Exploring Relational Context for Multi-Task Dense Prediction

Supplementary Material

David Bruggemann, Menelaos Kanakis, Anton Obukhov, Stamatios Georgoulis, Luc Van Gool
ETH Zurich
{brdavid, kanakism, obukhova, georgous, vangool}@vision.ee.ethz.ch

A. Training Details

In this section, we describe the training setup. For consistency, all experiments in the paper were repeated five times using the pipeline detailed below.

Data augmentation. We augment input images during training by random scaling with values between 0.5 and 2.0, random cropping to input size (425×560 for NYUD-v2—we use the cropped version of [3]—and padded to 512×512 for PASCAL-Context), random horizontal flipping and random color jitter. Image intensities are standardized. Depth labels are corrected for scaling and surface normal labels are corrected for horizontal flipping.

Task losses. The total loss of the multi-task network with parameters θ is a weighted sum of losses (for tasks $n \in \{1, ..., N\}$):

$$\mathcal{L}_{total}(\theta) = \sum_{n=1}^{N} \omega_n \mathcal{L}_n(\theta) \quad (1)$$

For semantic segmentation and human parts segmentation we use a cross-entropy loss (loss weights $\omega_n = 1$ and $\omega_n = 2$ respectively), for saliency estimation a balanced cross-entropy loss ($\omega_n = 5$), for depth estimation a $\mathcal{L}_1$ loss ($\omega_n = 1$), for surface normal estimation a $\mathcal{L}_1$ loss with unit vector normalization ($\omega_n = 10$) and for boundary detection a weighted cross-entropy loss ($\omega_n = 50$). For boundary detection, the positive pixels are weighted with 0.8 and the negative pixels with 0.2 on NYUD-v2, while on PASCAL-Context the weights are 0.95 and 0.05. ω_n for each task was determined through a logarithmic grid search over candidate values with single-task networks.

The auxiliary predictions A_n are trained with a cross-entropy loss using the same loss weights as above. However, the auxiliary head backpropagation is stopped from updating parameters of the main network.

Optimization hyperparameters. All backbones are initialized with ImageNet pretrained weights. We use Stochastic Gradient Descent (SGD) with momentum of 0.9 and weight decay of 0.0005 to optimize the model parameters. The initial learning rate is determined through a logarithmic grid search (..., 0.002, 0.005, 0.01, 0.02, ...), with the option of having a 10 times higher learning rate for the heads *vs.* the backbone. The initial value is decayed during training according to a 'poly' learning rate schedule [1]. For all experiments, we use a minibatch size of 8 and train for 40000 iterations.

Context type search. The architecture distribution parameters α are initialized with zeros. We use an Adam optimizer [4] to update them, with learning rate 0.0005 (no weight decay, no learning rate scheduler). The update occurs in the same round of backpropagation as the regular model parameters (single-level optimization). Over the course of training, the Gumbel-Softmax temperature λ is annealed linearly from 1.0 to 0.05 (following [9]). Also, to ensure a fair candidate context type selection, we disable learnable affine parameters of the last batch normalization of every context type attention mechanism.

As discussed in Sec. 3.3, we use entropy (H) regularization to control the sampling variance during the architecture search. Specifically, we calculate the mean entropy of the architecture parameter (α)-distributions over all Context Pooling (CP) blocks, scale it with a weight ω_H, and add it to the total loss.

$$\mathcal{L}_{search}(\theta, \alpha) = \sum_{n=1}^{N} \omega_n \mathcal{L}_n(\theta, \alpha) + \frac{\omega_H}{N^2} \sum_{j=1}^{N^2} H(\alpha_j) \quad (2)$$

j indexes the CP blocks. The scaling factor ω_H follows a linear schedule during the search, from -0.02 to 0.06. We found that this provides an adequate balance between candidate exploration and exploitation. For a given CP block j, architecture search is terminated prematurely if the difference between the two largest values of α_j exceeds 0.3. One candidate is then sampled using argmax (*i.e.*, α_j becomes a one-hot vector).

