prediction uncertainty.

The implementation of a BNN into *DeepLabv3+* is described in [9] for the backbone *Xception*. To be comparable, we implemented their approach into the *ResNet50* backbone [6] used in the main paper using the same design choices. Following the design choice of only modifying the *Middle Flow* of *Xception*, we add dropout layers to the third stage of the *ResNet50*. We add the dropout layers after each residual block as described in the Appendix of [9]. Furthermore, Mukhoti *et al.* remove the *Atrous Spatial Pyramid Pooling* (ASPP) modules from *DeepLabv3+* to reduce the complexity. Because ASPP modules increase the accuracy, we do not remove them.

We optimized the dropout rate and noticed, that $d = 0.5$ leads to significant drop in the accuracy, while $d = 0.25$ conserves the accuracy and also predicts satisfying uncertainty. Thus, we set d to 0.25 for all dropout layers. The mean and variance during test time is obtained with 20 samples.

The code integration of the Bayesian ResNet into *MMSegmentation* can be found here[4].

Hyperbolic Image Segmentation

The hyperbolic image segmentation framework (HIS) of Atigh *et al.* [1] is used in Sec. 4.3 in the main paper to compare uncertainty estimation of our and other methods with respect to the label prediction correction task. The approach HIS[5] provides code for training and inference for the datasets ADE20k and COCO-stuff10k. We trained the models for the datasets with original hyperparameters and created normalized uncertainty maps for every image of both datasets. We noticed that there are high-valued fragments at the border of the uncertainty prediction, which distort the subsequent normalization process. Thus, we set the uncertainty of the 10 pixels closest to the border to zero, to allow reasonable evaluation w.r.t. the prediction error correction task. We do not evaluate HIS for Cityscapes and KITTI-STEP, because optimized hyperparameters and evaluations are not given in [1] and we want to avoid unfair comparisons.

A.2. Datasets

MMSegmentation provides a standard setting including augmentations and data pre-processing methods. In the following, the configuration for the non-preconfigured dataset KITTI-STEP is explained.

[3] https://github.com/tnt-LUH/compensation_learning#NAL

[4] https://github.com/tnt-LUH/compensation_learning#BayesianNN

[5] https://github.com/MinaGhadimiAtigh/HyperbolicImageSegmentat

KITTI-STEP

The KITTI-STEP [11] dataset shares the same classes and domain as the dataset Cityscapes [4]. Augmentation and pre-processing for Cityscapes is defined in cityscapes.py. We use similar configurations than in Cityscapes, but we changed crop size in the pre-processing to 368×368. Images in KITTI-STEP are significantly smaller than in Cityscapes. The configuration file in *MMSegmentation* can be found in kittistep.py.

B. Metrics

We evaluate with the widely-used mIoU metric [10], which is also referenced as *Semantic Quality* [8, 11]. With the set of pixels X_c that are assigned to class c and ground truth labels $\hat{Y}_c$ annotated as c, the mIoU metric is defined as

$$\text{mIoU} = \frac{1}{|C|} \sum_{c \in C} \frac{|X_c \cap \hat{Y}_c|}{|X_c \cup \hat{Y}_c|} \,. \tag{3}$$

The mIoU metric evaluates the assignment of class labels and balances underrepresented classes. We also use the class accuracy (Acc_c) and aggregated accuracy (Acc_a)

$$\text{Acc}_c = \frac{|X_c \cap \hat{Y}_c|}{|\hat{Y}_c|} \quad \text{and} \quad \text{Acc}_a = \frac{\sum_{c \in C} |X_c \cap \hat{Y}_c|}{\sum_{c \in C} |\hat{Y}_c|} \tag{4}$$

to verify segmentation results on pixel level.

C. Noise Induction

We superficially describe the process of inducing synthetic label noise during training in the main paper, Sec. 4.4. This section elaborates this process in detail. The work of Liu *et al.* [7] corrupts ground truth annotations for medical images by dilating or eroding the masks of specific instances. We adapt this process for the multi-class datasets used in this paper.

With the help of the confusion matrices of Cityscapes and KITTI-STEP, we select some ambiguous class pairs that are not underrepresented and are often located in neighboured regions in images. Overall, we selected the class pairs *road-sidewalk*, *building-wall*, and *vegetation-terrain*.

Before we train a method with noise induction, we randomly sample a superior and inferior class for each class pair and each image. In an exemplary configuration, *road* could be superior in image A and *sidewalk* could be superior in image B. During the training process, the sampled class selection is fixed.

Then, we introduce a hyperparameter n, which reflects the noise level in the next step. We dilate regions of the superior classes, by adding pixels of the inferior class that are at most n pixels far away to a pixel of the superior class. This pattern mimes the uncertainty during human annotation.

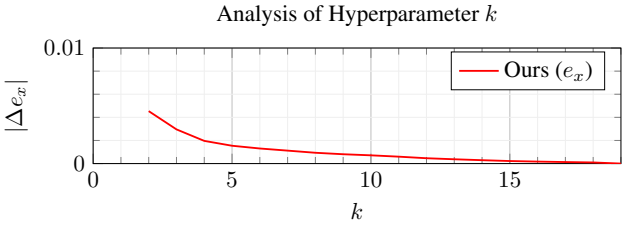


Figure 1. Mean difference of e_x with dynamic k against $k = |C|$.

Some examples with the RGB image, the original ground truth, and the ground truth after noise induction are shown in Fig. 2. The resulting label noise is close to human-like noise patterns.

D. Top-k uncertain classes

We introduced the hyperparameter k in Sec. 3.3 and set it to 5 to reduce the computational effort. Because additional experiments showed that k has neglectable impact on Acc_a, we measure the absolute difference of e_x with varying k against a fixed $k = |C|$. Fig. 1 presents the mean absolute difference $|\Delta e_x|$ over all pixels on Cityscapes. It shows that small k has most impact on the variance of e_x and additional computation with large k leads to marginal benefits.

E. Noise Detection

In the main paper, we analyzed the ability of noise detection in Sec. 4.3. Therefore, the accuracy with respect to the pixel area was reported in Fig. 4. The full plots are shown in larger resolution in Fig. 3 to 6.

F. Bias Induced Inference

This section provides more fine-grained results for the bias induced inference experiments presented in Sec. 4.5. In the main paper, we present how the accuracy for specific classes can be boosted without loosing significant accuracy over all classes. The Fig. 8a shows the accuracy of all classes in the experiment for the dataset KITTI-STEP. Most classes are not affected by the induction. The only exceptions are the classes *motorcycle* and *bicycle*, which are closely related to the induced classes. Moreover, we report the same experiment without induction for class *rider*. Since *person* and *rider* are ambiguous, the accuracy of class *rider* diminishes without induction.

We repeat the experiment for the similar dataset Cityscapes and observe the same behaviour. The results are presented in Fig. 8b.

