

Supplementary Materials

Simon Vandenhende¹
simon.vandenhende@kuleuven.be
 Stamatios Georgoulis²
stamatios.georgoulis@vision.ee.ethz.ch
 Luc Van Gool¹²
vangool@vision.ee.ethz.ch

¹ PSI-ESAT
 KU Leuven
 Leuven, Belgium
² CVL/TRACE
 ETH Zurich
 Zurich, Switzerland

1 Cityscapes

The encoder is a ResNet-50 model with dilated convolutions [9], pre-trained on ImageNet. We use a PSP module [10] for the task-specific decoders. Every input image is rescaled to 512 x 256 pixels. We upsample the output of the PSP decoders back to the input resolution during training. The outputs are upsampled to 2048 x 1024 pixels during testing. The semantic segmentation task is learned with a weighted pixel-wise cross-entropy loss. We reuse the approach from [9] for the instance segmentation task, i.e. we consider the proxy task of regressing each pixel to the center of the instance it belongs to. The depth estimation task is learned using an L1 loss. The losses are normalized to avoid having the loss of one task overwhelm the others during training. The hyperparameters were optimized with a grid search procedure to ensure a fair comparison across all compared approaches.

Single-task models We tested batches of size 4, 6 and 8, poly learning rate decay vs step learning rate decay with decay factor 10 and step size 30 epochs, and Adam (initial learning rates 2e-4, 1e-4, 5e-5, 1e-5) vs stochastic gradient descent with momentum 0.9 (initial learning rates 5e-2, 1e-2, 5e-3, 1e-3). This accounts for 48 hyperparameter settings in total. We repeated this procedure for every single task (semantic segmentation, instance segmentation and monocular depth estimation).

Baseline multi-task network We train with the same set of hyperparameters as before, i.e. 48 settings in total. We calculate the multi-task performance in accordance with [9]. In particular, the multi-task performance of a model m is measured as the average per-task performance increase/drop w.r.t. the single task models b :

$$\Delta_m = \frac{1}{T} \sum_{i=1}^T (-1)^{l_i} (M_{m,i} - M_{b,i}) / M_{b,i}, \quad (1)$$

where $l_i = 1$ if a lower value means better for measure M_i of task i , and 0 otherwise.

Branched multi-task network We reuse the hyperparameter setting with the best result for the baseline multi-task network. The branched multi-task architectures that were used for the quantitative evaluation on Cityscapes are shown in Fig. 4.

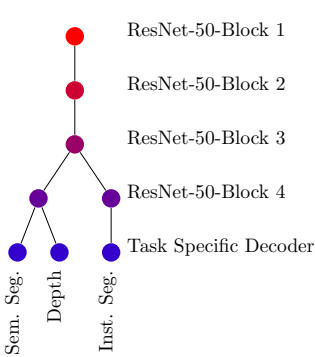


Figure 1: Ours - 1

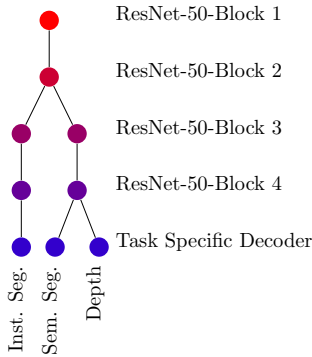


Figure 2: Ours - 2

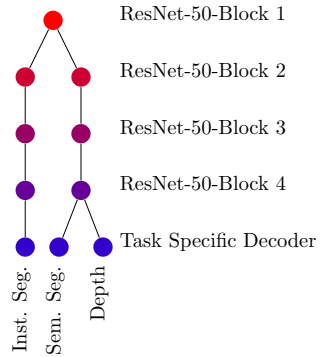


Figure 3: Ours - 3

Figure 4: Branched multi-task networks on Cityscapes that were generated by our method.

Cross-stitch networks / NDDR-CNN We insert a cross-stitch/NDDR unit after every ResNet block. We also tried to leave out the cross-stitch/NDDR unit after the final ResNet block, but this decreased performance. We tested two different initialization schemes for the weights in the cross-stitch/NDDR units, i.e. $\alpha = 0.8$, $\beta = 0.1$ and $\alpha = 0.9$, $\beta = 0.05$. The model weights were initialized from the set of the best single-task models above. We found that the Adam optimizer broke the initialization and refrained from using it. The best results were obtained with stochastic gradient descent with initial learning rate $1e-3$ and momentum 0.9. As also done in [14, 15], we set the weights of these units to have a learning rate that is 100 times higher than the base learning rate.

MTAN We re-implemented the MTAN model [16] using a ResNet-50 backbone based on the code that was made publicly available by the authors. We obtained our best results when using an Adam optimizer. Other hyperparameters were set in accordance with our other experiments.