After concluding five runs of the architecture search, we determine the final configuration by choosing the context type receiving the most votes over the five runs in each CP

Model	SemSeg ↑	Depth ↓	Normal ↓	Bound ↑
HRNet18, [7]	33.18	0.667	-	-
HRNet18, ours	38.02	0.610	20.94	76.22
HRNet48, [7]	45.70	0.547	-	-
HRNet48, ours	45.87	0.540	20.09	77.34

Table B-1. NYUD-v2 single task performances of HRNetV2-W18-small (HRNet18) and HRNetV2-W48 (HRNet48) models [8]. We compare the performances obtained using our implementation with the numbers published in [7].

block. Ultimately, this final configuration is retrained five times.

B. Implementation Verification

We verify the implementation of our pipeline by comparing HRNet single task performances with the numbers published in [7]. Table B-1 shows that the baselines trained with our pipeline outperform those of [7].

For implementing the various distillation modules in Table 1, we used the code provided by the authors whenever possible, and otherwise followed the information provided in the papers closely. For MTI-Net [7], we used the authors' model code within our pipeline.

Finally, we attempted to reimplement the full PSD [11] network based on a ResNet-50 backbone (as suggested in the original paper), but were unable to obtain competitive results.

C. Relational Context Schematics

Fig. C-1 depicts the different relational context types used in this work schematically. We use a $1{\times}1$ Conv-BN-ReLU layer as the learned non-linear transform. For all contexts except global, the similarity function is $\mathrm{sim}(q_i, k_j) = \exp(\frac{q_i k_j^\intercal}{d_k})$ (which corresponds to $\mathrm{softmax}$). For the global context, it is simply $\mathrm{sim}(q_i, k_j) = q_i k_j^\intercal$.

D. Label Context: Regression Tasks

In this section, we discuss how the label space of regression tasks can be partitioned into distinct regions for label context formation, as mentioned in Sec. 3.2.3. Regression tasks can be easily reformulated as classification tasks by binning the continuous ground truth values. However, the discretization scheme has to be tailored towards each task separately to obtain satisfactory performance.

For depth prediction, our approach is inspired by [5]. Specifically, we divide the range of depth values into 40 logarithmic bins, accounting for the fact that the estimation error for larger depth values is naturally larger. During training, we learn a classifier to assign the pixels to the bins. During evaluation, we use a soft-weighted-sum inference: Every bin is represented by its mean depth in log space. A weighted sum of bins (weight = prediction score) is used as the final prediction.

For surface normal estimation, we use the triangular coding technique of [10]. First, a codebook is learned with k-means. The codewords form a Delaunay triangulation cover on the unit sphere. Any surface normal can thus be expressed as a weighted combination of the three codewords marking its triangle. During training, we learn a classifier to predict those codeword weights. Following [10], we choose 40 codewords ($\widehat{=}$ 40 classes). Evaluation consists of two steps: (1) Find the triangle with maximum total probability. (2) Use the probabilities of the three codewords of that triangle to reconstruct the surface normal.

To verify the above discretization schemes, we trained single task models accordingly, and compare them to the regression models in Fig. D-1. The figure shows that the performance of classification—while slightly worse than regression—is satisfactory for both depth and surface normal estimation. The same conclusion can be drawn from a qualitative comparison, shown in Fig. D-2.

Model	SemSeg ↑	Depth ↓	Normal ↓	Bound ↑	Δ_m [%] ↑
Single task	38.02	0.6104	20.94	76.22	0.00
T-label, GT	46.71	0.5202	18.16	76.06	12.67
S-label, GT	47.71	0.5160	17.87	78.18	14.55

Table E-1. NYUD-v2 comparison of the performance upper bound of T-label and S-label context, using ground truth (GT) spatial region maps $A_n^{(GT)}$ (see Sec. 3.2.3).

E. Label Context: Performance Upper Bound

To estimate the potential of label context for multi-modal distillation, we conduct experiments using ground truth label regions. Instead of predicting the spatial maps A_n from the input image (see Sec. 3.2.3), we directly use the ground truth data $A_n^{(GT)}$ to partition the label space into distinct regions. This provides an upper bound for the performance of label context distillation. Table E-1 shows the results for both T-label and S-label context: The performance increases greatly (with the exception of T-label context for boundary detection), confirming that label region grouping is highly effective for multi-modal distillation.