G. Compensation Matrix

To complete the qualitative results presented in the main paper (see Sec. 4.2, main paper) and to give more

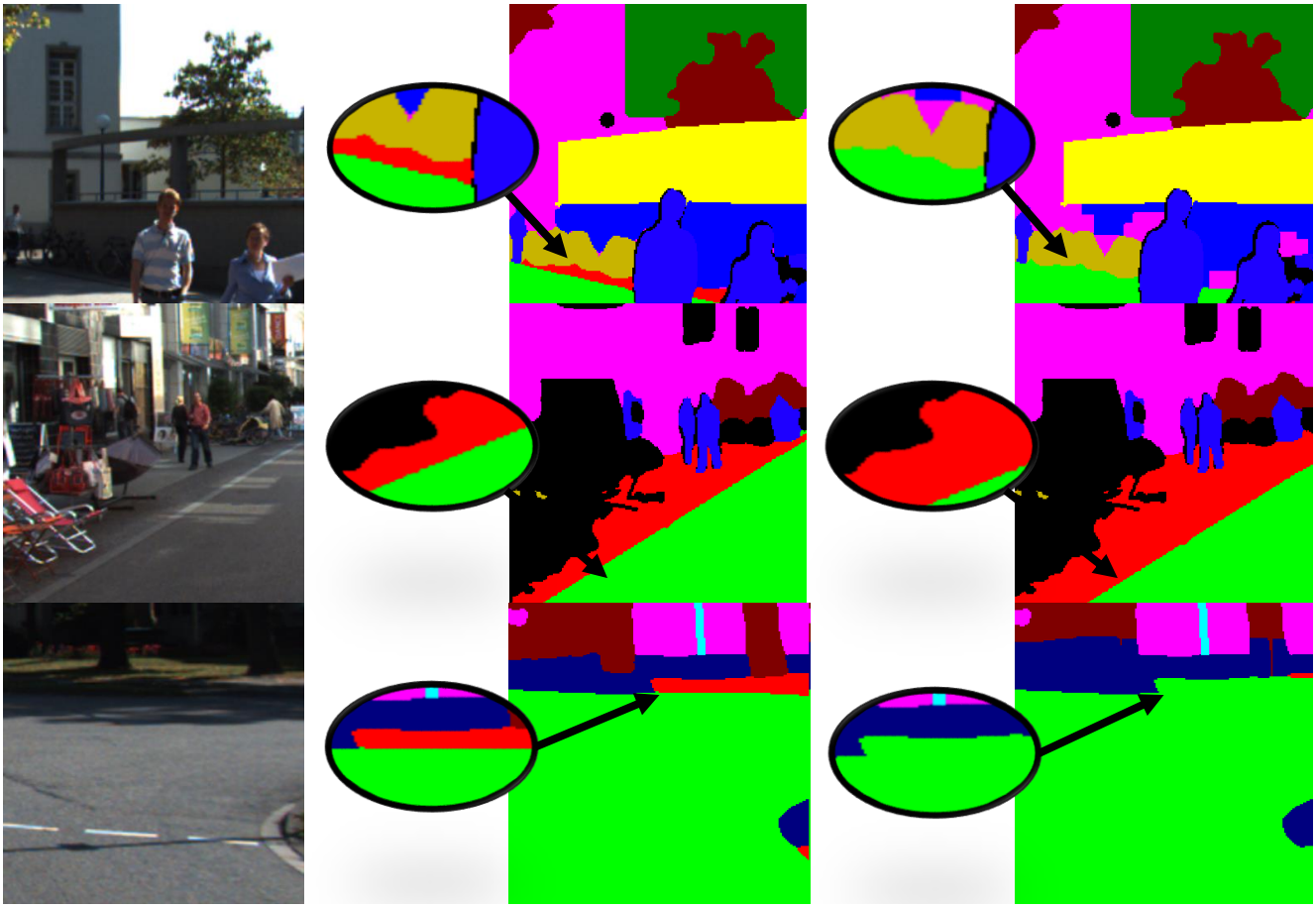


Figure 2. Synthetic ground truth degradation. Left image shows a training sample after random cropping. The center image contains ground truth annotations. The degraded ground truth annotations after applying our synthetic noise reduction with $n = 20$ are shown in the right image.

dataset specific insights with the learned compensation values in the compensation matrix B, this section presents the complete learned compensation matrices for the segmentation frameworks *DeepLabv3+* and *SegFormer* trained with the datasets Cityscapes, KITTI-STEP, ADE20K, and COCO10k-stuff. The compensation weights for the small-scale datasets are presented in Tab. 2 and 3 (*DeepLabv3+*) as well as in Tab. 4 and 5 (*SegFormer*). The compensation weights for the large-scale datasets are reported in Fig. 9. We recommend to inspect the figure details in the *pdf* version instead of a printed version.

Additional interesting insights are that the matrices for *DeepLabv3+* approximately form symmetrical, which is not happening for *SegFormer*. Moreover, the compensation weights for *SegFormer* have higher values, compared to *DeepLabv3+*. This is an indicator for the higher impact of compensation learning on transformer-based architectures.

H. Confusion Matrix

The compensation matrices of Sec. G can be compared to the confusion matrices. The confusion matrices for Cityscapes and KITTI-STEP generated on the validation and training dataset are presented in Tab. 6 to 9. It is notable that the confusion matrices on the training data does not behave like the confusion matrices on the unseen validation data. For example, in the Cityscapes and KITTI-STEP validation dataset, the class *building* interferes much with the classes *wall* and *fence*. These correlations are not equally represented in the training dataset. In practice, this behaviour leads to the need of an additional labeled dataset to obtain a suitable confusion matrix for unseen data during inference. Compared to this, our compensation weights are obtained during training without the need of additional data.

Table 2. Full compensation weight matrix B for *DeepLabv3* trained on KITTI-STEP.

