

MultiDepth: Single-Image Depth Estimation via Multi-Task Regression and Classification

Lukas Liebel, Marco Körner

Computer Vision Research Group, Chair of Remote Sensing Technology
Technical University of Munich, Germany
lukas.liebel@tum.de

We introduce MultiDepth, a novel training strategy and convolutional neural network (CNN) architecture that allows approaching single-image depth estimation (SIDE) as a multi-task problem. SIDE is an important part of road scene understanding. It, thus, plays a vital role in advanced driver assistance systems and autonomous vehicles. Best results for the SIDE task so far have been achieved using deep CNNs. However, optimization of regression problems, such as estimating depth, is still a challenging task. For the related tasks of image classification and semantic segmentation, numerous CNN-based methods with robust training behavior have been proposed. Hence, in order to overcome the notorious instability and slow convergence of depth value regression during training, MultiDepth makes use of depth interval classification as an auxiliary task. The auxiliary task can be disabled at test-time to predict continuous depth values using the main regression branch more efficiently. We applied MultiDepth to road scenes and present results on the KITTI depth prediction dataset. In experiments, we were able to show that end-to-end multi-task learning with both, regression and classification, is able to considerably improve training and yield more accurate results.

1 Introduction

Depth estimation is an important part of scene understanding in various domains. Traditionally, depth maps are derived from active sensor measurements, such as *light detection and ranging (LiDAR)* point clouds, or from stereo images. However, in the absence of observations allowing for the explicit reconstruction of pixel-wise depth values for a corresponding image, methods for directly estimating depth from a single monocular image are required. A typical application lies in robotics, most prominently *autonomous vehicles*, where a high degree of redundancy is of vital importance. Figure 1 exemplarily shows the result of *single-image depth estimation (SIDE)* for a road scene.

Predicting depth from a single image has seen substantial improvements due to the rise of *deep learning*-based methods. First approaches to SIDE for indoor scenes using deep *convolutional neural networks (CNNs)* were presented by Eigen et al. [11, 10]. Ever since then, various methods for the prediction [28, 35, 30, 34, 21, 53] and evaluation [25] of depth maps for indoor scenes have been proposed.

SIDE in unstructured outdoor environments poses an even greater challenge. Annotated training data is hard to obtain, as RGB-D cameras are not able to provide data at distances of more than 10 m and their low resolution is not able to capture scenes crowded with differently sized objects. Available datasets [14, 9, 22] use measurements from LiDAR sensors to accumulate depth maps as ground-truth, which are, however, naturally sparse. They can serve as an additional input for

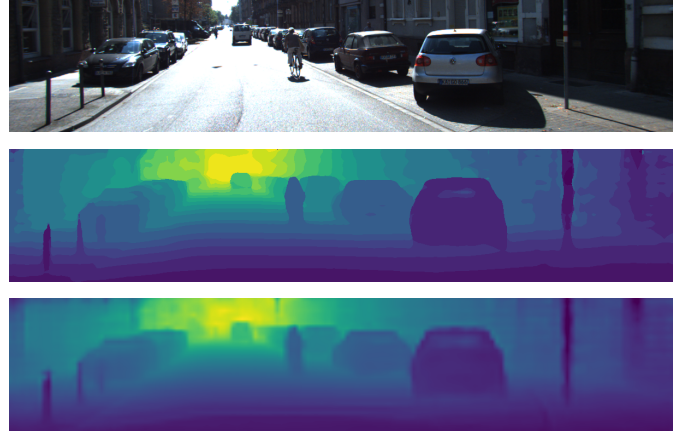


Figure 1: Depth prediction (bottom) for a single RGB image (top) from the KITTI depth prediction benchmark [45, 14]. Auxiliary depth interval classification result (middle) used during training to support the depth value regression via optimization of a multi-task objective.

depth completion [7] or as labels for actual depth prediction. Utilizing stereo image pairs and a photo-consistency loss for semi-supervised training to estimate disparities is another option used in recent approaches [15, 47, 27]. The application of SIDE in *advanced driver assistance systems (ADAS)* and autonomous vehicles requires high precision and robustness. Approaches to this challenging task have been proposed [13, 23, 43, 12, 18, 31, 26, 29, 50], but still require improvement for the application in *autonomous driving* [43].

Estimating continuous depth values is a typical regression task.

However, by discretizing depth space into intervals, it can be cast as a classification problem [1, 19, 12]. While this is less intuitive, classification methods have been found to converge faster and more reliably. This was shown by Fu et al. [12], who advanced this approach by taking into account the ordinal characteristic of depth intervals and achieved top-ranking results in the *KITTI depth prediction* benchmark [45, 14]. Combining the properties of both tasks, *i.e.*, depth regression and classification of depth intervals, in order to exploit their individual advantages yields a *multi-task* problem.

Multi-task learning [3, 2] enables training of CNNs that produce multiple outputs in a single round of inference. In his review article, Ruder [40] gives an extensive overview of CNN-based multi-task learning. Driven by advances in methodology [41, 23, 36, 17, 52], multi-task learning has become increasingly popular in computer vision. It has successfully been applied to numerous applications. In *road scene understanding*, problems that have been tackled using multi-task learning include object detection and bounding box regression [44, 6, 4], as well as SIDE in combination with surface normal estimation or semantic segmentation [37, 49, 10, 38, 47, 23].

A different approach to employing multi-task learning is the utilization of auxiliary tasks [33, 8, 46] that merely serve as additional supervision to the network during training and are discarded during test-time. This approach can be seen as an extension to comprehensive regularization terms in loss functions as used by Li et al. [32]. It could be shown that by adding auxiliary tasks to a network the performance of the main task increases [33, 8].

Considering this prior work, we approach SIDE by posing it as a multi-task problem with a main regression task and an auxiliary classification task. As both tasks use depth measurements as ground-truth, with minor pre-processing applied in order to segment the continuous depth space into intervals for classification, the auxiliary supervisory signal does not require additional annotations. By adding the auxiliary classification task as a regularizer, we expect training to converge faster and yield better results. Closely related to the idea of casting SIDE to a classification task is the *deep ordinal regression network (DORN)*, proposed by Fu et al. [12]. They do, however, use ordinal regression instead of classification and, furthermore, do not treat it as an additional task. Kendall et al. [23] propose uncertainty-based weighting of individual tasks, which we build upon, but do not make use of auxiliary tasks. Auxiliary tasks have been utilized before [33, 8, 46], however not with posing the same task in two different ways. In contrast to Gurram et al. [19], who use depth interval classification as pre-training, we train for regression and classification in an end-to-end manner.