F. Context Type Search Reliability

We consider the context type selection during architecture search as a rater decision. Since we repeat each run five times, we can evaluate the intra-rater reliability: The agreement among the five selected context types in all CP blocks.

The most intuitive way to quantify agreement is through percentage agreement (*i.e.*, counting the fraction of times a

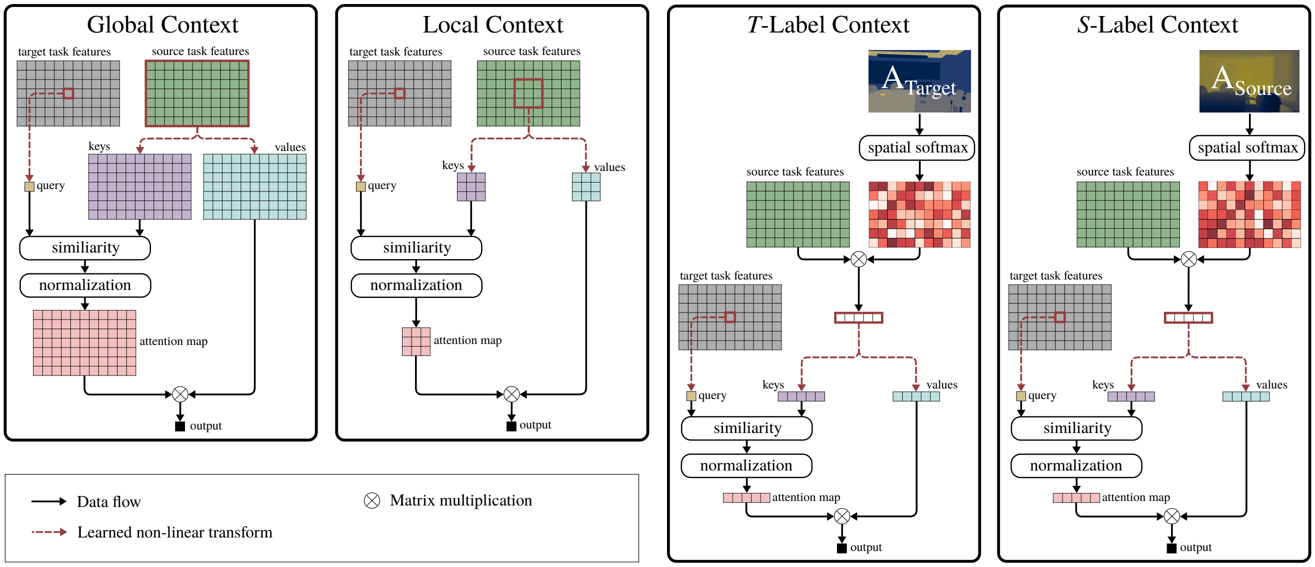

Figure C-1. Schematics of the different relational context types. The grids represent individual pixels (channels not shown), the attention mechanism is shown for one target pixel (framed in red box) respectively. Normalization is applied over all pixels of the attention map. A_* are the auxiliary predictions, as depicted in Fig. 2a.

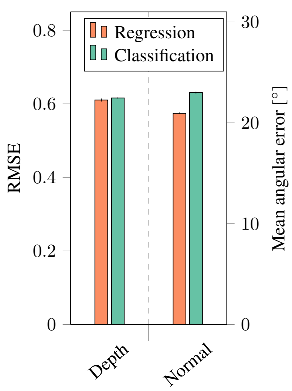

Figure D-1. Performance comparison of single task depth and surface normal estimation models, using either a regression or classification framework. Their similar performance confirms that we can exploit the classification scheme to form high-quality label regions for the label context.