B_{ij}	*stuff*											*thing*							
	road	*sidewalk*	*building*	*wall*	*fence*	*pole*	*tr. light*	*tr. sign*	*vegetation*	*terrain*	*sky*	*person*	*rider*	*car*	*truck*	*bus*	*train*	*motorcycle*	*bicycle*
road	0	-1.7	0	0	-0.1	-0.2	0	-0.1	-0.3	-1.1	0	-0.2	0	-0.6	-0.1	-0.1	0	0	-0.1
sidewalk	-1.6	0	-0.4	-0.1	-0.1	-0.2	0	0	-0.4	-0.8	0	-0.2	0	-0.3	0	0	0	0	-0.2
building	0	-0.4	0	-0.1	-0.3	-0.8	-0.1	-0.2	-1.7	-0.1	-0.5	-0.6	0	-0.4	-0.1	0	-0.1	0	-0.2
wall	0	0	-0.1	0	-0.2	0	0	0	-0.1	0	0	0	0	0	0	0	0	0	0
fence	-0.1	-0.1	-0.2	-0.2	0	-0.1	0	0	-0.7	-0.3	0	0	0	0	0	0	-0.1	0	0
pole	-0.1	-0.2	-0.8	0	-0.1	0	-0.1	-0.2	-1.1	-0.2	-0.3	0	0	-0.1	0	0	0	0	-0.1
traffic light	0	0	0	0	0	-0.1	0	0	-0.1	0	0	0	0	0	0	0	0	0	0
traffic sign	0	0	-0.2	0	0	-0.2	0	0	-0.4	0	0	0	0	0	0	0	0	0	0
vegetation	-0.3	-0.5	-1.8	-0.2	-0.8	-1.2	-0.2	-0.5	0	-1.3	-1.2	-0.2	-0.1	-0.5	-0.1	-0.1	0	0	-0.1
terrain	-1.1	-0.7	0	0	-0.3	-0.2	0	0	-1.2	0	0	0	0	-0.1	0	0	0	0	0
sky	0	0	-0.4	0	0	-0.3	0	-0.1	-1.1	0	0	0	0	0	0	-0.1	0	0	0
person	-0.1	-0.2	-0.5	0	0	0	0	0	-0.1	0	0	0	-0.2	-0.1	0	0	0	0	-0.2
rider	0	0	0	0	0	0	0	0	0	0	0	-0.1	0	0	0	0	0	0	-0.1
car	-0.6	-0.3	-0.4	0	0	-0.1	0	0	-0.4	-0.1	0	-0.1	0	0	-0.2	-0.4	0	0	0
truck	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.1	0	-0.7	0	0	0
bus	0	0	0	0	0	0	0	0	0	0	-0.1	0	0	-0.3	-0.7	0	0	0	-0.1
train	0	0	0	0	-0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
motorcycle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bicycle	0	-0.2	-0.1	0	0	0	0	0	0	0	0	-0.2	-0.1	0	0	-0.1	0	0	0

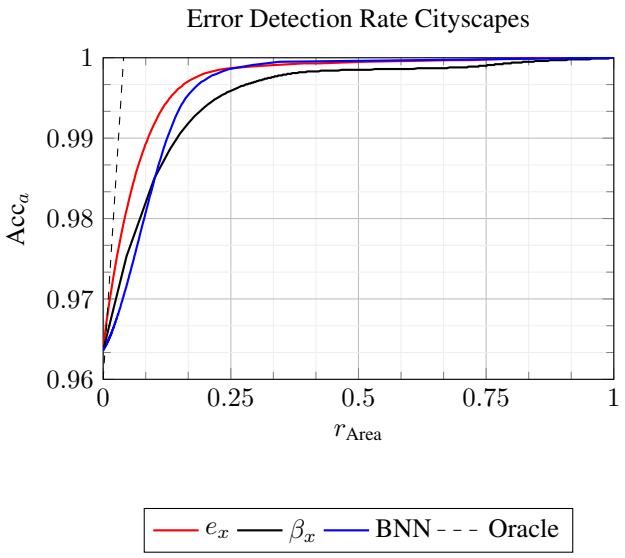


Figure 3. Noise detection experiment on Cityscapes extending Fig. 4 in the main paper.

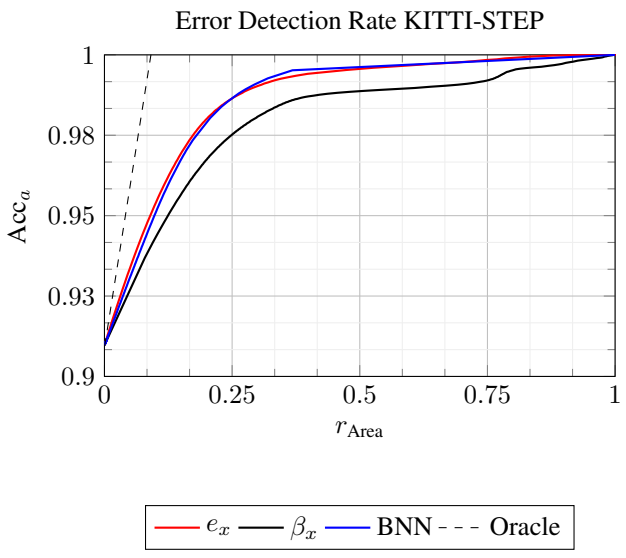


Figure 4. Noise detection experiment on KITTI-STEP extending Fig. 4 in the main paper.

Table 3. Full compensation weight matrix B for *DeepLabv3* trained on Cityscapes.

B_{ij}	*stuff*											*thing*							
	road	*sidewalk*	*building*	*wall*	*fence*	*pole*	*tr. light*	*tr. sign*	*vegetation*	*terrain*	*sky*	*person*	*rider*	*car*	*truck*	*bus*	*train*	*motorcycle*	*bicycle*
road	0	-2.1	-0.1	-0.1	-0.1	-0.1	0	0	-0.2	-0.6	0	-0.3	0	-1.0	-0.1	-0.1	0	0	-0.1
sidewalk	-1.9	0	-0.8	-0.3	-0.3	-0.5	0	0	-0.4	-1.1	0	-0.4	0	-0.4	0	0	0	0	-0.2
building	-0.1	-0.8	0	-0.9	-0.9	-1.9	-0.4	-0.8	-2.4	-0.2	-0.8	-0.9	-0.1	-0.9	-0.3	-0.2	-0.2	-0.1	-0.3
wall	0	-0.3	-0.8	0	-0.6	-0.1	0	0	-0.5	-0.1	0	-0.1	0	-0.1	0	0	0	0	0
fence	0	-0.2	-0.7	-0.6	0	-0.3	0	0	-0.6	-0.2	0	-0.1	0	-0.1	0	0	0	0	-0.1
pole	0	-0.4	-1.7	-0.1	-0.3	0	-0.2	-0.3	-1.3	-0.2	-0.2	-0.2	0	-0.3	0	0	0	0	-0.2
traffic light	0	0	-0.3	0	0	-0.2	0	0	-0.2	0	0	0	0	0	0	0	0	0	0
traffic sign	0	0	-0.7	0	0	-0.3	-0.1	0	-0.3	0	0	0	0	0	0	0	0	0	0
vegetation	-0.1	-0.5	-2.4	-0.6	-0.7	-1.4	-0.3	-0.4	0	-1.1	-0.8	-0.4	-0.1	-0.7	-0.1	-0.1	0	0	-0.1
terrain	-0.4	-1.0	-0.1	-0.1	-0.2	-0.2	0	0	-1.0	0	0	0	0	-0.1	0	0	0	0	-0.1
sky	0	0	-0.7	0	0	-0.2	0	0	-0.8	0	0	0	0	0	0	0	0	0	0
person	-0.2	-0.4	-0.8	-0.1	-0.1	-0.2	0	0	-0.3	0	0	0	-0.4	-0.3	0	0	0	-0.1	-0.2
rider	0	0	0	0	0	0	0	0	0	0	0	-0.3	0	0	0	0	0	-0.1	-0.3
car	-0.9	-0.4	-0.8	-0.1	-0.1	-0.3	0	-0.1	-0.6	-0.1	0	-0.4	-0.1	0	-0.3	-0.2	-0.1	-0.1	-0.2
truck	0	0	-0.2	0	0	0	0	0	0	0	0	0	0	-0.3	0	-0.1	0	0	0
bus	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.1	-0.1	0	-0.1	0	0
train	0	0	-0.1	0	0	0	0	0	0	0	0	0	0	0	0	-0.1	0	0	0
motorcycle	0	0	0	0	0	0	0	0	0	0	0	0	-0.1	-0.1	0	0	0	0	-0.1
bicycle	0	-0.2	-0.2	0	-0.1	-0.2	0	0	-0.1	0	0	-0.2	-0.3	-0.2	0	0	0	-0.1	0