The main contribution of this paper is the proposal of *Multi-Depth*, a novel multi-task approach to SIDE which incorporates both regression and classification. This training strategy facilitates fast and robust convergence. We, furthermore, provide an implementation of the proposed approach based on *PSPNet* [51] and uncertainty-based weighting [23] that we used to show the superiority of training with an auxiliary task as compared to

basic regression.

2 Depth Prediction Using a Multi-Task Regression and Classification Loss

Predicting depth maps from single images using a CNN requires learning to estimate continuous depth values for each pixel. This makes SIDE a prime example for regression tasks. A common approach to this problem is, thus, optimizing a CNN by the means of *mean squared error (MSE)* or one of its variants. However, it has been found that the convergence is slow and tends to yield suboptimal results [12].

In contrast to this, SIDE can be posed as a classification problem using quantized depth values [1, 19, 12]. This allows employing state-of-the-art semantic segmentation methods. Semantic segmentation, as another core task of many computer vision applications, such as autonomous driving, has been successfully tackled in recent years. State-of-the-art CNN architectures for semantic segmentation [51] most commonly utilize SoftMax cross entropy loss which is renowned for its stable convergence towards well-performing local minima. Exploiting the advantageous properties of semantic segmentation methods for SIDE, however, comes at the price of inevitable quantization errors.

Combining the advantages of both, regression and classification, is a promising approach to the accurate estimation of depth values with stable convergence. In order to achieve this goal, we propose MultiDepth, a multi-task training procedure for jointly learning two representations of a depth map, exemplarily shown in Figure 1. As network architectures for SIDE often use encoder-decoder structures, a second branch for the classification of depth intervals can easily be added as an auxiliary decoder [33, 8]. The auxiliary decoder is only active during training and can alternatively be disabled during test-time to make inference more time and memory efficient, or produce an additional redundant depth map that can be used to enhance or verify the regression result. By sharing a major part of the network weights, a common and robust representation for both tasks is learned during training. Especially the encoder part, which is expected to learn a rich representation of the input data, should be shared in order to encourage this behavior. The decoders, on the other hand, only contain few layers to extract and reshape task-specific information from the shared encoded features. State-of-the-art network architectures often use very deep classification networks as a backbone for deep feature extraction in the encoder, which, therefore, usually accounts for a vast majority of the parameters.

2.1 Efficient Optimization of Depth in Log-Space

In the context of autonomous driving, even small errors in distance estimates for objects close to the camera—and hence the vehicle—may prove fatal. Distances for faraway objects, on the other hand, only need to be estimated with lower accuracy. This requirement can be considered by estimation in non-linear space. Modern methods for CNN-based SIDE use sophisticated loss functions in order to incorporate such desired properties. One example of an objective function that considers a non-linear

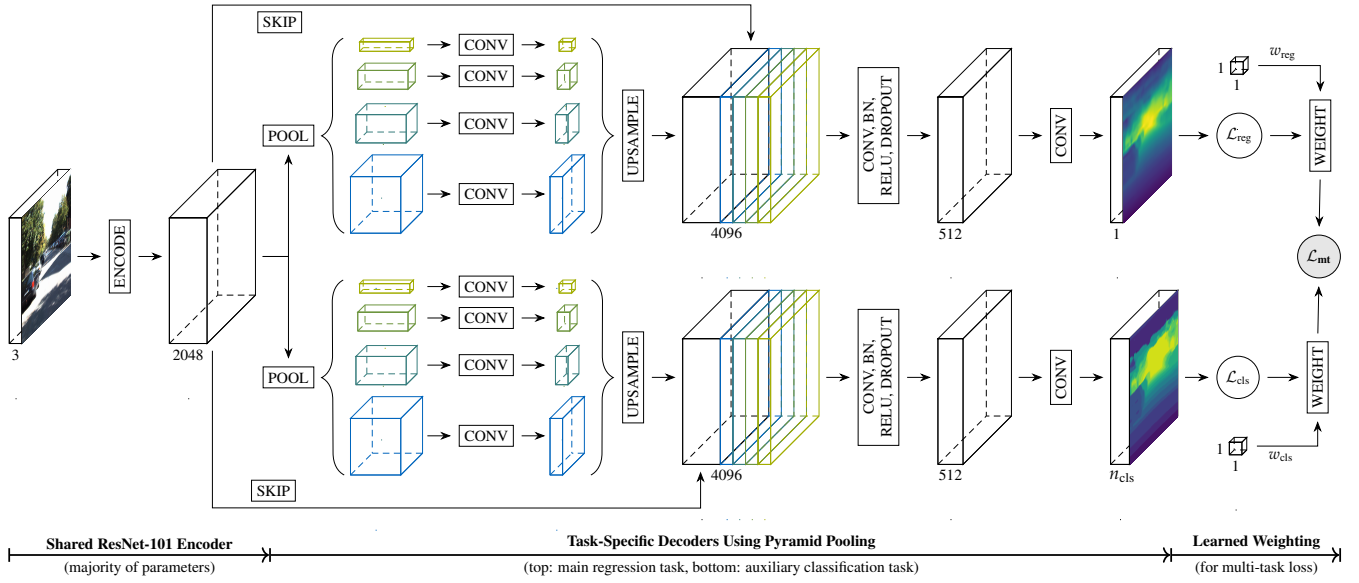


Figure 2: Network architecture for the proposed MultiDepth approach featuring a shared ResNet-101 encoder with dilated convolutions and task-specific decoders including pyramid pooling for the main regression and the auxiliary classification task. The multi-task loss is constructed from the contributing single-task losses utilizing learned weighting terms, which represent task uncertainties.

weighting of errors in log-space [11] is commonly used as a metric for scoring depth estimation results. We use the *scale invariant logarithmic error (SILog)*

$$\text{SILog}(D, D^*) = \frac{1}{n} \sum_{i,j} (\log d_{i,j} - \log d_{i,j}^*)^2 - \frac{1}{n^2} \left(\sum_{i,j} \log d_{i,j} - \log d_{i,j}^* \right)^2 \quad (1)$$

for comparing predicted depth maps $D^* = (d_{i,j}^*)$ with n depth values to the ground-truth D . Utilizing extensive loss functions for training often yields superior results over more basic ones, such as the MSE. The latter, however, are less computationally expensive thus speeding up the training process. While they might not directly optimize the final objective, they act as a proxy of reasonable quality [16].

