 DOI: 10.3217/978-3-85125-663-5-30

Efficient Multi-Task Learning of Semantic Segmentation and Disparity Estimation*

Robert Harb[1] and Patrick Knöbelreiter[1]

***Abstract*— We propose a jointly trainable model for semantic segmentation and disparity map estimation. In this work we utilize the fact that the two tasks have complementary strength and weaknesses. Traditional depth prediction algorithms rely on low-level features and often have problems at large textureless regions, while for semantic segmentation these regions are easier to capture. We propose a CNN-based architecture, where both tasks are tightly interconnected to each other. The model consists of an encoding stage which computes features for both tasks, semantic segmentation and disparity estimation. In the decoding stage we explicitly add the semantic predictions to the disparity decoding branch and we additionally allow to exchange information in the intermediate feature representations. Furthermore, we set the focus on efficiency, which we achieve by the usage of previously introduced ESP building blocks. We evaluate the model on the commonly used KITTI dataset.**

I. INTRODUCTION

Semantic segmentation and disparity estimation are active fields of research in computer vision. Most state-of-the-art work focuses on single-task models which perform either disparity estimation or semantic segmentation exclusively. In our work we create a multi-task model which performs both tasks simultaneously. This is mainly motivated by the following points: First, a single model performing multiple tasks jointly can be more efficient than using a separate model for each task. Therefore, multi-task learning is especially attractive for applications with limited resources and for real-time processing. Furthermore, it has been shown that the synergy among the individual tasks can boost their individual performances [22]. Except from these performance benefits information about both, the objects in the scene as well as their distance, respectively is important for many real world tasks. Such tasks could be for example autonomous driving or robot navigation.

In this work, we tackle both tasks, semantic segmentation *and* depth estimation, in a joint setting. We therefore start from two individually trained models and step by step fuse them to a joint model. The joint model allows to share computed features and therefore reduces the number of necessary operations yielding an efficient model. Furthermore, this setting allows to exchange information between the two tasks, which helps to increase the individual performances. We train the joint model end-to-end and evaluate the performance on the KITTI dataset.

*This work was supported by the research initiative Intelligent Vision Austria with funding from the AIT and the Austrian Federal Ministry of Science, Research and Economy HRSM programme (BGBl. II Nr. 292/2012)

[1]Institute of Computer Graphics and Vision, Graz University of Technology, Austria {robert.harb@student.tugraz.at, knoebelreiter@icg.tugraz.at}

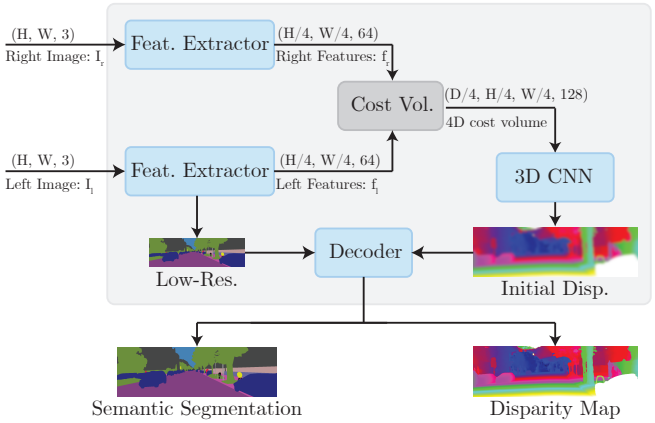


Fig. 1: **Overview** of our proposed architecture. Blue boxes denote learnable building-blocks of our model. The two feature extractor blocks have shared weights and are used for the semantic segmentation and for the disparity computation.