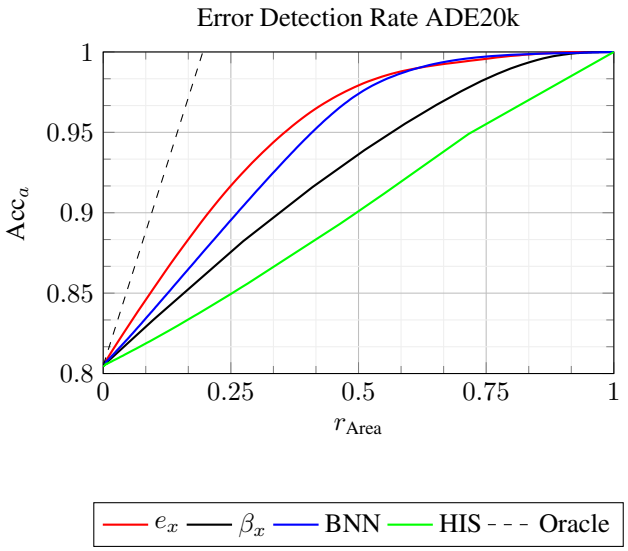


Figure 5. Noise detection experiment on ADE20k extending Fig. 4 in the main paper.

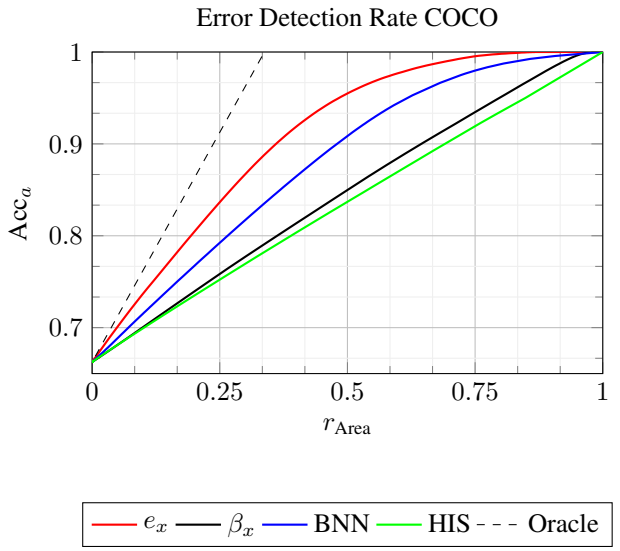


Figure 6. Noise detection experiment on COCO extending Fig. 4 in the main paper.