2 Taskonomy

We reuse the setup from [17]. All input images were rescaled to 256×256 pixels. We use a ResNet-50 encoder and replace the last stride 2 convolution by a stride 1 convolution. A 15-layer fully-convolutional decoder is used for the pixel-to-pixel prediction tasks. The decoder is composed of five convolutional layers followed by alternating convolutional and transposed convolutional layers. We use ReLU as non-linearity. Batch normalization is included in every layer except for the output layer. We use Kaiming He’s initialization for both encoder and decoder. We use an L1 loss for the depth (D), edge detection (E) and keypoint detection (K) tasks. The scene categorization task is learned with a KL-divergence loss. We report performance on the scene categorization task by measuring the overlap in top-5 classes between the predictions and ground truth.

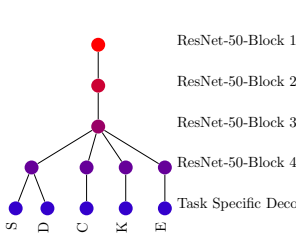


Figure 5: Ours - 1

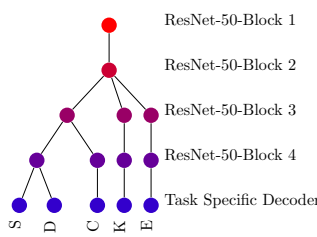


Figure 6: Ours - 2

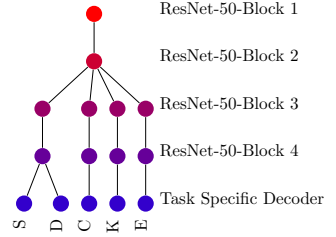


Figure 7: Ours - 3

Figure 8: Task groupings generated by our method on the Taskonomy dataset.

The multi-task models were optimized with task weights $w_s = 1, w_d = 1, w_k = 10, w_e = 10$ and $w_c = 1$. Notice that the heatmaps were linearly rescaled to lie between 0 and 1. During training we normalize the depth map by the standard deviation.

Single-task models We use an Adam optimizer with initial learning rate $1e-4$. The learning rate is decayed by a factor of 10 after 80000 iterations. We train the model for 120000 iterations. The batch size is set to 32. No additional data augmentation is applied. The weight decay term is set to $1e-4$.

Baseline multi-task model We use the same optimization procedure as for the single-task models. The multi-task performance is calculated using Eq. 1.

Branched multi-task models We use the same optimization procedure as for the single-task models. The architectures that were generated by our method are shown in Fig. 8. Fig. 12 shows the architectures that are found when using the task grouping method from [5]. We show some of the predictions made by our third branched multi-task network in Fig. 16 for the purpose of qualitative evaluation.

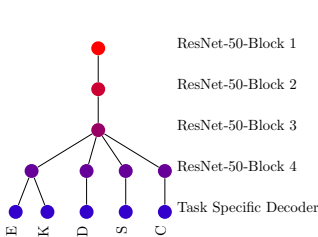


Figure 9: FAFS - 1

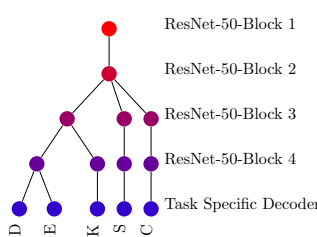


Figure 10: FAFS - 2

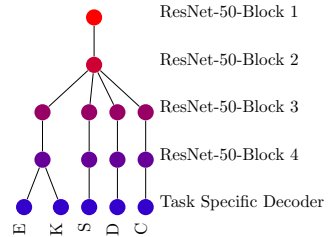


Figure 11: FAFS - 3

Figure 12: Task groupings generated on the Taskonomy dataset using the FAFS method from [5].

Cross-stitch networks / NDDR-CNN We reuse the hyperparameter settings that were found optimal on Cityscapes. Note that, these are in agreement with what the authors reported in their original papers. The weights of the cross-stitch/NDDR units were initialized with $\alpha = 0.8$ and $\beta = 0.05$.

MTAN Similar to the other models, we reused the hyperparameter settings that were found optimal on Cityscapes.