II. RELATED WORK

Semantic Segmentation: Given an input image, a semantic image segmentation algorithm assigns a semantically meaningful label for every pixel in the image. Labels can for example be *car*, *person* or *vegetation*. Applications of semantic segmentation exist in various areas including e.g., autonomous driving [6], [5], brain tumor segmentation [8], or segmentation of satellite images [21]. Recent work utilizes mainly Convolutional Neural Network (CNN) based approaches for semantic segmentation. In the following we divide them into two groups: The first group contains large and resource demanding models which try to improve top benchmark scores on common datasets like KITTI [6] or Cityscapes [5]. Top performing methods of this group are for example Deeplabv3+ [3] or the PSPNet [29]. Those models are able to produce high quality results at the cost of requiring a lot of parameters, memory and computational time. The prior mentioned method Deeplabv3+ uses $\approx$ 44M parameters and the PSPNet has $\approx$ 66M parameters. This makes them unsuitable for usage in memory restricted environments or for applications where real-time processing is needed.

The second group contains lightweight and therefore efficient models. They try to find optimal trade-offs between computational complexity and performance in terms of segmentation accuracy. Representatives of this group are e.g. MobileNet [10], ShuffleNetV2 [18], ENet [20] or ESPNet [19]. ESPNet is for example $\approx$ 22 times faster while being $\approx$ 180 times smaller than PSPNet. The semantic segmentation performance is still very good. ESPNet achieves a mean-Intersection-over Union (mIoU) score which is only $\approx$ 8% lower than the PSPNet score on Cityscapes benchmark. This

increase in computational efficiency is mainly achieved by incorporating the convolution factorization principle. Due to these benefits, we use parts of the ESPNet as basic building blocks in our model.

Disparity Estimation: Estimation of depth from image data has a wide variety of applications such as e.g. 3D reconstruction and augmented reality. In this work we focus on depth estimation from a rectified pair of stereo images. Similar as for semantic segmentation currently best performing models use CNNs. The first who showed how CNNs can be used for stereo matching have been Zbontar and LeCun [28]. Since then many CNN based stereo methods have been proposed such as PSMNet [2], Edge-Stereo [25] or CNN-CRF [14]. Even though CNN based architectures work well for stereo matching in many cases they still have problems in homogeneous regions, occlusions or reflections. One approach to mitigate this problems is to enlarge the receptive field by using e.g., multiscale features in a 3D CNN regression module [12] or dilated convolutions [17]. Another method is to incorporate semantic scene understanding, this has been done in non CNN based approaches by Ladicky et al. who construct separate random fields for disparity and semantic labeling and optimize them jointly [16]. And more recently by Displets which resolve ambiguities by incorporating object knowledge by modeling 3D vehicles [7], and SegStereo where semantic features are embedded in a cost volume for disparity prediction [27]. SegStereo is the most similar approach to our work; however, there are several fundamental differences. SegStereo uses a Resnet-50 for feature extraction, our architecture is based on ESPNet which is computationally more efficient. They perform a 1D-correlation on the cost volume, whereby we use a 3D-CNN for regularization. Additionally, they add an unsupervised photometric and smoothness component to their loss.

Multi-task learning: The general objective of multi-task learning is to improve generalization by leveraging the domain-specific information contained in the training signals of related tasks [1]. In neural network based architectures this can be done by sharing parts or the whole internal representation among different tasks. When considering efficiency, multi-task learning is attractive. Both memory and computational time can be reduced if we do not need a separate network for each task. This has been shown for example in UberNet [15] where as many as 7 tasks have been trained jointly including among others semantic segmentation and surface normals. Another representant is HyperFace [22] which performs face detection, landmark localization, pose estimation and gender recognition simultaneously while fusing intermediate layers of a deep CNN.