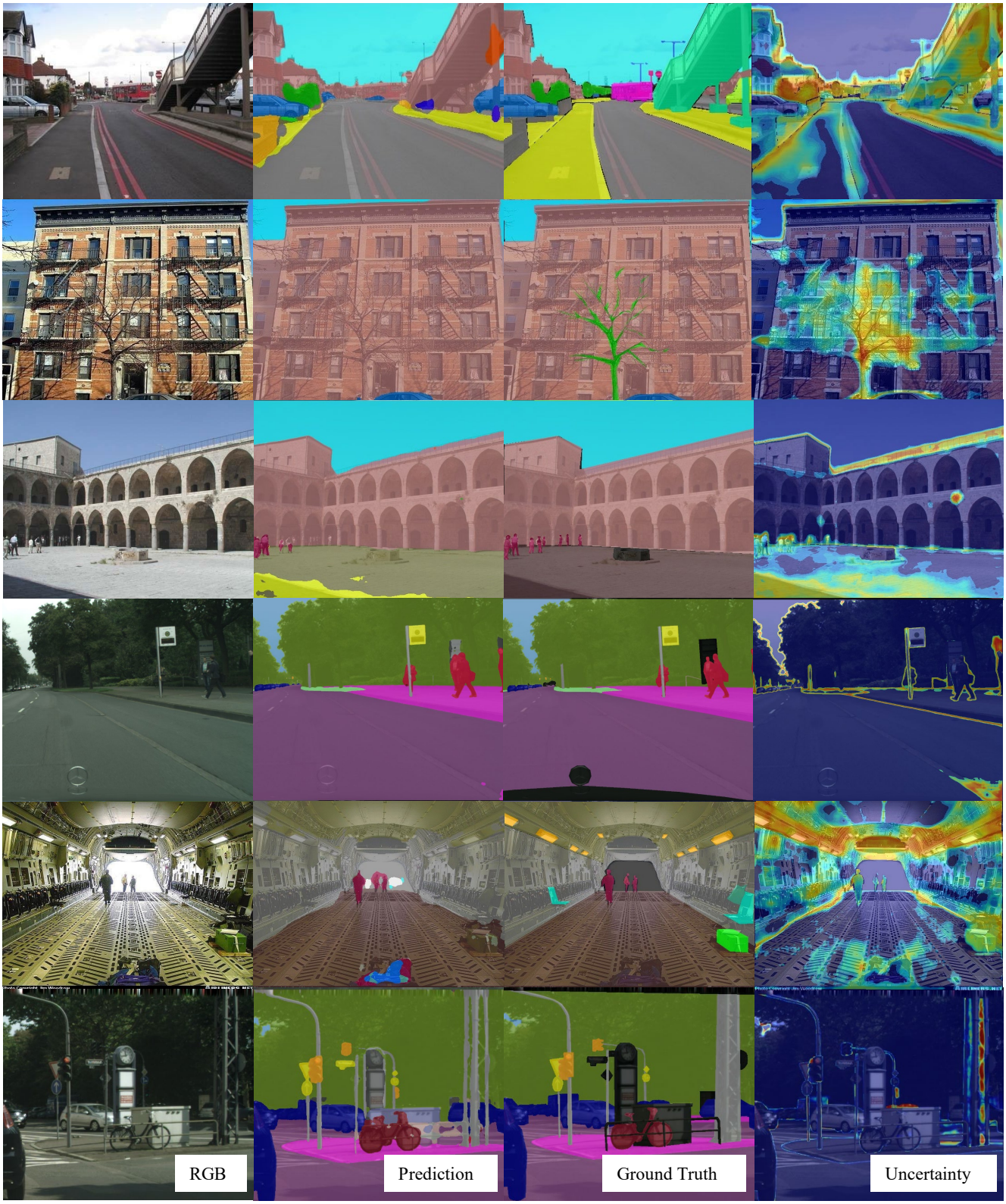
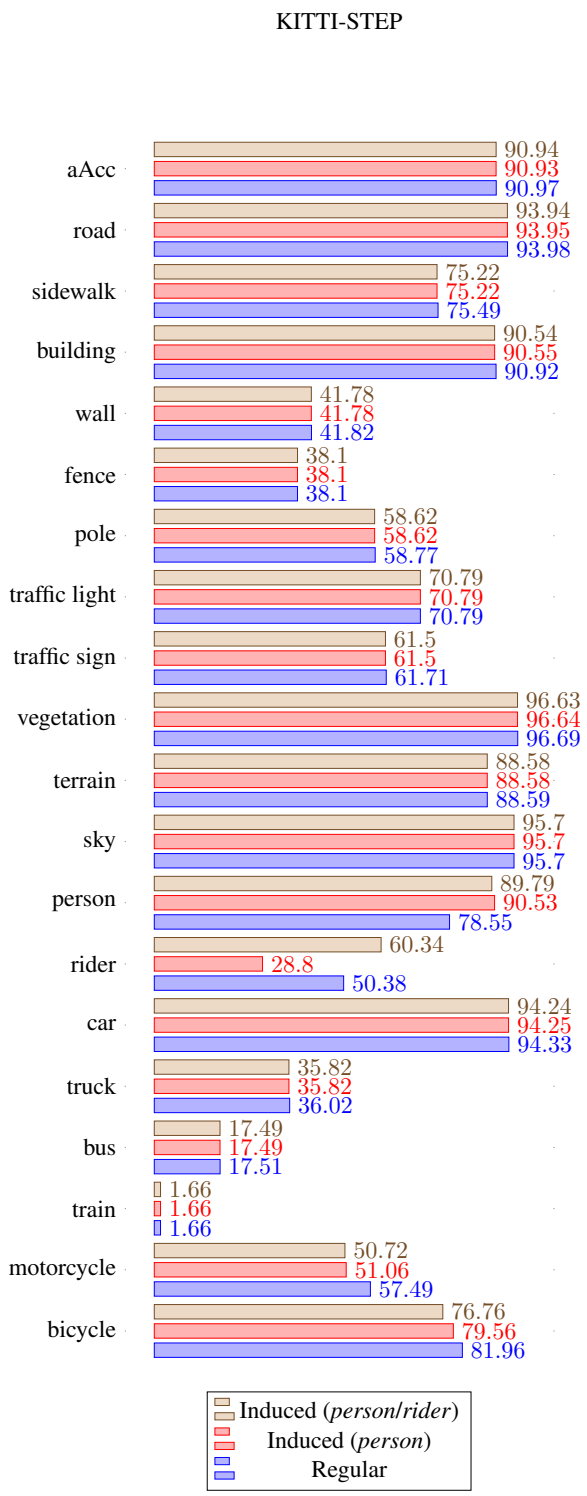
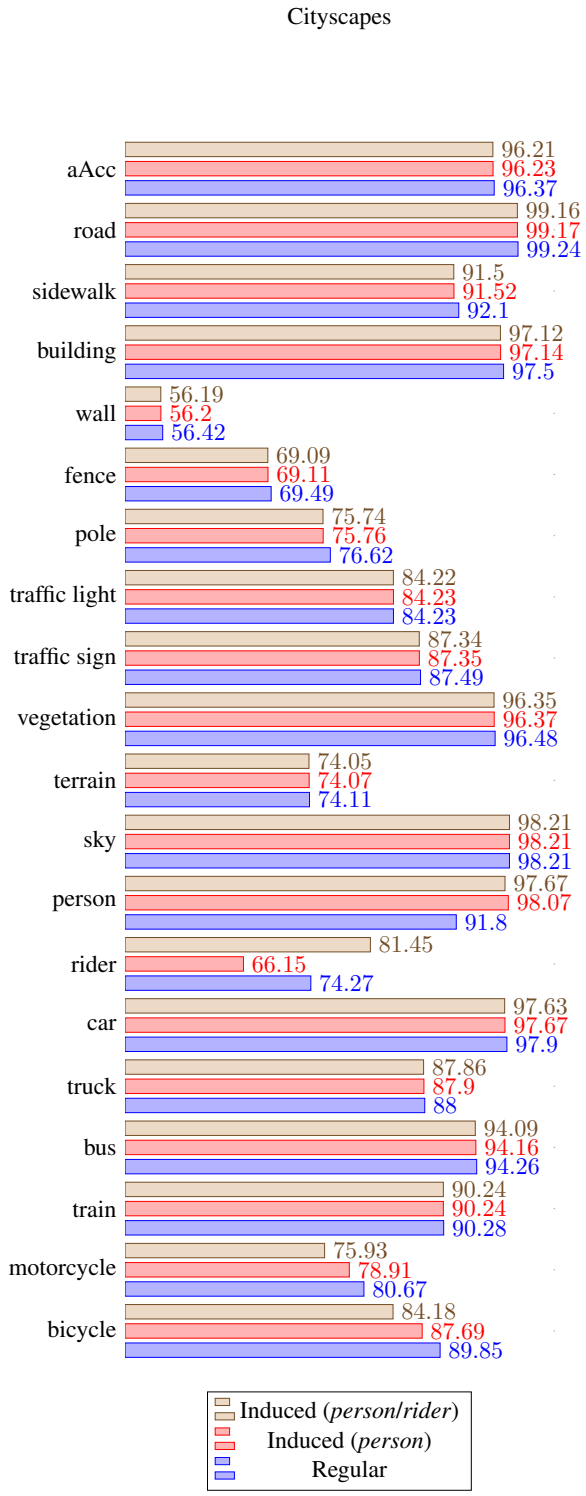


Figure 7. Samples of our uncertainty estimation with *DeepLabv3+*.



(a) Induction Experiments on KITTI-STEP



(b) Induction Experiments on Cityscapes

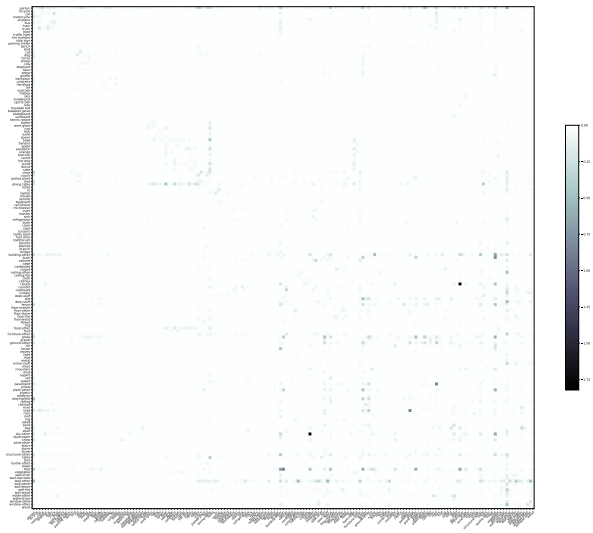
Figure 8. Induction experiments on Cityscapes and KITTI-STEP providing information about all class accuracy metrics to complete Fig. 6 in the main paper. Additionally, the figure presents the metrics, if only one class is induced during inference.

Table 4. Full compensation weight matrix B for *SegFormer* trained on KITTI-STEP.

