Appendices

As part of the supplementary materials for this paper, we present our hyper-parameters and show more visual and quantitative results as an extension to the ones shown in the paper. We also discuss our ablations to highlight the positive effect of InverseForm loss over cross-entropy loss.

A. Implementation details

In this section, we discuss our training setup in detail for every experiment. Depending on the experiment, we use 1 to 4 NVIDIA Tesla-V100 GPUs for training. For our baseline HR-Net experiments, we implemented HRNet-w18 and HRNet-w48 within our framework by using their codebase [46]. The inverse-transformation network has a multilayer perceptron architecture shown in Figure 3 of the paper. The number of outputs of the network corresponds to the degrees of freedom for the transformation matrix. It's set to be 6 when we consider euclidean distance, and 8 when we consider geodesic distance. We plug our InverseForm loss into a multitude of single-task and multi-task methods for various datasets as discussed in the paper. We will discuss these in more detail in the upcoming subsections.

A.1. NYUD-v2 dataset

Inverse-transformation network: To train the inverse-transformation network for NYUD-v2, we first extract boundaries from the segmentation labels using sobel filters [2], and binary threshold them at 0.5 intensity. Then we resize the images to 672x672 pixels and tile the images into 9 tiles(addressed in Section B.1) of 224x224 pixels. To train the network for 100 epochs, we use a pre-trained spatial-transformer [21] to generate transforms for each image and feed batch-sizes of 1 image(9 tiles) to the inverse-transformation network, with a learning rate of 0.1 decaying exponentially at 0.95 decay.

Vanilla HRNet: We augment the input images by random flips(p=0.5), random scaling(ratio in $\{1, 1.2, 1.5\}$) and cropping back to the original dimension of 480x640 pixels. We then train both HRNet backbones with stochastic gradient-descent for 484 epochs with a batch size of 12, setting the initial learning rate to 0.02 and a poly learning rate schedule with exponent 0.9. For InverseForm, we train these models within our framework described in Figure 4 of the paper. The weights β and γ in Equation 6 for the final loss are both set at 0.5. All experiments use a single Tesla-V100 GPU.

MTI-Net/PAD-Net HRNet: To reproduce MTI-Nets and PAD-Net algorithm, we essentially used the code given by MTI-Nets [44], and used their augmentations and hyperparameters for a fair comparison. For boundaries, they use balanced cross-entropy loss(weighted at 0.95 for positive pixels) and hence, use a higher initial weight of $\beta = 50$. We then train this network by adding the InverseForm loss term to the balanced cross-entropy loss. We add InverseForm to this loss term weighted at $\gamma = 0.5$. All experiments use a single Tesla-V100 GPU.

SA-Gates using ResNet-101: For reproducing SA-Gates scores, we use their repository [35] and hyper-parameters to train the baseline ResNet-101 model with DeepLab-V3+. We then train the same model with our framework shown in Figure 4 of the paper and loss function in Equation 6 of the paper, keeping all other hyper-parameters unchanged for a fair comparison. The weights β and γ in Equation 6 for the final loss are both set at 0.5. All experiments use 4 Tesla-V100 GPUs. For qualitative comparison in Figure 7, we use predictions from the baseline model we train without InverseForm loss, and compare them to the predictions output from the baseline trained with InverseForm loss.

A.2. PASCAL dataset

Inverse-transformation network: For training the inverse-transformation network for PASCAL, we extract boundaries from the segmentation labels and threshold at 0.5 intensity. We resize the images to 672x672 pixels and tile the images into 9 tiles(addressed in Section B.1) of 224x224 pixels. To train the network for 100 epochs, we use a random homography transform generator, implemented by [52], to generate transforms for each image and feed batch-sizes of 1 image(9 tiles) to the inverse-transformation network, with a learning rate of 0.1 decaying exponentially at 0.95 decay.