A common challenge in a multi-task setting is that every task has usually a separate domain specific loss function. One way to deal with multiple losses during training is to calculate a weighted linear combination of all losses. This total loss can then be optimized like in a single-task setting. However, due to e.g. different scalings in the task-specific losses weights are usually introduced. One possibility to determine optimal weights is to perform a grid search over

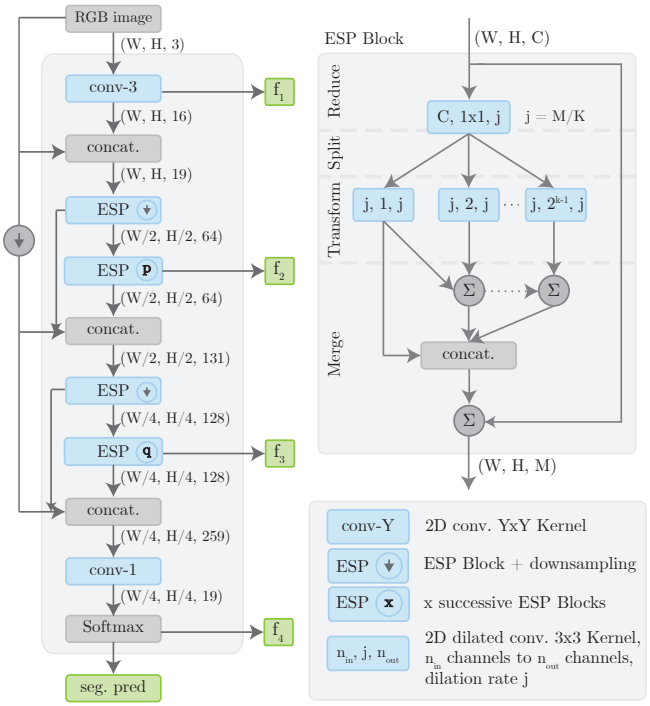


Fig. 2: ESP **feature extraction** module. Left: calculation of the feature maps f_1 to f_4 from an RGB input image. Top-right: Internal structure of one ESP block which maps from C input channels to M output channels. *Seg. pred.* and f_4 have the same content, but are drawn separately for clarity.

the weights. However the search space grows exponentially with the number of tasks and therefore one quickly runs into resource limitations. More sophisticated procedures for the weight calculation are proposed by [4], [11]. The former proposes GradNorm, where an adaptive weight is recalculated at each training step based on each tasks gradient magnitude. The latter introduced uncertainty weighting which considers the homoscedastic uncertainty of each task for its weighting. Another way to train CNN based multi-task architectures was introduced by Ozan et al. [24] who showed how to use gradient-based multiobjective optimization algorithms in such a setting.

III. Joint Segmentation and Disparity Network

Figure 1 gives a brief overview of our proposed architecture. At first the left and right input image I_l and I_r are passed through the feature extractor which calculates for both images the feature maps f_1 to f_4 and a low-resolution semantic segmentation prediction. Weights of both feature extraction modules are shared. Secondly by using pointwise convolutions and concatenation operations new features f_l and f_r are created from the last features f_1 to f_4. Those are aggregated to a 4D cost volume which is then regularized by a 3D CNN to obtain the disparity prediction. Finally the beforehand created low-resolution segmentation prediction and the disparity prediction are passed through a common decoder which refines the segmentation and disparity predictions.

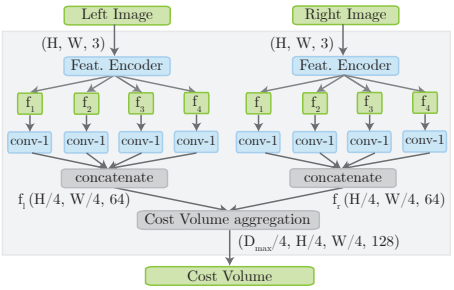


Fig. 3: Construction of the 4D **cost volume** using the left and right input image. conv-1 defines a 2D convolution with a 1x1 Kernel and 16 channels.

