MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving

Marvin Teichmann[123], Michael Weber[2], Marius Zöllner[2], Roberto Cipolla[3] and Raquel Urtasun[1]

[1] Department of Computer Science, University of Toronto
[2] FZI Research Center for Information Technology, Karlsruhe
[3] Department of Engineering, University of Cambridge

marvin.teichmann@googlemail.com, Michael.Weber@fzi.de,
zoellner@fzi.de, rc10001@cam.ac.uk, urtasun@cs.toronto.edu

Abstract

While most approaches to semantic reasoning have focused on improving performance, in this paper we argue that computational times are very important in order to enable real time applications such as autonomous driving. Towards this goal, we present an approach to joint classification, detection and semantic segmentation via a unified architecture where the encoder is shared amongst the three tasks. Our approach is very simple, can be trained end-to-end and performs extremely well in the challenging KITTI dataset, outperforming the state-of-the-art in the road segmentation task. Our approach is also very efficient, taking less than 100 ms to perform all tasks.

1. Introduction

Current advances in the field of computer vision have made clear that visual perception is going to play a key role in the development of self-driving cars. This is mostly due to the deep learning revolution which begun with the introduction of AlexNet in 2012 [23]. Since then, the accuracy of new approaches has been increasing at a vertiginous rate. Causes of this are the existence of more data, increased computation power and algorithmic developments. The current trend is to create deeper networks with as many layers as possible [17].

While performance is extremely high, when dealing with real-world applications, running times become important. New hardware accelerators as well as compression, reduced precision and distillation methods have been exploited to speed up current networks.

In this paper we take an alternative approach and design a network architecture that can very efficiently perform classification, detection and semantic segmentation simultaneously. This is done by incorporating all three task into a unified encoder-decoder architecture. We name our approach MultiNet. The encoder consists of the convolution and pooling layers from the VGG network [45] and is shared among all tasks. Those features are then utilized by task-specific decoders, which produce their outputs in real-time. In particular, the detection decoder combines the fast regression design introduced in Yolo [38] with the size-adjusting ROI-Pooling of Fast-RCNN [14], achieving a better speed-accuracy ratio.

Figure 1: Our goal: Solving street classification, vehicle detection and road segmentation in one forward pass.

We demonstrate the effectiveness of our approach in the challenging KITTI benchmark [13] and show state-of-the-art performance in road segmentation. Importantly, our ROI-Pooling implementation can significantly improve detection performance without requiring an explicit proposal generation network. This gives our decoder a significant speed advantage compared to Faster-RCNN. Our approach is able to benefit from sharing computations, allowing us to perform inference in less than 100 ms for all tasks.

2. Related Work

In this section we review current approaches to the tasks that MultiNet tackles, i.e., detection, classification and semantic segmentation. We focus our attention on deep learning based approaches.