In order to overcome the limitations of both approaches, we relocate computationally expensive operations to the data pre-processing stage. Thanks to parallel pre-loading of batches, this allows us to optimize our objective more directly while still maintaining the speed and efficiency of traditional loss functions. Hence, we transform depth values d in our training data to normalized log-space during data loading. After globally defining the upper and lower boundaries d_{\min} and d_{\max} , we calculate the transformed depth as

$$d_{\log} = \frac{\log_e (d - d_{\min} + 1)}{\log_e (d_{\max} - d_{\min} + 1)} \quad (2)$$

2.2 Regression of Depth Values

In order to extract the desired information from the encoder features, task-specific decoders can be added to the network.

For the main task, *i.e.*, regression of continuous depth values, the decoder directly predicts values in normalized log-space. Few network layers to condense and reshape information along all dimensions of the feature map are expected to suffice here, given the rich representation of the input data provided by the encoder.

2.3 Classification of Depth Intervals

The auxiliary task, *i.e.*, classification of discrete depth ranges, requires corresponding target labels. In order to partition the continuous depth space, we set a number of intervals n_{cls} . Choosing a relatively low number of classes ensures stable convergence while still preserving enough detail to support the regression part of the network with basic depth perception. While Fu et al. [12] report 80 intervals to be the optimum for their ordinal regression method, this is not directly applicable to our approach in which the classification task merely supports the regression branch of the network. Hence, trading high quantization errors for stable convergence and robust performance is desirable in our case. Following our general concept of estimating non-linearly scaled depth values and existing approaches [12], we set our intervals to be uniformly distributed in normalized log-space.

Apart from different target labels and a necessary change in the output feature dimension from one to n_{cls} , the decoder for the classification task closely resembles the structure of the regression branch. Keeping the task-specific branches shallow, and thus the ratio of shared vs. individual parameters high encourages the learning of high-level features that are suitable for both tasks.

2.4 Multi-Task Optimization

Jointly training both tasks requires combining the individual losses $\mathcal{L}_{\text{task}}$ to a single multi-task objective \mathcal{L}_{mt} , subject to optimization. A naïve approach to this is simply summing up $\mathcal{L}_{\text{mt}} = \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{cls}}$. The range and variance of values produced by \mathcal{L}_{reg} and \mathcal{L}_{cls} may differ significantly, especially since they belong to different families of loss functions. Hence, weighting with w_{reg} and w_{cls} , such that $\mathcal{L}_{\text{mt}} = w_{\text{reg}} \cdot \mathcal{L}_{\text{reg}} + w_{\text{cls}} \cdot \mathcal{L}_{\text{cls}}$ allows for adjusting the contribution of each loss. One possible way of finding appropriate values for the weights is treating them as hyperparameters to be manually tuned. This is a tedious process that results in a set of constant weights that might be suitable in general, but fail to adapt to changes during training. Therefore, automatically optimizing a set of dynamic parameters is a promising approach.

Existing methods propose to weight tasks with respect to their homoscedastic uncertainty [23] or difficulty [17]. We follow the former and introduce weighting parameters $w_{\text{reg}} = 0.5 \cdot \exp(-s_{\text{reg}}^2)$ and $w_{\text{cls}} = \exp(-s_{\text{cls}}^2)$ with $s_{\text{task}} = \log_e(\sigma_{\text{task}}^2)$. While σ_{task} represent the actual task uncertainties, we optimize for s_{task} , due to numerical stability, as advised by the original authors [23]. Since minimizing $\mathcal{L}_{\text{mt}} = w_{\text{reg}} \cdot \mathcal{L}_{\text{reg}} + w_{\text{cls}} \cdot \mathcal{L}_{\text{cls}}$ favors the trivial solution $w_{\text{reg}} = w_{\text{cls}} = 0$, they furthermore propose to add regularization terms to the weighted single-task losses. Adding such terms $r_{\text{task}} = 0.5 \cdot s_{\text{task}}$ yields our final multi-task loss

$$\mathcal{L}_{\text{mt}} = w_{\text{reg}} \cdot \mathcal{L}_{\text{reg}} + r_{\text{reg}} + w_{\text{cls}} \cdot \mathcal{L}_{\text{cls}} + r_{\text{cls}} \quad . \quad (3)$$

3 Experiments

To evaluate our approach, we implemented a deep CNN and trained it on a large road scene understanding dataset.

3.1 Network Implementation

As networks for SIDE and semantic segmentation share typical properties, such as their feed-forward auto-encoder structure, architectures are often shared across both tasks [28, 30]. Furthermore, our auxiliary pixel-wise depth interval classification task, in fact, is a segmentation task. Therefore, we deem well-performing deep CNN architectures for semantic segmentation suitable as a basis for our implementation.

With its proposed *pyramid pooling* module, the PSPNet [51] is able to accumulate information from spatial context. The very deep ResNet-101 [20] used as a backbone for the feature encoder, on the other hand, provides the necessary capacity to learn a rich and robust representation of the input data. The network also features a natural encoder-decoder structure using pyramid pooling and dilated convolutions [5, 48] to preserve high-resolution feature maps up to the output. As a result, the PSPNet achieves state-of-the-art results on various challenging semantic segmentation datasets [51].