A. Semantic feature extraction

Figure 2 shows the structure of our semantic feature extraction module. The module simultaneously calculates a low resolution segmentation prediction, and the features f_1 to f_4 for disparity prediction. The features are extracted at different levels such that they include both high-level and low-level information. E.g. f_4 contains semantic information and f_1 contains edge information. We also use skip connections from the input image to several points in the network. This is also done in ESPNet [19] where it is referred to as input reinforcement. The main building blocks of our feature extraction module are ESP blocks [19]. The internal structure of an ESP block is shown at the top-right of Figure 2. ESP blocks are based upon the convolutional factorization principle which decomposes a standard convolution into a pointwise convolution and a spatial pyramid of dilated convolutions. This enables them to cover a wide effective receptive field while keeping the computational complexity low. A single ESP block performs the following four steps successively. At first a pointwise convolution is used to **reduce** the number of input channels from C to $j = \frac{M}{K}$, whereby K is a hyperparameter called width divider and M is the number of output channels. The resulting j channels are then **split** into K parallel branches. In the **transform** step each of the K branches applies a dilated convolution on its input, whereby for each branch the same kernel-size of $n \times n$, and a different dilation rate taken from 2^{k-1} with $k \in [1, K)$ is used. Finally, the outputs of all K branches are **merged** together. In the ESPNet paper [19] it has been shown that direct summation can introduce unwanted checkerboard or gridding artifacts. To overcome this problem hierarchical feature fusion (HFF) is used. The outputs of the different branches are first hierarchically added before they are concatenated (also shown in Figure 2).

B. Disparity Estimation

For disparity estimation we reuse the features f_1 to f_4 from the left and right input image. At first, all features are rescaled to a common resolution which is $(H/4, W/4)$ in our case. The features f_1 and f_2 have a higher resolution and thus they are downsampled. Then on each feature output $f_{\{1,2,3,4\}}$ a 1×1 convolution is applied to reduce the number of channels to 16. The resulting features are then concatenated into the feature f_l for the left image and f_r for the right image. These features contain information from multiple scales and are therefore well suited for matching. This is done by concatenating f_l together with the corresponding right feature map f_r over all disparity levels. This results in a 4D cost volume of dimensionality $(D_{max}/4, H/4, W/4, 128)$. See Figure 3 for a visualization. Note that the first dimension of the cost volume equals $D_{max}/4$ and not D_{max} since we downsample over all 3 spatial dimensions. To regularize the cost volume we fed it into a 3D CNN with a similar architecture as in GC-Net [12] and PSMNet [2]. The network consists of five consecutive residual blocks containing two 3D convolutional layers with kernels of size $3 \times 3 \times 3$ and 32 channels, followed by a single $3 \times 3 \times 3$ 3D convolutional layer with one output channel. We upsample the output of the 3D CNN from $(D_{max}/4, H/4, W/4)$ to (D_{max}, H, W). This volume contains an estimated cost for each disparity value in the first dimension. A dense disparity prediction could be acquired by e.g. the argmin operation over the first dimension. However a regular argmin operation is non-differentiable and therefore does not allow training with sub-pixel accuracy. Following previous work like GC-Net [12] we use a differentiable SoftArgmin function instead. To calculate the SoftArgmin function we first take the negative values of the cost volume C to convert the cost volume to a probability volume. Then we normalize the probability volume with a softmax across the disparity dimension. Finally, we calculate the sum over all disparity values weighted by their normalized probability. This leads to the following calculation of the SoftArgmin $\hat{d}_i$ for every pixel i:

$$\hat{d}_i = \sum_{d}^{D_{max}} d \cdot \frac{\exp\left(-C_i(d)\right)}{\sum_{d'} \exp\left(-C_i\left(d'\right)\right)}, \tag{1}$$

whereby D_{max} is the maximum disparity respectively the size of the first dimension of the cost volume C. The SoftArgmin is differentiable and thus we can learn sub-pixel accurate disparity maps.