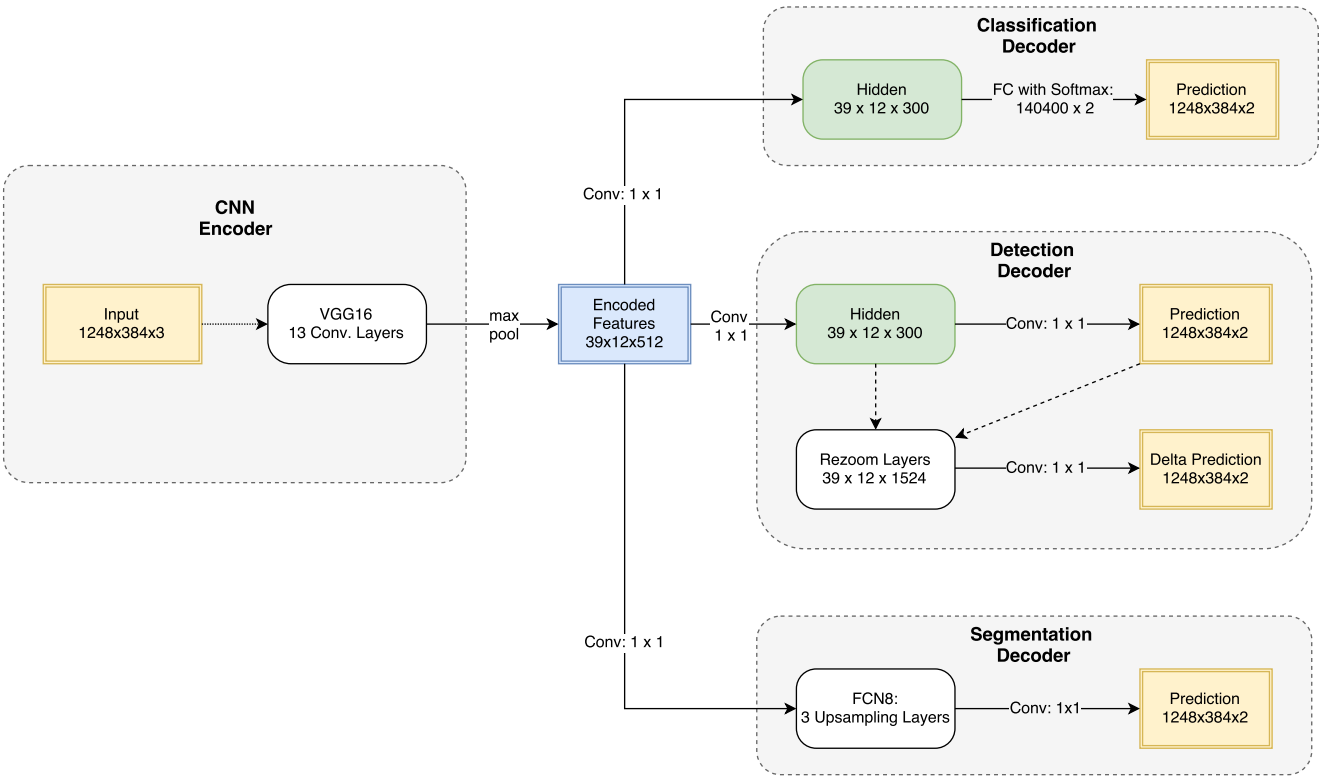

Figure 2: MultiNet architecture.

Classification: After the development of AlexNet [23], most modern approaches to image classification utilize deep learning. Residual networks [17] constitute the state-of-the-art, as they allow to train very deep networks without problems of vanishing or exploding gradients. In the context of road classification, deep neural networks are also widely employed [31]. Sensor fusion has also been exploited in this context [43]. In this paper we use classification to guide other semantic tasks, i.e., segmentation and detection.

Detection: Traditional deep learning approaches to object detection follow a two step process, where region proposals [25, 20, 19] are first generated and then scored using a convolutional network [15, 40]. Additional performance improvements can be gained by using convolutional neural networks (CNNs) for the proposal generation step [8, 40] or by reasoning in 3D [5, 4]. Recently, several methods have proposed to use a single deep network that is trainable end-to-end to directly perform detection [44, 38, 39, 27]. Their main advantage over proposal-based methods is that they are much faster at both training and inference time, and thus more suitable for real-time detection applications. However, so far they lag far behind in performance. In this paper we propose an end-to-end trainable detector which reduces significantly the performance gap. We argue that the main advantage of proposal-based methods is their ability to have size-adjustable features. This inspired our zoom layer that as shown in our experience results in large improvements in performance.

Segmentation: Inspired by the successes of deep learning, CNN-based classifiers were adapted to the task of semantic segmentation. Early approaches used the inherent efficiency of CNNs to implement implicit sliding-window [16, 26]. Fully Convolutional Networks (FCNs) were proposed to model semantic segmentation using a deep learning pipeline that is trainable end-to-end. Transposed convolutions [50, 6, 21] are utilized to upsample low resolution features. A variety of deeper flavors of FCNs have been proposed since [1, 34, 41, 36]. Very good results are archived by combining FCNs with conditional random fields (CRFs) [52, 2, 3]. [52, 42] showed that mean-field inference in the CRF can be cast as a recurrent net allowing end-to-end training. Dilated convolutions were introduced in [48] to augment the receptive field size without losing resolution. The aforementioned techniques in conjunction with residual networks [17] are currently the state-of-the-art.

Joint Reasoning: Multi-task learning techniques aim at learning better representations by exploiting many tasks.

Several approaches have been proposed in the context of CNNs [30, 28] but applications have mainly been focussed on face recognition tasks [51, 47, 37]. [18] reasons jointly about classification and segmentation using an SVM in combination with dynamic programming. [46] proposed to use a CRF to solve many tasks including detection, segmentation and scene classification. In the context of deep learning, [7] proposed a model which is able to jointly perform pose estimation and object classification. To our knowledge no unified deep architecture has been proposed to solve segmentation, classification and detection.

3. MultiNet for Joint Semantic Reasoning

In this paper we propose an efficient and effective feed-forward architecture, which we call *MultiNet*, to jointly reason about semantic segmentation, image classification and object detection. Our approach shares a common encoder over the three tasks and has three branches, each implementing a decoder for a given task. We refer the reader to Fig. 2 for an illustration of our architecture. MultiNet can be trained end-to-end and joint inference over all tasks can be done in less than 100ms. We start our discussion by introducing our joint encoder, follow by the task-specific decoders.