Since these properties match with our requirements, we re-implemented their network architecture in PyTorch and adapted it to our multi-task approach. An overview of our final Multi-Depth network architecture is shown in Figure 2. The auxiliary

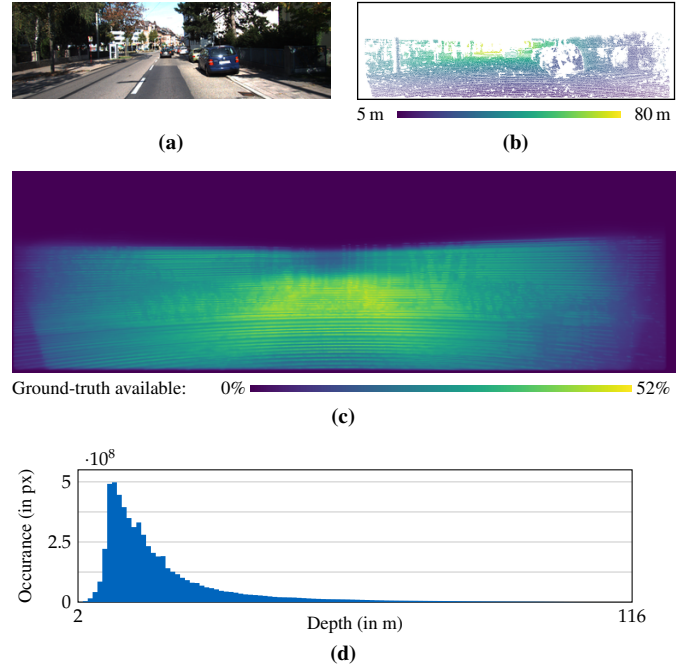


Figure 3: Sample from the KITTI depth prediction dataset [45, 14] with RGB image (a) and sparse ground-truth depth (b). Distribution of (c) available ground-truth depth values per pixel in the training set with an apparent lack of measurements in the upper part of the images, and (d) of valid depth values in the training set.

loss used in the original implementation of Zhao et al. [51] was not implemented.

Parameters between both tasks are shared through the ResNet encoder, which accounts for approximately 50% of the total number of parameters in the network. Sharing a majority of parameters between both tasks enforces the learning of a common representation in the 2048-dimensional feature map. Since both tasks represent a perception of depth, we expect these features to already contain high-level information that strongly hints towards the final outputs. This representation needs to be condensed for each task. The weights of the pyramid pooling modules, the following blocks containing a single convolution (along with batch normalization, ReLU activation, and dropout), and the final convolutional layers for reshaping the intermediate representation to the required output format are task-specific. We use skip connections as in the original PSPNet to retain spatial structure.

3.2 Training and Validation Data

Predicting depth with high robustness, redundancy, and accuracy is especially important for autonomous driving. As Multi-Depth is designed to fulfill these conditions, we chose to train and evaluate our method on images of road scenes. Amongst the most popular and comprehensive datasets for road scene understanding is the *KITTI* dataset [14], which provides ground-truth for several highly relevant tasks. The KITTI depth prediction dataset [45, 14] provides sparse point clouds, acquired using LiDAR sensors. The dataset contains roughly 86 000 training samples. Ground-truth depth values, as exemplarily shown in Figure 3b, are available for roughly 12% of the pixels in each

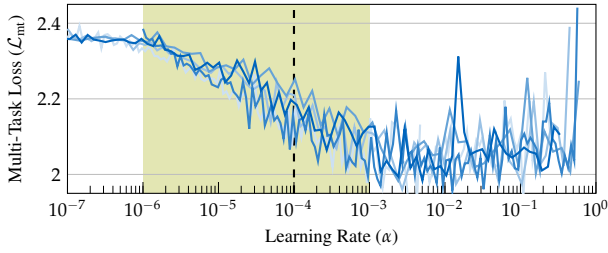


Figure 4: Experimental estimation of a suitable learning rate. The shaded interval of learning rates decreases the loss during training, the dashed line marks the selected learning rate.

image on average. For some samples, however, this number drops to only 0.8%. Figure 3c shows the unequal spatial distribution of depth measurements with a significant lack of data for the upper part of the images.

We train on the full set of training images and validate on the 1000 images of the provided “val_selection_cropped” set. Even though the evaluation of our approach is, naturally, largely based on relative comparisons and ablation studies, we still want to evaluate MultiDepth on a challenging dataset with respect to the state of the art. Hence, we score our final results on the official KITTI depth prediction benchmark [45].

3.3 Data Augmentation and Scaling

To produce appropriately sized input images for our network, we implemented random cropping with a resolution of 128×128 px to 256×256 px, depending on the respective experiment. While both of the used patch sizes are relatively large, using smaller patches leads to an increasing number of images with a distinct lack of ground-truth information (*cf.* Figure 3c). We refrained from scaling images as a data augmentation measure. This allows us to exploit the fully-convolutional nature of our network during inference since the size of objects in the cropped patches does not differ from the original full-sized images. Counterintuitively, horizontal flipping appears to improve results even for road scenes, where differences in the left and right side of the images are systematic and meaningful [39].

Depth values were scaled according to Equation (2) with $d_{\min} = 2$ m and $d_{\max} = 125$ m, following the distribution of depth values in the training dataset (*cf.* Figure 3d). While the lower bound was tightly set, in order not to waste accuracy in the critical lower part of the normalized log-space, the upper bound is more generous, to allow for higher depth values which account for very little accuracy in the normalized log-space. Class-labels for the classification branch of the network were derived by binning the scaled depth values to intervals in $[d_{\min}, d_{\max}]$, equally spaced in normalized log-space. By setting the lower and upper clipping planes d_{\min} and d_{\max} for the quantization to $d_{\min} > d_{\min}$ and $d_{\max} < d_{\max}$, we simplify the auxiliary task. As the classification task is only meant to support training, this helps to keep the auxiliary task sufficiently easy to train.

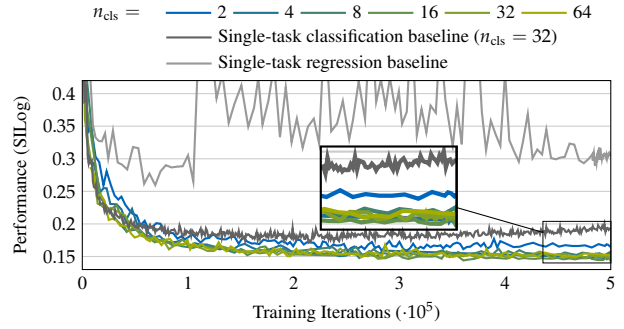


Figure 5: Validation results for single-task regression and classification vs. multi-task training with different n_{cls} for the auxiliary classification task. The regression baseline diverges quickly at first and converges to an undesirable local minimum afterward. As expected, the classification baseline converges fast and stable. Multi-task training with $n_{\text{cls}} \geq 2$ considerably improves performance over both baselines. Best results achieved for $n_{\text{cls}} \geq 4$.