C. Decoder

In order to allow the two tasks to benefit from each other, we propose a decoder which jointly creates the final prediction for both the semantic segmentation and the disparity map. The structure of the decoder is shown in Figure 4. In the decoder the low-resolution disparity and segmentation predictions are upsampled in two separate paths, whereby the two paths are connected at several positions to allow information to flow between the disparity and segmentation path. The inputs of the decoder are a disparity map and a segmentation prediction of size $\left(\frac{H}{4} \times \frac{W}{4}\right)$. Those are then upsampled by the decoder in three consecutive stages, whereby each stage is processing data at the increasing resolutions, i.e. $\left(\frac{H}{4} \times \frac{W}{4}\right) \rightarrow \left(\frac{H}{2} \times \frac{W}{2}\right) \rightarrow (H \times W)$. The input of each stage is either a disparity or segmentation prediction at the respective resolution of the stage, and the output is a refined version of the input. This is done with a residual connection [9] which allows to add finer grained information at the next stage without the need of completely reconstructing the result.

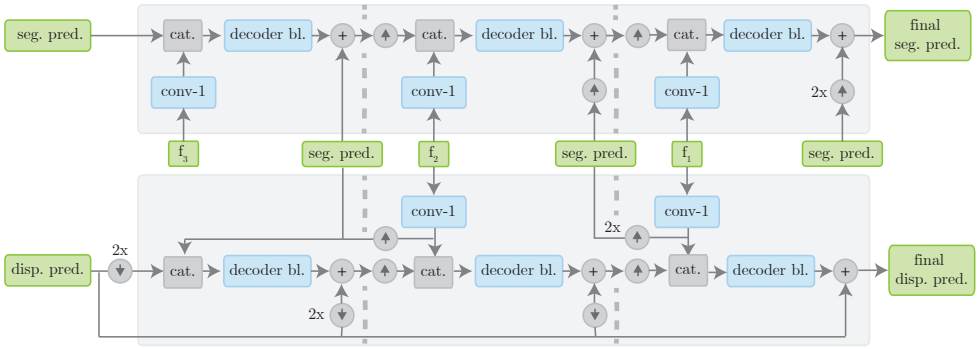


Fig. 4: The **decoder** upsamples the low-resolution segmentation prediction and refines the disparity prediction. Stages of the decoder are separated by vertical dashed lines.

Let us now look into one stage (=one processing resolution) in more detail. To provide fine details which have been lost during the former downsampling and processing we concatenate the encoded feature maps at the respective resolutions. This is similar to the skip connections between the contracting and expanding path in U-Net [23]. We squeeze the concatenated features with a 1×1 convolution to match the number of semantic classes. In the disparity path we additionally concatenate the segmentation prediction.

The result is then processed by a decoder block. The input of a decoder block is first passed through two consecutive ESP blocks with 48 output channels and then through another 1×1 convolution to reduce the output to the 19 channels for the segmentation path and to 1 channel for the disparity path, respectively. Finally, an elementwise addition between the output of the decoder block and the respective prediction of each path is calculated to get the output of the stage.

D. Multi-task loss

To train our model we use a cross-entropy loss for the segmentation task, and a smooth $L1$ loss for the disparity task. We calculate our disparity loss as follows:

$$L_d(d,\hat{d}) = \frac{1}{N_d}\sum_{i=1}^{N_d} \text{smooth}_{L_1}\left(d_i - \hat{d}_i\right), \tag{2}$$

whereby we mask out pixels with invalid disparity label and the smooth L1 loss is given as:

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases}. \tag{3}$$

In the KITTI training set the amount of pixel occurrences of each class varies substantially. To account for this imbalance we weight each class by a different weight w_{class} in the segmentation loss. The weights are calculated as follows:

$$w_{\text{class}} = \frac{1}{\ln(c + p_{\text{class}})}, \tag{4}$$

whereby p_{class} denotes the probability of class occurrence at one pixel in the training set and c is a hyperparameter. We calculate the total loss L as a weighted sum of L_s and L_d :

$$L(\theta_j, \theta_s, \theta_d) = w_s L_s(\theta_j, \theta_s) + w_d L_d(\theta_j, \theta_d), \tag{5}$$

where $\Theta \in \mathbb{R}^N$ contains all N parameters of our model, $\theta_s \subset \Theta$ contains the parameters only used for the segmentation task, $\theta_d \subset \Theta$ the parameters only used for the disparity task and $\theta_j = \theta_s \cup \theta_d$ are the jointly used parameters. The weights for the segmentation and disparity loss are denoted by w_s and w_d. We evaluate how the performance of our model changes if we set the weights w_{disp} and w_{seg} to several fixed values, and if the weights are calculated using adaptive loss balancing with GradNorm [4].