MTI-Net/ASTMT: To generate MTI-Nets and ASTMT scores, we use the code made publicly available by the same authors [44] [31] and reproduce their results. To ensure fair comparison, we use the same augmentation strategy and hyper-parameters to train these models by adding InverseForm loss. To detect edges, they use balanced cross-entropy loss(weighted at 0.95 for positive pixels) and thus, have a higher initial weight of $\beta = 50$. We train the baseline networks by adding the InverseForm loss term, weighted at $\gamma = 0.5$, to the balanced cross-entropy loss. All experiments use a single Tesla-v100 GPU.

A.3. Cityscapes dataset

Inverse-transformation network: To generate the inverse-transformation network for Cityscapes, we follow the same boundary detection step as done for the above two datasets. Then, we resize the images to 672x1344 pixels and tile the images into 18 tiles(addressed in Section B.1) of 224x224 pixels. To train the network for 100 epochs, we use the spatial transformer based homography generator used for NYUD-v2, to generate transforms for each image. A batch-size of 1 image(18 tiles) with it's transform is fed to the inverse-transformation network, with a learning rate of 0.1 decaying exponentially at 0.95 decay.

HRNet-48 with OCR and/or Hierarchical multi-scale attention: To reproduce the numbers for both Hierarchical Multi-Scale Attention [40] and HRNet-48 with OCR [49] models, we use the code made available publicly by the HMS authors [40]. We train the networks using 8 GPUs instead of 16(mentioned in the original paper). Hence, we train with a smaller batch-size of 8. Apart from the batch-size, we use the same hyper-parameters and augmentation schemes used by the HMS paper. We train the same models within our framework shown in Figure 4 of the paper and loss function in Equation 6 of the paper, keeping the hyper-parameters constant for a fair comparison. The weights β and γ in Equation 6 for the final loss are both set at 0.5, as done for earlier datasets. To qualitatively compare with this work in Figure 6, we select images inferred from their uploaded state-of-the-art model on Cityscapes test, and compare with the predictions from the same backbone, but trained with InverseForm. To compare qualitatively with SegFix [50] in Figure 6, we select one of the Cityscapes test predictions uploaded to drive by the authors of this work, and use HRNet-OCR-InverseForm model to generate a prediction for the same image.

WRN with Gated-SCNN [39]: To train GSCNN-InverseForm models in Table 4, we use their publicly available code to train a Gated-SCNN model and add InverseForm loss with a weight γ of 0.5 to their existing boundary-loss. We train both with and without the auto-labeled coarse annotations. To qualitatively compare with this work in Figure 6, we select one of the Cityscapes val predictions uploaded to drive by the authors of this work, and use GSCNN-InverseForm model trained with fine annotations to generate a prediction for the same image.

B. Extra experiments

B.1. Effect of tiling:

As discussed in Section 3.2, we tile the predicted and ground-truth boundary maps before calculating InverseForm loss. The tiling helps preserve local resolution while retaining the global context. We run experiments on HRNet-w18 and HRNet-w48 backbones with and without our InverseForm loss using the setup described in Figure 4. We resize NYU-Depth-v2 boundary maps to 672×672 pixels and train both networks with a varying range of tiling dimensions. As shown in Table B.1, we find that splitting the image into 9 tiles of 224×224 pixels results in the highest mIoU score. We hypothesize that tiles larger than this size lack enough local context to quantify a distance measure. Tiles smaller than this size will negatively affect the inverse-transformation network's training, as there would not be enough global context to measure the homography transform between input. We use the same resizing and tiling dimension for PASCAL images. For Cityscapes images, we resize the image to 672×1344 and tile it to 18 tiles of 224×224 pixels.

Tiling dim.	N_{tiles}	HRNet-w48	HRNet-w18
672	1	46.94	34.24
336	4	46.98	34.46
224	9	**47.42**	**34.79**
112	36	47.14	34.63

Table B.1: Performance comparison(mIoU) of different tiling dimensions for HRNet-w48 and HRNet-w18 on NYUD-v2.