The task of the encoder is to process the image and extract rich abstract features [49] that contain all necessary information to perform accurate segmentation, detection and image classification. The encoder of MultiNet consists of the first 13 layers of the VGG16 network [45], which are applied in a fully convolutional manner to the image producing a tensor of size $39 \times 12 \times 512$. This is the output of the 5^{th} pooling layer, which is called *pool5* in the VGG implementation [45].

3.1. Classification Decoder

The classification decoder is designed to take advantage of the encoder. Towards this goal, we apply a 1×1 convolution followed by a fully connected layer and a softmax layer to output the final class probabilities.

3.2. Detection Decoder

FastBox, our detection decoder, is designed to be a regression based detection system. We choose such a decoder over a proposal based one because it can be train end-to-end, and both training and inference can be done very efficiently. Our approach is inspired by ReInspect [39], Yolo [38] and Overfeat [44]. In addition to the standard regression pipeline, we include an ROI pooling approach, which allows the network to utilize features at a higher resolution, similar to the much slower Faster-RCNN.

The first step of our decoder is to produce a rough estimate of the bounding boxes. Towards this goal, we first pass the encoded features through a 1×1 convolutional layer with 500 filters, producing a tensor of shape $39 \times 12 \times 500$, which we call *hidden*. This tensor is processed with another 1×1 convolutional layer which outputs 6 channels at resolution 39×12. We call this tensor *prediction*, the values of the tensor have a semantic meaning. The first two channels of this tensor form a coarse segmentation of the image. Their values represent the confidence that an object of interest is present at that particular location in the 39×12 grid. The last four channels represent the coordinates of a bounding box in the area around that cell. Fig. 3 shows an image with its cells.

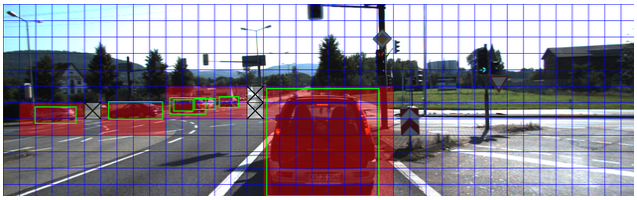

Figure 3: Visualization of our label encoding. Blue grid: cells, Red cells: cells containing a car, Grey cells: cells in don't care area. Green boxes: ground truth boxes.

Such prediction, however, is not very accurate. In this paper we argue that this is due to the fact that resolution has been lost by the time we arrive to the encoder output. To alleviate this problem we introduce a rezoom layer, which predicts a residual on the locations of the bounding boxes by exploiting high resolution features. This is done by concatenating subsets of higher resolution VGG features (156×48) with the hidden features (39×12) and applying 1×1 convolutions on top of this. In order to make this possible, a 39×12 grid needs to be generated out of the high resolution VGG features. This is achieved by applying ROI pooling [40] using the rough prediction provided by the tensor prediction. Finally, this is concatenated with the $39 \times 12 \times 6$ features and passed through a 1×1 convolution layer to produce the residuals.

3.3. Segmentation Decoder

The segmentation decoder follows the FCN architecture [29]. Given the encoder, we transform the remaining fully-connected (FC) layers of the VGG architecture into 1×1 convolutional layers to produce a low resolution segmentation of size 39×12. This is followed by three transposed convolution layers [6, 21] to perform up-sampling. Skip layers are utilized to extract high resolution features from the lower layers. Those features are first processed by a 1×1 convolution layer and then added to the partially up-sampled results.

4. Training Details

In this section we describe the loss functions we employ as well as other details of our training procedure including initialization.

Label encoding: We use one-hot encoding for classification and segmentation. For the detection, we assigned a positive confidence if and only if it intersects with at least one bounding box. We parameterize the bounding box by the x and y coordinate of its center and the width w and height h of the box. Note that this encoding is much simpler than FasterRCNN or ReInspect.

Loss Functions: We define our loss function as the sum of the loss functions for classification, segmentation and detection. We employ cross-entropy as loss function for the classification and segmentation branches, which is defined as

$$\text{loss}_{\text{class}}(p, q) := -\frac{1}{|I|} \sum_{i \in I} \sum_{c \in C} q_i(c) \log p_i(c) \quad (1)$$

where p is the prediction, q the ground truth and C the set of classes. We use the sum of two losses for detection: Cross entropy loss for the confidences and an L1 loss on the bounding box coordinates. Note that the L1 loss is only computed for cells which have been assigned a positive confidence label. Thus

$$\text{loss}_{\text{box}}(p, q) := \frac{1}{|I|} \sum_{i \in I} \delta_{q_i} \cdot (|x_{p_i} - x_{q_i}| + |y_{p_i} - y_{q_i}| + |w_{p_i} - w_{q_i}| + |h_{p_i} - h_{q_i}|) \quad (2)$$

where p is the prediction, q the ground truth, C the set of classes and I is the set of examples in the mini batch.