IV. Experiments

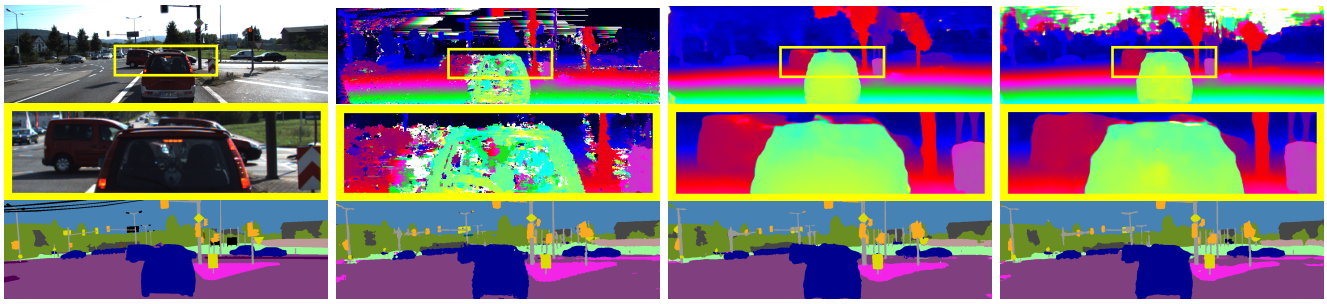
A. Data

We use the KITTI dataset [6] for training and evaluation of our model. It contains ground truth data for both semantic segmentation and disparity. The dataset consists of 200 training and 200 test images of driving scenes, whereby the labels of the 200 test images are not available to the public but hold back. This provides a reliable performance benchmark of different models on a public online leaderboard. To evaluate our model we use a 80/20 random split of the 200 training images. Following most recent work which evaluates segmentation on KITTI [27] we use the most common 19 classes for training and evaluation. To account for the class imbalance in the KITTI we set the hyperparameter c in the calculation of w_{class} to 1.02, as in ENet [20]. This restricts the class weights to be in the interval of $[1, 50]$.

B. Training

All convolutions are followed by a batch normalization layer except the last layer of the 3D regularization module, leaky ReLU units ($\alpha = 0.01$) are used as activation functions, except the 3D regularization module where ReLU units are used for activation. In the ESP feature extraction module the number of successive blocks is set to $p = 5$ and $q = 3$. The width divider K of all ESP blocks is set to 5. We use the Adam [13] optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with an initial learning rate of 0.0005 and decay it by a factor of 0.5 at epoch 400 and 500. The complete training is performed end-to-end and no pre-training is done. Following PSMNet [2], we set the maximum disparity to 192. For all experiments a batch size of 2 was used. All input images are normalized, using the mean and variance of the training set.

As input we use the full resolution KITTI images. However to avoid memory problems during training, we perform a 114×128 random crop on the cost volume. The memory



(e) Input/GT (f) Baseline (g) Model A (h) Model B

Fig. 5: Qualitative comparison. (a) shows the input image and the semantic ground-truth. (b-d) show the predictions of our models. Top row: Color-coded disparity maps with green=near and blue=far. Middle row: Close ups of the selected regions. Bottom row: Semantic segmentation. Parameters for *Model A* and *Model B* have been obtained from the same training session but at different epochs.

requirements of our 3D CNN are relatively high, and all recent networks with similar architectures like PSMNet [2] use cropped input images for training. With our approach the feature extraction module can exploit information from the complete input image, while we still save a significant amount of memory in the 3D CNN. It has also been shown by [26] that information from the complete input image is benefical for stereo matching. Note during inference no cropping in the 3D CNN is necessary.