B.2. Multi-task performance on NYUD-v2

Our results in Table 1 of the paper only show segmentation mIoU for models trained in Multi-task settings. In this section, we present our multi-task performance on NYUD-v2 and compare it to the baseline used (MTI-Nets [44]). The experimental setup is exactly the same as Table 1; the following results are an extension of the previous table for it's multi-task performance. To calculate multi-task metric, we use semantic segmentation and depth as main tasks, and treat edges and surface normals as auxiliary tasks to ensure a fair comparison with the MTI-Nets paper [44]. Our boundary loss is added to the already existing cross-entropy loss term for the boundary task. We compare the effect of using our loss to the single-task numbers reported in their paper (and also reproduced by us). As seen from Table B.2, training with InverseForm loss shows clear improvement in segmentation mIoU as well as depth rmse. We also consistently outperform the MTI-Net baseline on the multi-task metric.

Network	Backbone	Tasks	Seg mIoU (↑)	Depth rmse (↓)	$\Delta_m(\%)$ (↑)
HRNet-w18	HRNet-w18	S	33.18	0.667	0.00
PAD-Net	HRNet-w18	S+D	32.80	0.660	-0.02
PAD-Net	HRNet-w18	S+D+E+N	33.10	0.655	0.52
PAD-Net w/ InverseForm	HRNet-w18	S+D+E+N	**34.70**	**0.641**	**4.24**
MTI-Net	HRNet-w18	S+D	35.12	0.62	6.40
MTI-Net	HRNet-w18	S+D+E+N	37.49	0.607	10.91
MTI-Net w/ InverseForm	HRNet-w18	S+D+E+N	**38.71**	**0.594**	**13.80**

Table B.2: Quantitative results on NYU-Depth-v2 using an HRNet-w18 backbone in a multi-task learning setting using our loss. Tasks, S:Semantic segmentation, E:Edge detection, N:Surface normal estimation, D: Depth estimation. Consistent improvement in multi-task performance is visible over all baselines.

B.3. Studying the effect of varying blending parameters for InverseForm loss and cross-entropy loss

In this section, we discuss the effect of the blending parameter for InverseForm term as described in Equation 6 of the paper:

$$L_{total} = L_{xe}(\boldsymbol{y_{pred}}, \boldsymbol{y_{gt}}) + \beta L_{xe}(\boldsymbol{b_{pred}}, \boldsymbol{b_{gt}}) + \gamma L_{if}(\boldsymbol{b_{pred}}, \boldsymbol{b_{gt}})$$

We run experiments for a multi-task setting using ASTMT on two ResNet backbones with DeepLab-V3+ head, to model two tasks: Semantic segmentation and boundary detection. The boundary detection task uses a positive-balanced cross-entropy loss.

Network	Tasks	γ	β	mIoU (↑)
SE-ResNet-50	S	-	-	67.70
	S+E	0.0	50	68.38
	S+E	0.1	50	69.57
	S+E	0.5	50	**70.15**
	S+E	1.0	50	69.64
	S+E	0.5	0	68.43
SE-ResNet-101	S	-	-	69.76
	S+E	0.0	50	71.04
	S+E	0.1	50	71.93
	S+E	0.5	50	**72.18**
	S+E	1.0	50	72.03
	S+E	0.5	0	71.21

Table B.3: Ablation study on PASCAL showing effect of the blending parameter for InverseForm loss γ and for cross-entropy loss β. Tasks, S: Semantic Segmentation, E: Edge Detection. We observe that a weight of 0.5 for InverseForm loss is optimal, and the effect of InverseForm loss is best observed when used with cross-entropy loss.

We add our InverseForm loss function to the weighted XE loss term, and show mIoU results for different parameters. β is the weight for boundary cross-entropy loss term L_{xe} and γ is the weight for boundary InverseForm loss term L_{if}.

Notice from Table B.3, that a positive weight of 0.5 for InverseForm loss is optimal, and the effect of InverseForm loss is best observed when used with cross-entropy loss. Hence, we use both loss terms for our experiments.