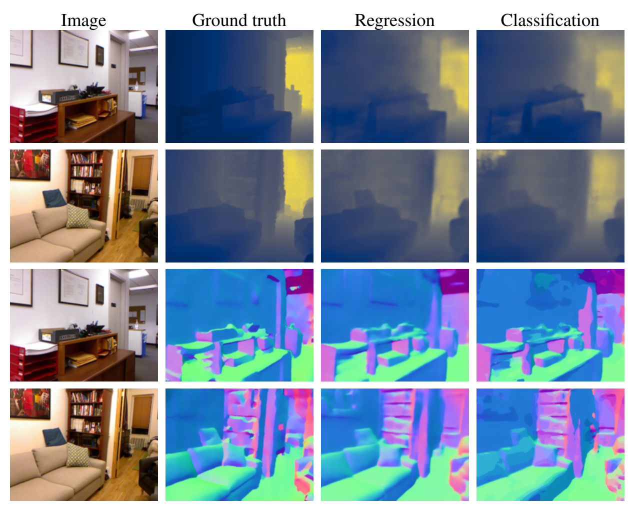

Figure D-2. Qualitative NYUD-v2 comparison of regression and classification schemes for depth (top two rows) and surface normal (bottom two rows) estimation. Classification achieves satisfactory results on both tasks.

pair of runs agree on a decision). However, this measure does not take into account that agreement may happen purely due to chance. We thus report also Light's kappa [6], which is an agreement score calculated by averaging Cohen's kappa [2] over all pairs of runs. Kappa statistics are corrected for chance agreement, with the drawback that their interpretation is less intuitive. A value of 0 indicates no agreement, -1 indicates perfect disagreement, and 1 perfect agreement. We obtain an overall percentage agreement of 71.2% and a Light's kappa of 0.48 for the search on NYUD-v2 with a HRNet18 backbone.

We emphasize that reliability is not strictly necessary for an effective search algorithm. If there is no dominating choice of context type (*e.g.*, none of the options lead to significant performance gain), then even a valid search algorithm is expected to be unreliable. In such cases, introducing a tie-breaking auxiliary objective could help promote convergence (*e.g.*, a resource loss).

G. How Important is Self-Attention in ATRC?

The permutation testing results of Sec. 4.4 can be utilized to partly address this question. We conclude there that self-attention constitutes the most important distillation module for 3 out of 4 investigated tasks. However, other cross-task connections contribute significantly too. To investigate further, we provide in Table G-1 the performance of ATRC without self-attention. The multi-task performance Δ_m for this model is 0.87% (*vs*. 1.56% for the full ATRC), outperforming the single task configuration. This confirms that, even though self-attention is vital according to permutation testing, the cross-task distillation modules are able to provide a substantial performance boost on their own.

Distillation module	SemSeg ↑		Depth ↓		Normal ↓		Bound ↑		Δ_m [%] ↑
	mean	std.	mean	std.	mean	std.	mean	std.	
None (single task baseline)	38.02	0.14	0.6104	0.0041	20.94	0.08	76.22	0.07	0.00
None (multi-task baseline)	36.35	0.26	0.6284	0.0034	21.02	0.06	76.36	0.05	-1.89
ATRC (ours)	38.90	0.43	0.6010	0.0046	20.48	0.02	76.34	0.12	**1.56**
ATRC (no self-attention)	38.19	0.46	0.6032	0.0038	20.55	0.05	76.22	0.10	0.87

Table G-1. Effect of removing the self-attention blocks on NYUD-v2 with a HRNet18 backbone. First three lines correspond to the numbers reported in Table 1.

References

[1] 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. *TPAMI*, 40(4):834–848, 2017.

[2] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[3] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*, 2014.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[5] Bo Li, Yuchao Dai, and Mingyi He. Monocular depth estimation with hierarchical fusion of dilated cnns and soft-weighted-sum inference. *Pattern Recognition*, 83:328–339, 2018.

[6] Richard J Light. Measures of response agreement for qualitative data: some generalizations and alternatives. *Psychological bulletin*, 76(5):365, 1971.

[7] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. In *ECCV*, 2020.

[8] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2020.

[9] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: Stochastic neural architecture search. In *ICLR*, 2019.

[10] Bernhard Zeisl, Marc Pollefeys, et al. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014.

[11] Ling Zhou, Zhen Cui, Chunyan Xu, Zhenyu Zhang, Chaoqun Wang, Tong Zhang, and Jian Yang. Pattern-structure diffusion for multi-task learning. In *CVPR*, 2020.