C. Baseline systems

We compare our final results also to simple baseline systems. For disparity estimation we implemented a network similar to the CNN-7 model used in CNN-CRF [14] without the CRF. The network consists of a 7-layer unary CNN with 100 filters in each layer, whereby the filter size in the first layer is (3×3) and in all following layers (2×2), as activation function the *tanh* is used. The unary CNN is then used to extract features from the left and right input image which are passed through a cross-correlation function which outputs the softmax normalized scalar products of corresponding feature vectors. For baseline segmentation we use an 8 layer CNN network, whereby the first 7 layer have 64 filters and a kernel size of (3×3) followed by BatchNormalization and a ReLU activation function. We use a dilation rate of 2 for the layers 3 and 4, and a dilation rate of 4 for the layers 5, 6, 7, and 8. After the second and fourth layer we apply a pooling operation with a factor of 2. The last layer has a kernel size of (1×1) and 19 output filters.

D. Results

During training we monitor the mean Intersection-over-Union (mIoU) and the bad3 pixel error, i.e. the percentage of pixels with disparity error > 3.0 pixels. When only disparity or segmentation is trained, we report the respective score at the best performing epoch. In the multi-task setting, the epoch where one task reaches its best performance is not necessarily the same epoch where the other task performs best. Therefore we need to choose an epoch which gives a good trade-off between performance of both tasks. To give a better insight in our model, we will examine two models both obtained from the same training session ($w_{disp} = 5$ and $w_{seg} = 1$), but parameters are taken at different epochs. One model was chosen with slightly better performance in segmentation and the other in disparity estimation. We will simply refer to them as *Model A* and *Model B*. Figure 5 shows a qualitative comparison of our results. In the top-region *Model B* gives a relative noisy disparity prediction. The segmentation close-ups show that *Model B* is able to predict the car contours better than *Model A*.

Model	NOC	All	Seg.	Params	Time
	bad3	bad3	mIoU	#	sec.
ours - ESP					
Model A	3.16	3.75	47.04	0.64M	0.28
Model B	3.84	4.37	49.43	0.64M	0.28
GradNorm $\alpha = 0.5$	4.07	4.54	46.6	0.64M	0.28
GradNorm $\alpha = 1.5$	3.95	4.68	48.9	0.64M	0.28
Only disp	3.69	4.24	–	0.57M	0.28
Only seg	–	–	50.48	0.21M	0.08
ours - baseline					
Only disp	14.10	15.6	–	0.28M	–
Only seg	–	–	44.13	0.23M	–
other					
PSMNet [2]	2.14	2.32	–	5.23M	0.44
SegStereo [27]	–	2.25	59.10	25.60M[1]	0.60
CNN-CRF [14]	4.84	5.50	–	0.28M	0.76

TABLE I: Results. Note that we report performance on our validation set, and the published results report performance on the retained test set of KITTI.

Table I shows the results of our experiments. Disparity estimation is evaluated on non-occluded (NOC) pixels, i.e. pixels visible in both images, and on all pixels. We have been able to significantly improve upon our baseline systems by incorporating ESP blocks, which shows that they are an efficient building-block for light-weight segmentation and stereo matching networks. When doing multi-task training, performance in segmentation gets slightly worse, but for disparity estimation performance increases upon the single-task setting. Looking at the number of parameters of each model we note that i) all of our models are light-weight compared to most other recent work and ii) only a relative increase in

[1]SegStereo does not report an exact number of parameters for their model, therefore we report the parameters of ResNet-50 which they use as a backbone.