We also show visual results comparing the ResNet-101 model trained with cross-entropy loss with and without InverseForm in Figure B.1. Notice the highlighted regions in boundary predictions for model trained without InverseForm loss. These regions contain artifacts which aren't present in boundary predictions for the model trained with InverseForm loss. The InverseForm models segmentation outputs are improved in these regions which show lesser boundary artifacts.

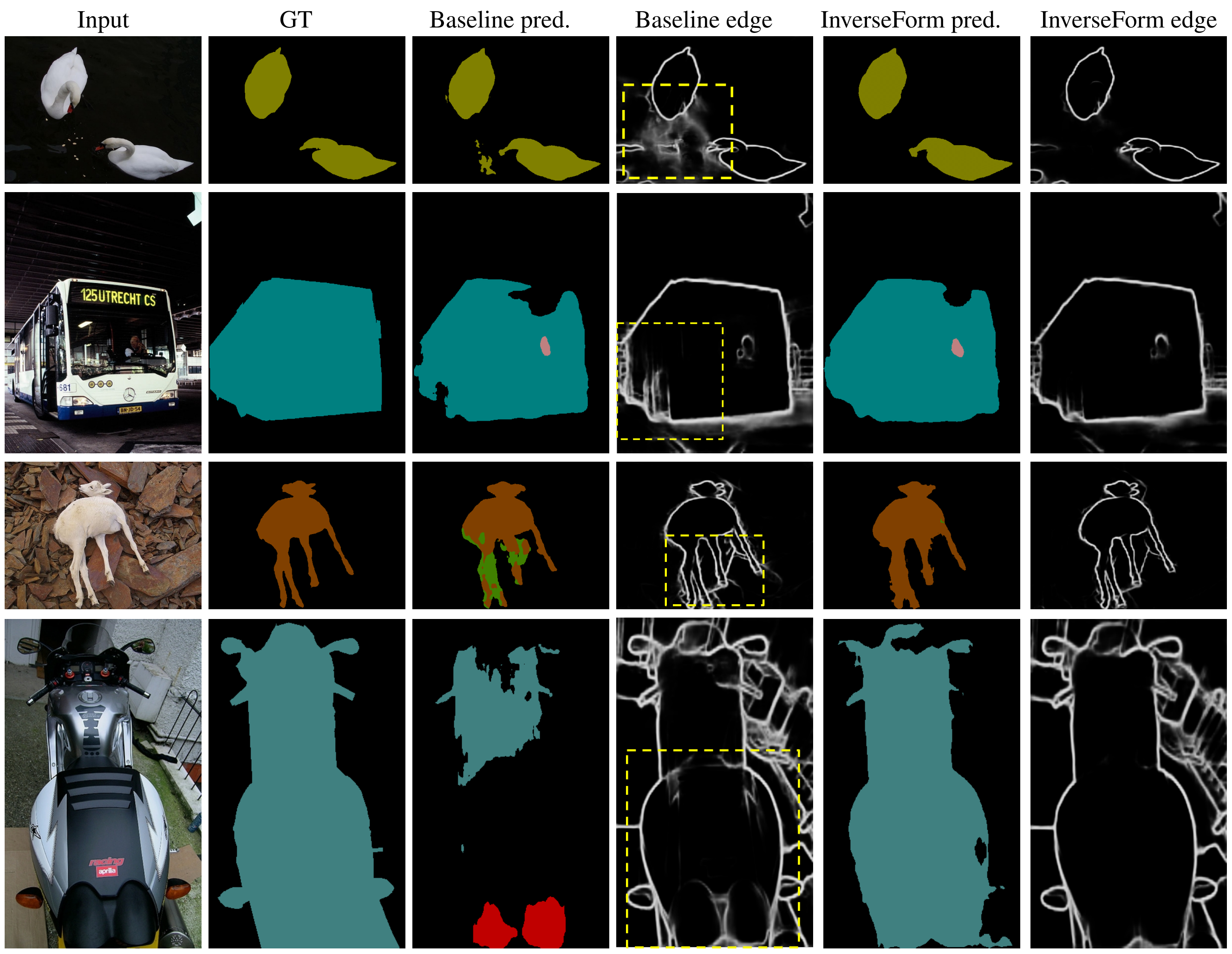

Figure B.1: Qualitative results on PASCAL showing visual effect of training SE-ResNet-101 baseline with and without InverseForm loss. Notice the improvement in boundary predictions leads to better semantic segmentation predictions.

C. Visual results

C.1. Visual results for NYUD-v2

In this section, we show more predicted images from our state-of-the-art results on NYUD-v2. These are achieved with a ResNet-101 backbone and DeepLab-V3+ head, with SA-Gates, trained with InverseForm loss. These images are continued from Figure 7 in the paper, and their quantitative reults are available in Table 2 of the paper.