3.4 Training Procedure

Since ground-truth for both of our tasks is sparse, we implemented a sparse MSE loss for the regression task and a sparse SoftMax cross entropy loss for the classification task. The multi-task loss, which is subject to optimization, was constructed as presented in Equation (3). The weighting parameters for the multi-task loss were initialized as $s_{\text{reg}} = s_{\text{cls}} = 1$ and optimized with the network parameters. Kendall et al. [23] point out the robustness of the weighting parameters with regard to their initialization, which our experiments confirmed.

Settings for batch and patch sizes are coupled and bound by each other in terms of GPU memory consumption. Due to the sparsity of the ground-truth data, larger patches are favorable here. The optimization procedure, on the other hand, generally benefits from larger batch sizes. Using a batch size of 32 for our experiments with a patch size of 128×128 px realized the best trade-off in our experiments. We used Adam [24] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a weight decay $\lambda = 0.0001$ for optimization.

The probably most crucial hyperparameter for the optimization of a deep CNN is the *learning rate* α [42]. We followed the approach of Smith [42], who propose to run training for few iterations, *i.e.*, optimization steps, with increasing learning rate. By starting from minuscule learning rates and exponentially increasing them to excessively high values, we cover the full space of realistic α for training with only 250 steps of optimization. Repeating this experiment starting from different α and using varying parameters for the exponential scheduler, we obtained an α vs. \mathcal{L}_{mt} diagram, shown in Figure 4. Four typical intervals of α can be observed. Too conservative settings with $\alpha < 10^{-6}$ result in a stagnation of \mathcal{L}_{mt} . The second interval of values $10^{-6} < \alpha < 10^{-3}$ yields a decrease of \mathcal{L}_{mt} , marking the desired range. When surpassing this interval, \mathcal{L}_{mt} no longer decreases and finally diverges for $\alpha > 10^{-1}$. From these findings, we deem learning rates in the second interval as suitable and selected $\alpha = 10^{-4}$, which is close to the steepest decent of \mathcal{L}_{mt} , for our experiments. We applied a polynomial scheduler with $\gamma = 0.9$ to slowly decay α to zero over $5 \cdot 10^5$ training

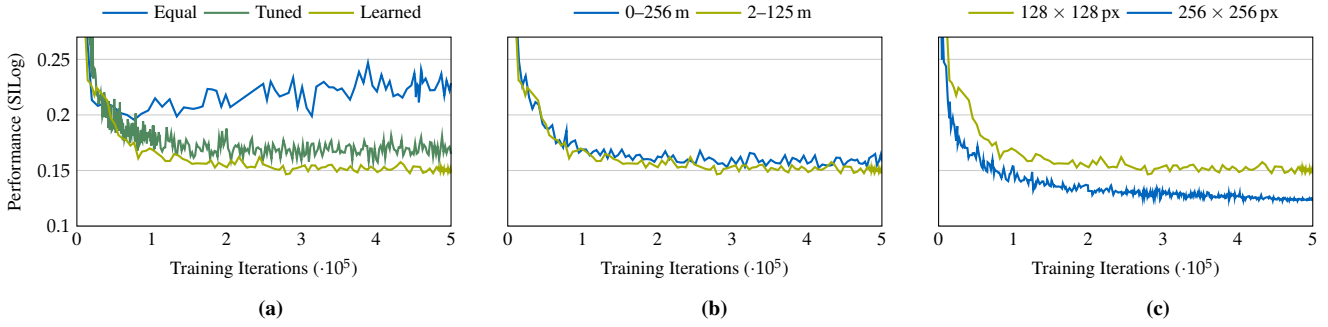


Figure 6: Validation results using different settings for (a) multi-task weights, (b) depth value scaling, and (c) patch size. Models using learned weighting, normalization bounds adapted to the data distribution, and large patches for training yield best results with weighting being the most important factor.

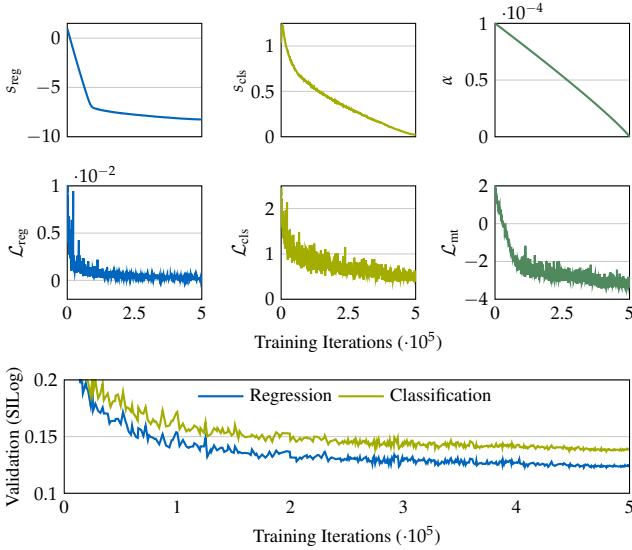


Figure 7: Training results for the final model configuration showing an initial phase of adjusting the weights s_{task} according to the uncertainty of $\mathcal{L}_{\text{task}}$ followed by a continuous decrease of both. The combined \mathcal{L}_{m} is optimized using a decaying α . Validation results show the stable convergence of both outputs with the regression yielding superior results due to quantization errors in the auxiliary classification output.

iterations, which roughly equals 90 epochs. Note that due to random cropping of patches from the input data, the number of epochs is not a very well-suited measure here.

3.5 Ablation Studies

Evaluation of our approach with learned and dynamic weighting between the differently posed depth estimation tasks that mutually support each other requires a careful protocol. To analyze the proposed approach, we conducted a number of ablation studies and present the results in the following subsections. Training was conducted from scratch without using pre-trained weights for the ResNet encoder. Networks were trained from scratch on a simple desktop machine, equipped with a single NVIDIA GeForce 1080Ti GPU, in approximately 5 days.