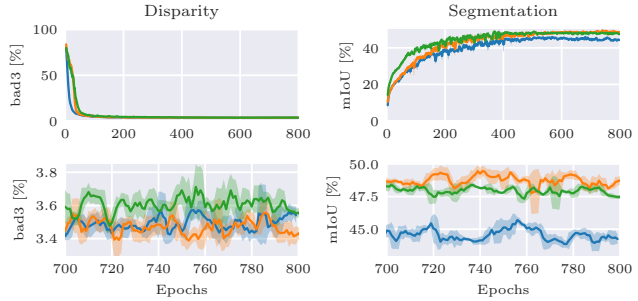


Fig. 6: Training curves, Shaded regions show standard deviation. Left: lower is better. Right: higher is better. Blue: $(w_{disp} = 10, w_{seg} = 1)$, green: $(w_{disp} = 10, w_{seg} = 1)$, orange: $(w_{disp} = 5, w_{seg} = 1)$.

parameters of 12% is needed from our model which only performs disparity estimation to the model which calculates both disparity and segmentation. Overall SegStereo and PSMNet still perform better, we mainly attribute this to the fact that they use significantly more complex models without a strong focus on efficiency. Compared to CNN-CRF we are able to achieve slightly better performance, however it should be noted that their model uses even less parameters than ours.

We also investigated how different weightings of the disparity and segmentation loss influence training. Figure 6 shows training curves for other weightings, one can see that especially the performance in segmentation degrades if its relative weighting is too small. The usage of GradNorm did not help to increase performance. The best results we achieved were obtained by using a fixed weighting of $w_{disp} = 5$ and $w_{seg} = 1$.

V. Conclusion

We have proposed a model which jointly performs semantic segmentation and stereo matching. Compared to performing both tasks separately, our shared architecture reduces runtime and model complexity, while achieving the same performance. Furthermore we carefully designed our model with a focus on efficiency and are able to reduce runtime compared to similar state-of-the art methods.

References

[1] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, 1997.

[2] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[3] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[4] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 2018.

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, 2013.

[7] F. Guney and A. Geiger, "Displets: Resolving stereo ambiguities using object knowledge," in *CVPR*, 2015.

[8] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, and H. Larochelle, "Brain tumor segmentation with deep neural networks," *Medical image analysis*, vol. 35, 2017.

[9] K. He, X. Zhang, and S. Ren, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[11] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[12] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[14] P. Knöbelreiter, C. Reinbacher, A. Shekhovtsov, and T. Pock, "End-to-end training of hybrid cnn-crf models for stereo," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[16] L. Ladickỳ, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, and P. H. Torr, "Joint optimization for object class segmentation and dense stereo reconstruction," *International Journal of Computer Vision*, vol. 100, no. 2, 2012.

[17] Z. Li and L. Yu, "Compare stereo patches using atrous convolutional neural networks," in *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, ser. ICMR '18. ACM, 2018.

[18] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[19] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[20] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[21] M. Pesaresi and J. A. Benediktsson, "A new approach for the morphological segmentation of high-resolution satellite imagery," *IEEE transactions on Geoscience and Remote Sensing*, vol. 39, no. 2, 2001.

[22] R. Ranjan, V. M. Patel, and R. Chellappa, "Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, 2019.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015.

[24] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *NIPS*, 2018.

[25] X. Song, X. Zhao, H. Hu, and L. Fang, "Edgestereo: A context integrated residual pyramid network for stereo matching," *arXiv preprint arXiv:1803.05196*, 2018.

[26] S. Tulyakov, A. Ivanov, and F. Fleuret, "Practical deep stereo (pds): Toward applications-friendly deep stereo matching," in *Advances in Neural Information Processing Systems*, 2018.

[27] G. Yang, H. Zhao, J. Shi, Z. Deng, and J. Jia, "Segstereo: Exploiting semantic information for disparity estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[28] J. Zbontar, Y. LeCun, *et al.*, "Stereo matching by training a convolutional neural network to compare image patches." *Journal of Machine Learning Research*, vol. 17, no. 1-32, 2016.

[29] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *CVPR 2017*, vol. 2017-January, 2017.