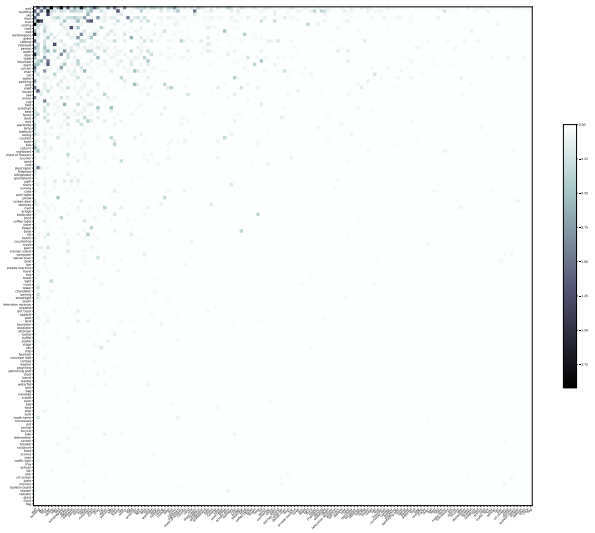
B_{ij}	*stuff*											*thing*							
	road	*sidewalk*	*building*	*wall*	*fence*	*pole*	*tr. light*	*tr. sign*	*vegetation*	*terrain*	*sky*	*person*	*rider*	*car*	*truck*	*bus*	*train*	*motorcycle*	*bicycle*
road	0	-7.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sidewalk	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
building	0	0	0	-1.1	0	-3.3	0	0	0	0	0	0	0	0	0	0	-3.7	-2.2	0
wall	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fence	0	0	0	-0.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pole	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
traffic light	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
traffic sign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
vegetation	0	0	0	-4.0	-5.1	-6.2	-5.9	-5.3	0	-6.2	0	0	0	0	0	0	0	-0.6	0
terrain	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sky	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
person	0	0	0	0	0	0	0	0	0	0	0	0	-2.0	0	0	0	0	0	0
rider	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
car	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
truck	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1.6	0	0	0
bus	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-3.4	0	0	0	0
train	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
motorcycle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bicycle	0	0	0	0	0	0	0	0	0	0	0	0	-4.2	0	0	0	0	0	0

Table 5. Full compensation weight matrix B for *SegFormer* trained on Cityscapes.

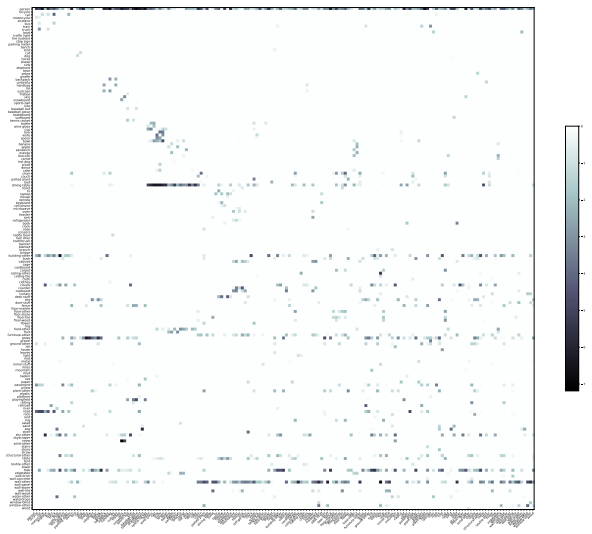
B_{ij}	*stuff*											*thing*							
	road	*sidewalk*	*building*	*wall*	*fence*	*pole*	*tr. light*	*tr. sign*	*vegetation*	*terrain*	*sky*	*person*	*rider*	*car*	*truck*	*bus*	*train*	*motorcycle*	*bicycle*
road	0	-7.1	0	0	0	0	0	0	0	-1.0	0	0	0	0	0	0	0	0	0
sidewalk	0	0	0	0	0	0	0	0	0	-0.2	0	0	0	0	0	0	0	0	0
building	0	0	0	-3.3	-4.2	-7.3	-7.5	-6.8	0	0	0	-5.7	0	0	-2.0	-2.2	-2.9	-3.3	0
wall	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fence	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pole	0	0	0	0	0	0	-2.6	0	0	0	0	0	0	0	0	0	0	0	0
traffic light	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
traffic sign	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
vegetation	0	0	0	-4.0	-5.1	-4.8	-5.5	-2.5	0	-5.7	-7.9	0	0	0	0	0	0	0	0
terrain	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sky	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
person	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rider	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
car	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-2.5	0	0	-0.4	0
truck	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bus	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
train	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
motorcycle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bicycle	0	0	0	0	0	0	0	0	0	0	0	0	-4.6	0	0	0	0	-1.7	0



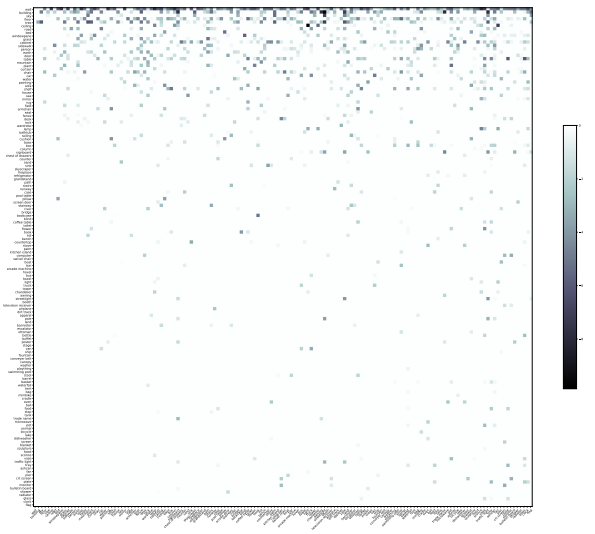
(a) COCO-stuff10k with *DeepLabv3+*.



(b) ADE20k with *DeepLabv3+*.



(c) COCO-stuff10k with *SegFormer*.



(d) ADE20k with *SegFormer*.

Figure 9. Complete compensation weight matrices B for the large-scale datasets ADE20k and COCO-stuff10k for our baseline methods *SegFormer* and *DeepLabv3+*. We recommend to view the plots digital in the PDF version and zooming in to discover details such as class names and value ranges.

Table 6. Confusion Matrix on the training set of Cityscapes after training with *DeepLabv3+*.