3.5.1 Number of Classes

To obtain a baseline result, we trained the regression branch of our network without the additional classification branch. As

seen in Figure 5, the single-task results get better at first but diverge after iteration 10^5 . Convergence towards an undesirable local minimum using vanilla regression has also been reported by Fu et al. [12]. We also trained a model for single-task classification by removing the regression branch from the network architecture. As expected, casting SIDE to a depth interval classification task eliminates the problem of slow and unstable convergence and yields much better results overall.

Following our MultiDepth approach by adding an auxiliary classification branch to the regression network helps to stabilize training, even when using as little as two classes. This result is related to the observations of Li et al. [32], who include a term for foreground/background separation in their loss function. Increasing the number of classes further improves stability during training and the overall performance. However, using $n_{\text{cls}} \geq 4$ only yields minor improvements. Another notable outcome is that using more classes than necessary does not significantly degrade results. It does, however, yield a secondary output that suffers from far lower quantization errors and could, thus, potentially be used as a redundant signal. An example of both outputs is shown in Figure 1. Models with $4 \leq n_{\text{cls}} \leq 64$ achieved almost identical results on the validation set. We selected $n_{\text{cls}} = 32$ for further experiments due to best convergence.

3.5.2 Multi-Task Weighting

In order to evaluate the influence of learned weights [23] on the training process, we ran additional experiments with equally weighted tasks and manually tuned weighting terms. Figure 6a shows the training progress, evaluated on the validation set, for configurations with $n_{\text{cls}} = 32$ (cf. Section 3.5.1). The baseline with fixed and equal weights for both tasks converges to an unstable local minimum and its best result is significantly worse than the performance of the model trained with learned weights. Manually tuned pairs of fixed weights led to much better results compared to the equally weighted baseline, but still perform significantly worse than the learned dynamic weighting, as seen in Table 1. In our experiments, learned sets of weights always outperformed manually tuned weights, showing the effectiveness of this technique. Note that manual tuning becomes increasingly complex when adding more tasks [23, 33], but even for the limited set of tasks used in our experiments, learning the

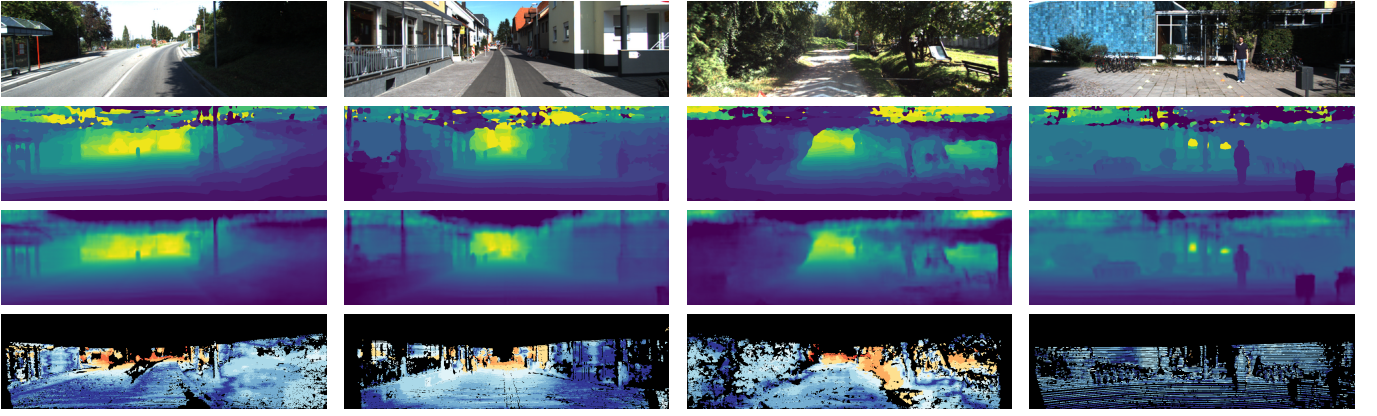


Figure 8: Qualitative evaluation of our estimation results for unseen KITTI depth prediction test images [45, 14] (first row) with results of the auxiliary classification task (second row), results of the main regression task (third row) and color coded error images of the regression result compared to the sparse LiDAR point cloud (fourth row).

weighting terms saves precious computation time while introducing as little as two additional parameters to the optimization problem.

3.5.3 Log-Normalization of Depth Values

In the data pre-processing stage, which was done online in parallel threads, we scaled the depth values according to Equation (2). We conducted dedicated experiments on how setting the limits d_{\min} and d_{\max} for the normalization influences the quality of the final results. We set them according to the minimum and maximum depth present in the dataset (*cf.* Figure 3d), which is a good estimate of real-world conditions, and compared them to scaling from 0–256 m, which are the theoretical bounds for the depth encoding in the KITTI data format. Again, both training runs were conducted using $n_{\text{cls}} = 32$ (*cf.* Section 3.5.1) and learned task weights (*cf.* Section 3.5.2). The former outperformed the latter, as scaling to the relevant range spreads the numerical precision with respect to data and application. Note that this relevant range of depth values might differ for other applications, such as indoor settings.

3.5.4 Patch Size

In a final study, we trained on different patch sizes. As already noted earlier, this is especially relevant due to the sparse ground-truth data. In order to show this, we trained on bigger patches of 256×256 px and report the results in Figure 6c. Since training with similar settings as in the preceding experiments but a bigger patch size is more memory intensive, we ran this experiment on two NVIDIA 1080Ti GPUs using parallelization with batch splitting and a batch size of 16. Training using bigger patches outperformed training on smaller ones, which was expected. The results achieved using this configuration yielded the best results in our series of experiments and were thus submitted to the KITTI depth prediction benchmark for final scoring.

3.6 Final Training Results

Based on our experiments (see Section 3.5), we selected the model with the highest validation score for submission to the

Table 1: Performance of models covering multiple aspects of MultiDepth on the validation set. Classification results given for $n_{\text{cls}} = 32$. Regression and classification predictions were inferred in a single pass for the MultiDepth models. All configurations (with a negligible exception) outperform the single-task baselines by far.