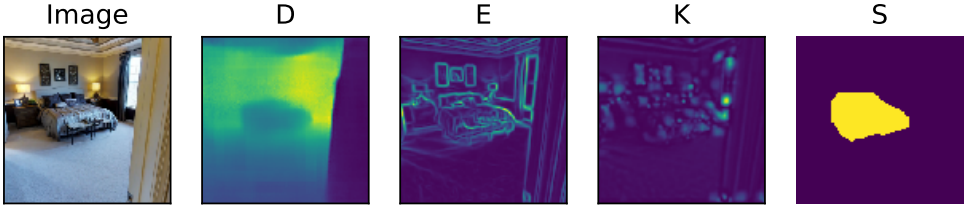


Figure 13: Example - 1

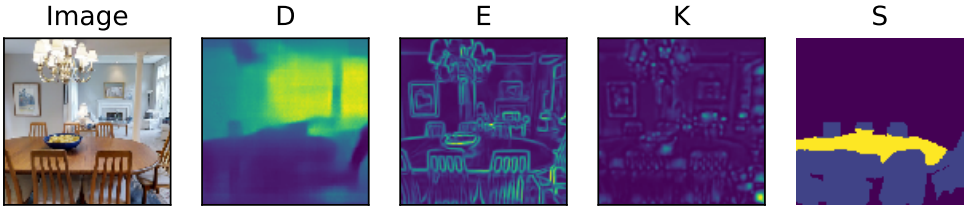


Figure 14: Example - 2

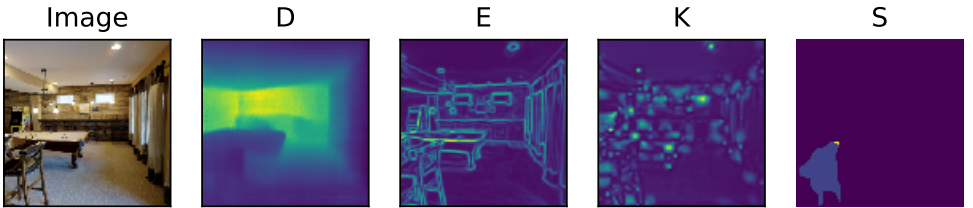


Figure 15: Example - 3

Figure 16: Predictions made by our branched multi-task network on images from the Taskonomy test set.

3 CelebA

We reuse the thin- ω model from [8]. The CNN architecture is based on the VGG-16 model [8]. The number of convolutional features is set to the minimum between ω and the width of the corresponding layer in the VGG-16 model. The fully connected layers contain $2 \cdot \omega$ features. We train the branched multi-task network using stochastic gradient descent with momentum 0.9 and initial learning rate 0.05. We use batches of size 32 and weight decay 0.0001. The model is trained for 120000 iterations and the learning rate divided by 10 every 40000 iterations. The loss function is a sigmoid cross-entropy loss with uniform weighing scheme.

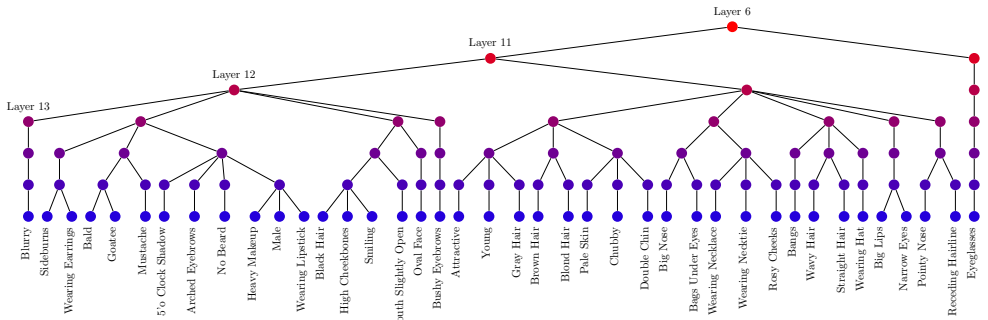


Figure 17: Grouping of 40 person attribute classification tasks on CelebA in a thin VGG-16 architecture.

4 Computational Analysis

We provide an analysis to identify the computational costs related to the different steps when calculating the task affinity scores. We adopt the notation from the main paper. The following three steps can be identified:

- Train N single task networks. It is possible to use a subset of the available training data to reduce the training time. We verified that using a random subset of 500 train images on Cityscapes resulted in the same task groupings.
- Compute the RDM matrix for all N networks at D pre-determined layers. This requires to compute the features for a set of K images at the D pre-determined layers in all N networks. The K images are usually taken as held-out images from the train set. We used $K = 500$ in our experiments. In practice this means that computing the image features comes down to evaluating every model on K images. The computed features are stored on disk afterwards. The RDM matrices are calculated from the stored features. This requires to calculate $N \times D \times K \times K$ correlations between two feature vectors (can be performed in parallel). We conclude that the computation time is negligible in comparison to training the single task networks.
- Compute the RSA matrix at D locations for all N tasks. This requires to calculate $D \times N \times N$ correlations between the lower triangle part of the $K \times K$ RDM matrices. The computation time is negligible in comparison to training the single task networks.

We conclude that the computational cost of our method boils down to training N single task networks plus some overhead. Notice that cross-stitch networks [14] and NDDR-CNNs [15] also pre-train a set of single-task networks first, before combining them together using a soft parameter sharing mechanism. We conclude that our method only suffers from minor computational overhead compared to these methods.

References

- [1] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *CVPR*, pages 3205–3214, 2019.

- [2] 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*, 37(9):1904–1916, 2015.
- [3] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.
- [4] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *CVPR*, 2019.
- [5] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *CVPR*, 2017.
- [6] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *CVPR*, pages 1851–1860, 2019.
- [7] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016.
- [8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
- [9] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
- [10] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.