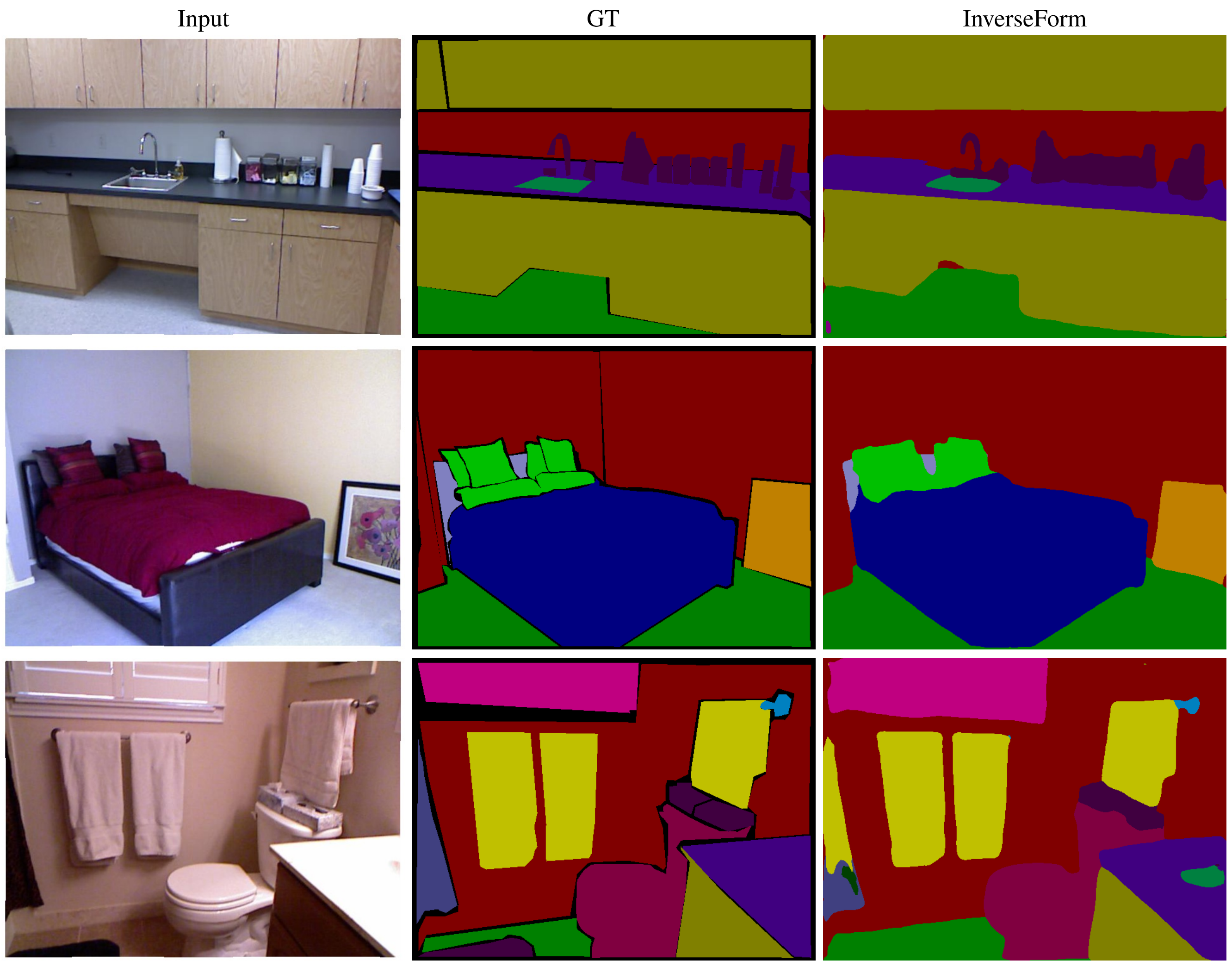

Figure C.1: Qualitative results on NYU-Depth-v2 showing visual effect of training SA-Gates [35] model with InverseForm loss.

C.2. Multi-task visual results for PASCAL

In Figure C.2, we present some qualitative results from the ResNet-101 backbone with ASTMT trained with InverseForm loss for