%	*stuff*											*thing*							
	road	*sidewalk*	*building*	*wall*	*fence*	*pole*	*tr. light*	*tr. sign*	*vegetation*	*terrain*	*sky*	*person*	*rider*	*car*	*truck*	*bus*	*train*	*motorcycle*	*bicycle*
road	99.4	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0
sidewalk	1.7	95.8	0.5	0.1	0.1	0.2	0.0	0.0	0.3	0.6	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.0	0.1
building	0.0	0.1	97.6	0.0	0.0	0.3	0.0	0.0	1.2	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
wall	0.1	0.8	3.5	89.8	2.0	0.4	0.0	0.0	2.3	0.3	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.0	0.0
fence	0.2	0.9	2.0	1.5	90.4	0.7	0.0	0.0	2.7	0.4	0.0	0.3	0.0	0.3	0.0	0.0	0.0	0.0	0.3
pole	0.2	1.0	8.6	0.2	0.4	81.2	0.6	0.5	5.0	0.3	0.8	0.3	0.0	0.5	0.0	0.0	0.0	0.0	0.3
traffic light	0.0	0.0	7.0	0.0	0.0	3.3	84.3	0.3	4.2	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
traffic sign	0.1	0.0	5.3	0.0	0.2	1.2	0.2	89.8	2.3	0.0	0.4	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0
vegetation	0.0	0.0	1.4	0.0	0.1	0.3	0.0	0.0	97.3	0.2	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
terrain	1.2	3.3	0.3	0.2	0.3	0.3	0.0	0.0	3.8	90.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.1
sky	0.0	0.0	0.5	0.0	0.0	0.1	0.0	0.0	1.3	0.0	98.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
person	0.7	0.7	2.4	0.0	0.2	0.3	0.0	0.0	0.6	0.0	0.0	93.6	0.2	0.5	0.0	0.0	0.0	0.0	0.3
rider	0.5	0.3	1.6	0.0	0.1	0.2	0.0	0.0	0.8	0.0	0.0	3.4	86.1	0.8	0.0	0.0	0.0	1.5	4.5
car	0.5	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0	98.2	0.0	0.0	0.0	0.0	0.0
truck	0.2	0.0	1.1	0.0	0.0	0.1	0.0	0.0	0.4	0.0	0.0	0.1	0.0	1.3	96.3	0.0	0.0	0.0	0.0
bus	0.2	0.0	0.7	0.0	0.0	0.2	0.0	0.0	0.5	0.0	0.0	0.2	0.0	0.3	0.2	97.4	0.0	0.0	0.0
train	0.1	0.0	1.1	0.0	0.0	0.2	0.0	0.0	0.3	0.0	0.0	0.1	0.0	0.3	0.0	0.0	97.6	0.0	0.0
motorcycle	0.4	0.6	1.7	0.0	0.1	0.4	0.0	0.0	0.6	0.1	0.0	0.6	1.3	1.0	0.0	0.0	0.0	91.6	1.5
bicycle	0.5	1.1	2.1	0.0	0.5	1.0	0.0	0.0	0.8	0.2	0.0	1.0	1.6	1.0	0.0	0.0	0.0	0.6	89.5

Table 7. Confusion Matrix on the validation set of Cityscapes after training with *DeepLabv3+*.

%	*stuff*											*thing*							
	road	*sidewalk*	*building*	*wall*	*fence*	*pole*	*tr. light*	*tr. sign*	*vegetation*	*terrain*	*sky*	*person*	*rider*	*car*	*truck*	*bus*	*train*	*motorcycle*	*bicycle*
road	99.1	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0
sidewalk	4.4	92.7	0.6	0.1	0.2	0.4	0.0	0.0	0.4	0.5	0.0	0.3	0.0	0.3	0.0	0.0	0.0	0.0	0.2
building	0.0	0.1	96.8	0.0	0.0	0.5	0.0	0.1	1.7	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
wall	0.2	3.8	19.2	63.9	4.5	0.7	0.0	0.0	5.1	1.1	0.0	0.6	0.0	0.5	0.0	0.0	0.0	0.0	0.3
fence	0.5	1.1	13.5	3.9	73.1	1.3	0.0	0.4	3.4	0.4	0.0	0.5	0.0	0.5	0.3	0.0	0.0	0.1	0.8
pole	0.2	1.2	8.9	0.1	0.7	80.3	0.6	0.5	5.2	0.3	0.5	0.4	0.0	0.6	0.0	0.0	0.0	0.0	0.4
traffic light	0.0	0.0	6.2	0.0	0.0	3.3	84.1	0.2	5.5	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
traffic sign	0.3	0.1	5.4	0.0	0.9	1.6	0.2	87.8	2.7	0.0	0.1	0.2	0.0	0.2	0.1	0.0	0.0	0.0	0.2
vegetation	0.0	0.1	1.5	0.0	0.1	0.4	0.0	0.0	96.9	0.4	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
terrain	1.1	8.0	0.5	0.5	0.6	0.5	0.0	0.0	13.4	74.8	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.2
sky	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.0	1.0	0.0	98.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
person	0.5	0.6	2.8	0.0	0.1	0.6	0.0	0.0	0.7	0.0	0.0	92.2	0.7	0.7	0.0	0.0	0.0	0.0	0.7
rider	0.3	0.3	1.6	0.0	0.0	0.2	0.0	0.0	1.2	0.0	0.0	13.4	75.4	0.9	0.0	0.1	0.0	0.8	5.6
car	0.4	0.0	0.5	0.0	0.0	0.1	0.0	0.0	0.3	0.0	0.0	0.1	0.0	98.0	0.2	0.0	0.0	0.0	0.1
truck	0.3	0.1	2.7	0.0	0.0	0.2	0.0	0.0	0.9	0.0	0.3	0.0	0.0	4.6	89.1	1.5	0.0	0.0	0.0
bus	0.3	0.0	1.3	0.0	0.4	0.2	0.0	0.0	1.3	0.0	0.0	0.1	0.0	0.7	0.3	95.1	0.0	0.0	0.0
train	0.3	0.0	4.3	0.0	0.2	0.7	0.1	0.2	2.3	0.0	0.1	0.0	0.0	0.8	0.2	1.0	89.4	0.0	0.0
motorcycle	0.3	1.0	2.1	0.0	1.2	0.9	0.0	0.0	0.6	0.0	0.0	3.5	2.8	1.7	0.0	0.0	0.4	80.2	5.3
bicycle	0.4	0.8	2.2	0.0	0.5	0.8	0.0	0.0	1.0	0.2	0.0	1.5	2.3	0.9	0.0	0.0	0.0	0.5	88.8

Table 8. Confusion Matrix on the training set of KITTI-STEP after training with *DeepLabv3+*.