Experiment			Results (SILog)	
Method	Patch Size	Weighting	Classification	Regression
Baseline	128×128 px	single-task	—	25.96
Baseline	128×128 px	single-task	17.22	—
MultiDepth	128×128 px	equal	17.63	19.56
MultiDepth	128×128 px	man. tuned	15.99	15.75
MultiDepth	128×128 px	learned	15.62	14.65
MultiDepth	256×256 px	learned	13.70	12.27

KITTI depth prediction benchmark. Table 1 lists the performance of the most relevant configurations. Figure 7 shows the training progress including the task weights, the learning rate, single-task and multi-task losses, and results on the validation set for the outputs of the regression and classification branch. Since the task uncertainties differ widely, as seen in \mathcal{L}_{reg} and \mathcal{L}_{cls} , s_{reg} quickly shifted to a suitable range (note that due to log-scaling in the weighting function $s_{\text{reg}} < 0$ corresponds to weights close to zero). After a minor initial increase, s_{cls} stabilizes and slowly decreases together with s_{reg} over the course of training to account for the converging single-task losses. Validation results show the expected superiority of the regression over the classification results due to quantization errors, good convergence, and no signs of overfitting.

Depth maps for the images of the anonymous test set were predicted using the regression branch of the final model and submitted to the KITTI depth prediction benchmark where they scored an SILog of 16.05. Top ranking methods achieve comparable to better scores (DORN [12]: 11.77, VGG16-UNet [18]: 13.41, HGR framework [50]: 15.47). Qualitative results, given in Figure 8, show that our network is able to infer useful depth maps overall. Note that we did not implement any pre-training or post-processing in order to enhance the prediction results since this might interfere with convergence during training—thus rendering the observation of relative improvements made using our method impossible.

4 Conclusion

We presented MultiDepth, a method for introducing an auxiliary depth interval classification task to SIDE networks. An implementation of our network has been applied to the challenging problem of SIDE in road scene understanding. In extensive experiments, we showed the benefits of posing SIDE as a multi-task problem with additional supervision through automatically derived ground-truth labels.

The single-task regression and classification baselines were outperformed by far using our method with 32 classes for the auxiliary classification branch and learned weighting between the tasks. MultiDepth achieved an SILog of 16.05 on the anonymous KITTI depth prediction test set [45, 14], showing that our training strategy also yields good results overall.

Source code for all experiments presented in this paper is publicly available online¹.

References

- [1] Y. Cao, Z. Wu, and C. Shen. “Estimating Depth From Monocular Images as Classification Using Deep Fully Convolutional Residual Networks”. In: *TCSVT* 28.11 (2018), pp. 3174–3182.
- [2] Rich Caruana. “Multitask Learning”. In: *Machine Learning* 28.1 (1997), pp. 41–75.
- [3] Richard Caruana. “Multitask Learning: A Knowledge-Based Source of Inductive Bias”. In: *ICML*. 1993, pp. 41–48.
- [4] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Celine Teuliere, and Thierry Chateau. “Deep MANTA: A Coarse-To-Fine Many-Task Network for Joint 2D and 3D Vehicle Analysis From Monocular Image”. In: *CVPR*. 2017, pp. 1827–1836.
- [5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs”. In: *ICLR*. 2015, pp. 1–14. arXiv: 1412.7062v4 [cs.CV].
- [6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. “Multi-View 3D Object Detection Network for Autonomous Driving”. In: *CVPR*. 2017, pp. 6526–6534.
- [7] Xinjing Cheng, Peng Wang, and Ruigang Yang. “Depth Estimation via Affinity Learned with Convolutional Spatial Propagation Network”. In: *ECCV*. 2018, pp. 108–125.
- [8] Sumanth Chennupati, Ganesh Sistu, Senthil Yogamani, and Samir Rawashdeh. “AuxNet: Auxiliary tasks enhanced Semantic Segmentation for Automated Driving”. In: *VISAPP*. 2019, pp. 1–8. arXiv: 1901.05808v1 [cs.CV].
- [9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *CVPR*. 2016, pp. 3213–3223.
- [10] David Eigen and Rob Fergus. “Predicting Depth, Surface Normals and Semantic Labels With a Common Multi-Scale Convolutional Architecture”. In: *ICCV*. 2015, pp. 2650–2658.
- [11] David Eigen, Christian Puhrsch, and Rob Fergus. “Depth Map Prediction from a Single Image using a Multi-Scale Deep Network”. In: *NIPS*. 2014, pp. 2366–2374.
- [12] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. “Deep Ordinal Regression Network for Monocular Depth Estimation”. In: *CVPR*. 2018, pp. 2002–2011.
- [13] Yukang Gan, Xiangyu Xu, Wenxiu Sun, and Liang Lin. “Monocular Depth Estimation with Affinity, Vertical Pooling, and Label Enhancement”. In: *ECCV*. 2018, pp. 232–247.
- [14] A Geiger, P Lenz, C Stiller, and R Urtasun. “Vision meets robotics: The KITTI dataset”. In: *Int. J. Robotics Res.* 32.11 (2013), pp. 1231–1237.
- [15] Clement Godard, Oisín Mac Aodha, and Gabriel J. Brostow. “Unsupervised Monocular Depth Estimation With Left-Right Consistency”. In: *CVPR*. 2017, pp. 6602–6611.
- [16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [17] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. “Dynamic Task Prioritization for Multitask Learning”. In: *ECCV*. 2018, pp. 282–299.
- [18] Xiaoyang Guo, Hongsheng Li, Shuai Yi, Jimmy Ren, and Xiaogang Wang. “Learning Monocular Depth by Distilling Cross-domain Stereo Networks”. In: *ECCV*. 2018, pp. 506–523.
- [19] Akhil Gurram, Onay Urfalioglu, Ibrahim Halfaoui, Fahd Bouzaraa, and Antonio M. Lopez. “Monocular Depth Estimation by Learning from Heterogeneous Datasets”. In: *IV*. 2018, pp. 2176–2181.
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *CVPR*. 2016, pp. 770–778.
- [21] Minhyeok Heo, Jaehan Lee, Kyung-Rae Kim, Han-UI Kim, and Chang-Su Kim. “Monocular Depth Estimation Using Whole Strip Masking and Reliability-Based Refinement”. In: *ECCV*. 2018, pp. 39–55.
- [22] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. “The ApolloScape Dataset for Autonomous Driving”. In: *CVPR Workshops*. 2018, pp. 1067–1037.
- [23] Alex Kendall, Yarin Gal, and Roberto Cipolla. “Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics”. In: *CVPR*. 2018, pp. 7482–7491.
- [24] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *ICLR*. 2015, pp. 1–15.
- [25] Tobias Koch, Lukas Liebel, Friedrich Fraundorfer, and Marco Körner. “Evaluation of CNN-based Single-Image Depth Estimation Methods”. In: *ECCV Workshops*. 2018, pp. 331–348.
- [26] Shu Kong and Charles Fowlkes. “Pixel-wise Attentional Gating for Scene parsing”. In: *WACV*. 2019, pp. 1024–1033.
- [27] Yevhen Kuznetsov, Jorg Stuckler, and Bastian Leibe. “Semi-Supervised Deep Learning for Monocular Depth Map Prediction”. In: *CVPR*. 2017, pp. 2215–2223.
- [28] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. “Deeper depth prediction with fully convolutional residual networks”. In: *3DV*. 2016, pp. 239–248.
- [29] Bo Li, Yuchao Dai, and Mingyi He. “Monocular depth estimation with hierarchical fusion of dilated CNNs and soft-weighted-sum inference”. In: *Pattern Recognit.* 83 (2018), pp. 328–339.