boundary detection. The model is trained on the PASCAL split used in this paper and ASTMT [31]. The quantitative scores for this model are presented in Table 3 of the paper.

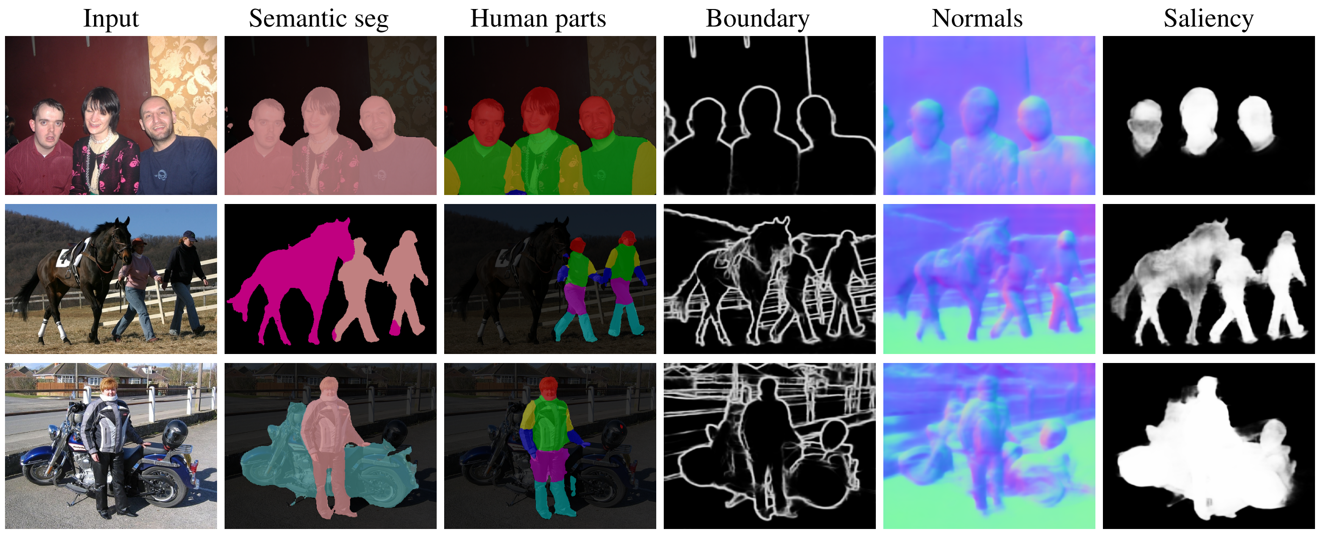

Figure C.2: Multitask results on PASCAL showing visual effect of training ResNet-101 with DeepLab-V3+ using InverseForm loss.