%	*stuff*											*thing*							
	road	*sidewalk*	*building*	*wall*	*fence*	*pole*	*tr. light*	*tr. sign*	*vegetation*	*terrain*	*sky*	*person*	*rider*	*car*	*truck*	*bus*	*train*	*motorcycle*	*bicycle*
road	98.2	0.8	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0
sidewalk	2.2	95.2	0.3	0.0	0.0	0.7	0.0	0.0	0.4	0.5	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.0	0.2
building	0.0	0.2	97.4	0.0	0.0	0.6	0.0	0.0	1.0	0.0	0.2	0.2	0.0	0.1	0.0	0.0	0.0	0.0	0.0
wall	0.1	2.0	1.0	92.3	0.6	1.0	0.0	0.0	1.8	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.2
fence	0.3	0.7	0.3	0.4	94.7	0.6	0.0	0.0	1.9	0.6	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0
pole	1.8	2.4	8.0	0.1	0.7	67.6	0.8	0.9	12.3	1.2	2.3	0.2	0.0	0.9	0.2	0.0	0.1	0.0	0.2
traffic light	0.0	0.0	1.0	0.0	0.0	4.2	86.2	0.3	7.7	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
traffic sign	0.9	0.1	1.8	0.0	0.0	1.5	0.0	89.6	4.9	0.1	0.4	0.0	0.0	0.2	0.1	0.0	0.0	0.0	0.0
vegetation	0.0	0.1	0.6	0.0	0.1	0.6	0.0	0.3	97.4	0.3	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
terrain	1.4	0.7	0.0	0.0	0.4	0.6	0.0	0.0	2.4	94.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
sky	0.0	0.0	0.2	0.0	0.0	0.3	0.0	0.0	1.3	0.0	98.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
person	1.7	2.0	3.2	0.2	0.0	0.3	0.0	0.0	0.7	0.0	0.0	90.5	0.2	0.3	0.0	0.0	0.0	0.0	0.7
rider	1.2	0.9	1.2	0.0	0.0	1.5	0.0	0.0	1.7	0.0	0.0	0.8	85.3	0.7	0.3	0.0	0.0	0.0	6.2
car	0.9	0.3	0.3	0.0	0.0	0.2	0.0	0.0	0.4	0.1	0.0	0.0	0.0	97.4	0.0	0.2	0.0	0.0	0.0
truck	0.4	0.3	0.3	0.0	0.0	0.3	0.0	0.0	0.7	0.0	0.1	0.0	0.0	0.4	88.9	8.2	0.0	0.0	0.2
bus	0.4	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.6	0.0	0.0	0.5	1.2	95.9	0.1	0.0	0.5
train	0.0	0.0	1.3	0.0	1.2	0.8	0.1	0.1	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	96.1	0.0	0.0
motorcycle	0.5	1.5	1.9	0.0	0.0	0.8	0.0	0.0	1.9	0.0	0.0	0.2	0.0	0.6	0.0	0.0	0.0	91.7	0.7
bicycle	0.7	3.1	1.3	0.1	0.0	0.6	0.0	0.0	0.7	0.1	0.0	0.6	0.8	0.0	0.0	0.1	0.0	0.0	91.6

Table 9. Confusion Matrix on the validation set of KITTI-STEP after training with *DeepLabv3+*.

%	*stuff*											*thing*							
	road	*sidewalk*	*building*	*wall*	*fence*	*pole*	*tr. light*	*tr. sign*	*vegetation*	*terrain*	*sky*	*person*	*rider*	*car*	*truck*	*bus*	*train*	*motorcycle*	*bicycle*
road	94.7	3.8	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.8	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0
sidewalk	16.4	75.5	0.5	0.0	0.2	0.8	0.0	0.0	1.0	5.1	0.0	0.2	0.0	0.3	0.0	0.0	0.0	0.0	0.0
building	0.0	0.7	90.4	0.6	2.1	0.7	0.0	0.0	3.4	0.0	1.3	0.2	0.0	0.2	0.2	0.0	0.0	0.0	0.1
wall	0.2	5.1	23.7	35.7	17.3	1.9	0.0	0.0	14.5	0.6	0.0	0.0	0.0	0.8	0.0	0.0	0.0	0.0	0.0
fence	1.7	9.5	7.9	0.6	36.5	2.3	0.0	0.1	26.6	12.8	0.0	0.2	0.0	1.3	0.1	0.0	0.0	0.0	0.2
pole	2.1	2.6	6.0	0.0	0.3	58.0	0.7	0.7	20.4	2.8	5.2	0.3	0.0	0.4	0.0	0.0	0.0	0.0	0.3
traffic light	0.0	0.0	2.9	0.0	0.0	5.4	71.7	3.2	14.6	0.0	2.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
traffic sign	2.4	0.2	7.0	0.0	0.0	6.2	1.6	63.8	12.5	1.2	3.2	0.8	0.0	0.7	0.1	0.0	0.0	0.0	0.0
vegetation	0.0	0.0	0.6	0.0	0.1	0.7	0.0	0.0	96.7	1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
terrain	2.1	1.7	0.0	0.0	0.4	1.1	0.0	0.0	5.1	89.4	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0
sky	0.0	0.0	0.7	0.0	0.0	0.7	0.0	0.0	2.2	0.0	96.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
person	2.3	3.8	7.2	0.0	0.0	0.8	0.0	0.0	2.3	0.4	0.0	78.9	1.1	0.8	0.0	0.0	0.0	0.0	2.2
rider	0.9	2.0	4.8	0.0	0.0	0.9	0.0	0.0	3.4	0.0	0.0	27.7	47.2	1.4	0.0	0.0	0.0	0.0	11.6
car	1.1	0.4	1.0	0.0	0.0	0.2	0.0	0.0	0.7	0.1	0.0	0.0	0.0	94.7	1.0	0.3	0.0	0.0	0.0
truck	1.0	0.6	12.3	0.0	0.0	0.7	0.0	2.7	9.6	0.3	0.2	0.0	0.0	37.8	33.3	1.2	0.0	0.0	0.0
bus	1.4	0.1	20.2	0.0	0.0	1.5	0.2	0.3	5.5	0.0	0.0	0.3	0.0	20.1	8.1	29.8	12.4	0.0	0.0
train	0.2	1.9	61.8	0.0	0.4	0.8	0.0	1.5	0.4	0.0	0.0	0.0	0.0	0.5	24.2	7.1	1.1	0.0	0.0
motorcycle	0.0	6.1	3.2	0.0	0.0	1.9	0.0	0.0	0.0	0.0	0.0	5.9	2.4	0.2	0.0	0.0	0.0	72.4	7.8
bicycle	1.3	4.6	4.6	0.0	0.0	0.7	0.0	0.0	2.8	0.3	0.0	2.3	1.1	1.3	0.2	0.0	0.0	0.1	80.6

References

[1] Mina Ghadimi Atigh, Julian Schoep, Erman Acar, Nanne van Noord, and Pascal Mettes. Hyperbolic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022. 2

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 2

[3] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 1

[4] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3

[5] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *Proceedings of the International Conference on Learning Representations*, 2017. 1

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2016. 2

[7] Sheng Liu, Kangning Liu, Weicheng Zhu, Yiqiu Shen, and Carlos Fernandez-Granda. Adaptive early-learning correction for segmentation from noisy annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022. 3

[8] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 3

[9] Jishnu Mukhoti and Yarin Gal. Evaluating bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018. 2

[10] Martin Thoma. A survey of semantic segmentation. *arXiv preprint arXiv:1602.06541*, 2016. 3

[11] Mark Weber, Jun Xie, Maxwell Collins, Yukun Zhu, Paul Voigtlaender, Hartwig Adam, Bradley Green, Andreas Geiger, Bastian Leibe, Daniel Cremers, Aljosa Osep, Laura Leal-Taixe, and Liang-Chieh Chen. Step: Segmenting and tracking every pixel. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. 3