¹<https://github.com/lukasliebel/MultiDepth>

- [30] Jun Li, Reinhard Klein, and Angela Yao. “A Two-Streamed Network for Estimating Fine-Scaled Depth Maps From Single RGB Images”. In: *CVPR*. 2017, pp. 3372–3380.
- [31] Ruibo Li, Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, and Lingxiao Hang. “Deep attention-based classification network for robust depth prediction”. In: *ACCV*. (ACCV). 2018, pp. 1–17. eprint: 1807.03959.
- [32] Zhengqi Li and Noah Snavely. “MegaDepth: Learning Single-View Depth Prediction From Internet Photos”. In: *CVPR*. 2018, pp. 2041–2050.
- [33] Lukas Liebel and Marco Körner. “Auxiliary Tasks in Multi-task Learning”. In: (2018), pp. 1–8. arXiv: 1805.06334v2 [cs.CV].
- [34] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. “PlaneNet: Piece-wise Planar Reconstruction from a Single RGB Image”. In: *CVPR*. 2018, pp. 2579–2588.
- [35] Fayao Liu, Chunhua Shen, and Guosheng Lin. “Deep convolutional neural fields for depth estimation from a single image”. In: *CVPR*. 2015, pp. 5162–5170.
- [36] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. “Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights”. In: *ECCV*. 2018, pp. 72–88.
- [37] Xiaojuan Qi, Renjie Liao, Zhengzhe Liu, Raquel Urtasun, and Jiaya Jia. “GeoNet: Geometric Neural Network for Joint Depth and Surface Normal Estimation”. In: *CVPR*. 2018, pp. 283–291.
- [38] Zhongzheng Ren and Yong Jae Lee. “Cross-Domain Self-Supervised Multi-Task Feature Learning Using Synthetic Imagery”. In: *CVPR*. 2018, pp. 762–771.
- [39] E. Romera, L. M. Bergasa, J. M. Alvarez, and M. Trivedi. “Train Here, Deploy There: Robust Segmentation in Unseen Domains”. In: *IV*. 2018, pp. 1828–1833.
- [40] Sebastian Ruder. “An Overview of Multi-Task Learning in Deep Neural Networks”. In: (2017), pp. 1–14. arXiv: 1706.05098v1 [cs.LG].
- [41] Ozan Sener and Vladlen Koltun. “Multi-Task Learning as Multi-Objective Optimization”. In: *NIPS*. 2018, pp. 525–536.
- [42] L. N. Smith. “Cyclical Learning Rates for Training Neural Networks”. In: *WACV*. 2017, pp. 464–472.
- [43] Nikolai Smolyanskiy, Alexey Kamenev, and Stan Birchfield. “On the Importance of Stereo for Accurate Depth Estimation: An Efficient Semi-Supervised Deep Neural Network Approach”. In: *CVPR Workshops*. 2018, pp. 1120–1128.
- [44] Marvin Teichmann, Michael Weber, J. Marius Zöllner, Roberto Cipolla, and Raquel Urtasun. “MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving”. In: *IV*. 2018, pp. 1013–1020.
- [45] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. “Sparsity Invariant CNNs”. In: *3DV*. 2017, pp. 11–20.
- [46] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. “PAD-Net: Multi-Tasks Guided Prediction-and-Distillation Network for Simultaneous Depth Estimation and Scene Parsing”. In: *CVPR*. 2018, pp. 675–684.
- [47] Guorun Yang, Hengshuang Zhao, Jianping Shi, Zhidong Deng, and Jiaya Jia. “SegStereo: Exploiting Semantic Information for Disparity Estimation”. In: *ECCV*. 2018, pp. 660–676.
- [48] Fisher Yu and Vladlen Koltun. “Multi-Scale Context Aggregation by Dilated Convolutions”. In: *ICLR*. 2016, pp. 1–13.
- [49] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Zequn Jie, Xiang Li, and Jian Yang. “Joint Task-Recursive Learning for Semantic Segmentation and Depth Estimation”. In: *ECCV*. 2018, pp. 238–255.
- [50] Zhenyu Zhang, Chunyan Xu, Jian Yang, Ying Tai, and Liang Chen. “Deep hierarchical guidance and regularization learning for end-to-end depth estimation”. In: *Pattern Recognit.* 83 (2018), pp. 430–442.
- [51] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. “Pyramid Scene Parsing Network”. In: *CVPR*. 2017, pp. 6230–6239.
- [52] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. “A Modulation Module for Multi-task Learning with Applications in Image Retrieval”. In: *ECCV*. 2018, pp. 415–432.
- [53] Nikolaos Zioulis, Antonis Karakottas, Dimitrios Zarpalas, and Petros Daras. “OmniDepth: Dense Depth Estimation for Indoors Spherical Panoramas”. In: *ECCV*. 2018, pp. 453–471.