

Supplementary Material: Automated Search for Resource-Efficient Branched Multi-Task Networks

David Bruggemann

brdavid@vision.ee.ethz.ch

Menelaos Kanakis

kanakism@vision.ee.ethz.ch

Stamatios Georgoulis

georgous@vision.ee.ethz.ch

Luc Van Gool

vangool@vision.ee.ethz.ch

Computer Vision Lab

ETH Zurich

Switzerland

1 Training Settings

In this section, we describe the training setup used for experiments on PASCAL-Context. On NYUD-v2, the exact same setup was used, except that the number of training iterations was halved. Training and evaluation code for this project was written using the `PyTorch` library [8].

Data augmentation. We augment input images during training by random scaling with values between 0.5 and 2.0 (in increments of 0.25), random cropping to input size (which was fixed to 512×512 for full-resolution PASCAL-Context) and random horizontal flipping. Image intensities are rescaled to the $[-1, 1]$ range.

Task losses. For semantic segmentation and human parts segmentation we use a cross-entropy loss (loss weights $\omega_t = 1$ and $\omega_t = 2$ in Equation 3 of the main text, respectively), for saliency estimation a balanced cross-entropy loss ($\omega_t = 5$), for depth estimation a \mathcal{L}_1 loss ($\omega_t = 1$), for surface normal estimation a \mathcal{L}_1 loss with unit vector normalization ($\omega_t = 10$) and for edge detection a weighted cross-entropy loss ($\omega_t = 50$). For edge detection, the positive pixels are weighted with 0.95 and the negative pixels with 0.05 on PASCAL-Context, while on NYUD-v2 the weights are 0.8 and 0.2. ω_t for each task was found by conducting a logarithmic grid search over candidate values with single-task networks.

Optimization hyperparameters. Model weights θ are updated using Stochastic Gradient Descent (SGD) with momentum of 0.9 and weight decay 0.0001. The initial learning rate is set to 0.005 and decayed during training according to a ‘poly’ learning rate policy [11]. For the architecture distribution parameters α , we use an Adam optimizer [14] with learning rate 0.01 and weight decay 0.00005. We use a batchsize of 8 and 16 for ResNet-50 and MobileNetV2, respectively.

Architecture search. We update the supergraph sequentially for each task. Before the architecture search, we ‘warm up’ the supergraph by training each operation’s model weights

θ (initialized with ImageNet weights) on the corresponding task only for 2000 iterations. The architecture distribution parameters α are initialized with zeros. During the search, we alternatively train α on 20% of the data and θ on the other 80%. This cycle is repeated until θ has received 40000 updates. Over the course of training, the Gumbel-Softmax temperature τ is annealed linearly from 5.0 to 0.1. Importantly, we use the batch-specific statistics for batch normalization during the α update phase and reset the batch statistics before training θ after every architecture change. Furthermore, to equalize the scale of candidate operations for the search, we disable learnable affine parameters in the last batch normalization of every operation. Finally, the momentum of the θ -optimizer is reset after every change to the architecture.

Branched network training. After the architecture search, the resulting branched network is retrained from scratch for 40000 iterations. The encoder network weights are initialized with ImageNet weights. For all operations that are shared between several tasks, we divide the learning rate by the number of tasks sharing, since those operations receive more updates.

2 Implementation of Baselines

For FAFS [8], we pre-train a fully shared network on all the tasks and then calculate the task groupings in all layers greedily (starting from the last) according to the task affinity measure described in [8]. Note that for edge detection, we use the loss instead of the optimal dataset F-measure to determine sample difficulty. Finding branching structures via the BMN approach [8] involved training separate single-task networks and computing the Representational Similarity Analysis matrix from the resulting task-specific feature maps, exactly as described in the paper. Various branching structures can be found by exhaustively searching candidates among a reduced pool, containing all possible structures below a specified MAdds value. To keep the comparison fair, we trained the branched structures resulting from BMN and FAFS in exactly the same setting as ours.

We implemented cross-stitch networks [20], NDDR-CNN [20] and MTAN [8] based on the code provided by the authors and information given in the papers. We use a similar training setup as the one described for our method, however we conducted a logarithmic grid search over learning rates for each baseline individually. For cross-stitch networks, applying one unit per feature tensor (as opposed to channel-wise) yielded more stable results. The weights of cross-stitch- and NDDR-CNN-units are initialized with $\alpha = 0.8$ and $\beta = \frac{0.2}{T-1}$, where T is the number of tasks. Both methods are applied on the fully pre-trained single-task networks. For MTAN with the MobileNetV2 backbone, we change the 3×3 convolutions in the attention modules to depthwise separable convolutions. In general, all ReLU activations are replaced with ReLU6 for MobileNetV2.

3 Implementation Verification

To show that our implementations of DeepLabv3+ with the above mentioned backbones perform competitively for the tasks of interest, we compare in Table S1 our single-task performances on PASCAL-Context with published results in [8]. Note that a direct comparison is inconclusive even though the architectures are analogous, as the results in [8] are obtained with different training setups. Nevertheless, the numbers demonstrate that our single-task

Backbone	SemSeg \uparrow	PartSeg \uparrow	Sal \uparrow	Norm \downarrow	Edge \uparrow
MobileNetV2, [10]	62.10	54.88	66.30	14.88	69.50
MobileNetV2, ours	65.11	57.54	65.41	13.98	69.50
ResNet-50, [10]	68.30	60.70	65.40	14.61	72.70
ResNet-50, ours	70.43	63.93	67.34	13.39	74.10

Table S1: DeepLabv3+ performance in a single-task setting on PASCAL-Context using either MobileNetV2 or ResNet-50 as a backbone. We compare the performance obtained using our implementation with the results published in [10].

Model	MAdds \downarrow	SemSeg \uparrow	PartSeg \uparrow	Sal \uparrow	Norm \downarrow	Edge \uparrow	Δ_m [%] \uparrow
Single	345.7B	70.43	63.93	67.34	13.39	74.10	-
Shared	154.6B	68.24	62.18	65.16	14.98	71.90	-4.80
BMTAS-1	199.0B	68.17	62.36	65.64	14.09	72.20	-3.20
BMTAS-2	225.8B	66.92	62.93	65.82	13.70	72.90	-2.56
BMTAS-3	298.2B	69.58	64.36	66.68	13.65	73.00	-1.00

Table S2: Comparison of our method with simple baselines on PASCAL-Context using a ResNet-50 backbone. The resource loss weights λ for BMTAS were 2e-2, 5e-3 and 1e-3 respectively.

networks represent a strong baseline for comparison.

4 Complementary Results

Due to space limitations, the performances of our method and simple baselines for a ResNet-50 backbone on PASCAL-Context (plotted on the right in Figure 2 of the main text) are presented here in Table S2. For this setting, we choose not to report scores for cross-stitch networks [10], NDDR-CNN [10] and MTAN [10] since we were unable to obtain competitive performances for those approaches, despite the extensive learning rate grid-search.

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. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.
- [2] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3205–3214, 2019.
- [3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [4] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning

- with attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1871–1880, 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5334–5343, 2017.
- [6] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1860, 2019.
- [7] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.
- [8] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
- [9] Simon Vandenhende, Stamatios Georgoulis, Bert De Brabandere, and Luc Van Gool. Branched multi-task networks: deciding what layers to share. *arXiv preprint arXiv:1904.02920*, 2019.