Combined Training Strategy: Joint training is performed by merging the gradients computed by each loss on independent mini batches. This allows us to train each of the three decoders with their own set of training parameters. During gradient merging all losses are weighted equally. In addition, we observe that the detection network requires more steps to be trained than the other tasks. We thus sample our mini batches such that we alternate an update using all loss functions with two updates that only utilize the detection loss.

Initialization: The encoder is initialized using pretrained VGG weights on ImageNet. The detection and classification decoder weights are randomly initialized using a uniform distribution in the range $(-0.1, 0.1)$. The convolutional layers of the segmentation decoder are also initialized using VGG weights and the transposed convolution layers are initialized to perform bilinear upsampling. The skip connections on the other hand are initialized randomly with very small weights (i.e. std of $1e-4$). This allows us to perform training in one step (as opposed to the two step procedure of [29]).

Experiment	max steps	eval steps [k]
Segmentation	16,000	100
Classification	18,000	200
Detection	180,000	1000
United	200,000	1000

Table 1: Summary of training length.

Optimizer and regularization: We use the Adam optimizer [22] with a learning rate of $1e-5$ to train our MultiNet. A weight decay of $5e-4$ is applied to all layers and dropout with probability 0.5 is applied to all (inner) 1×1 convolutions in the decoder.

5. Experimental Results

In this section we perform our experimental evaluation on the challenging KITTI dataset.

5.1. Dataset

We evaluate MultiNet in he KITTI Vision Benchmark Suite [12]. The Benchmark contains images showing a variety of street situations captured from a moving platform driving around the city of Karlruhe. In addition to the raw data, KITTI comes with a number of labels for different tasks relevant to autonomous driving. We use the road benchmark of [10] to evaluate the performance of our semantic segmentation decoder and the object detection benchmark [13] for the detection decoder. We exploit the automatically generated labels of [31], which provide us with road labels generated by combining GPS information with open-street map data.

Detection performance is measured using the average precision score [9]. For evaluation, objects are divided into three categories: easy, moderate and hard to detect. The segmentation performance is measured using the MaxF1 score [10]. In addition, the average precision score is given for reference. Classification performance is evaluated by computing accuracy and precision-recall plots.

5.2. Performance evaluation

Our evaluation is performed in two steps. First we build three individual models consisting of the VGG-encoder and the decoder corresponding to the task. Those models are tuned to achieve highest possible performance on the given task. In a second step MultiNet is trained using one encoder

Metric	Result
MaxF1	95.83 %
Average Precision	92.29 %
Speed (msec)	94.6 ms
Speed (fps)	10.6 Hz

Table 2: Validation performance of the segmentation decoder.

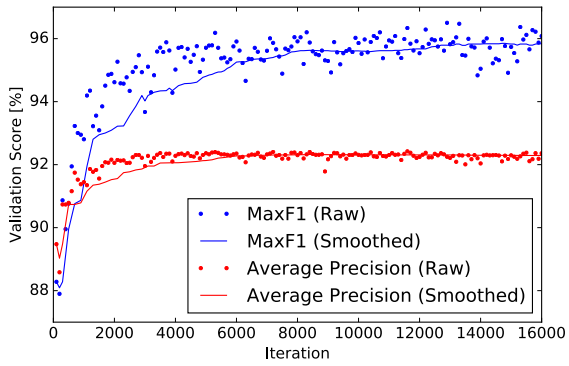

Figure 4: Convergence behavior of the segmentation decoder.

and three decoders in a single network. We evaluate both settings in our experimental evaluation. We report a set of plots depicting the convergence properties of our networks in Figs. 4, 6 and 8. Evaluation on the validation set is performed every k iterations during training, where k for each tasks is given in Table 1. To reduce the variance in the plots the output is smoothed by computing the median over the last 50 evaluations performed.

Segmentation: Our Segmentation decoder is trained using the KITTI Road Benchmark [10]. This dataset is very small, providing only 289 training images. Thus the network has to transfer as much knowledge as possible from pre-training. Note that the skip connections are the only layers which are randomly initialized and thus need to be trained from scratch. This transfer learning approach leads to very fast convergence of the network. As shown in Fig. 4 the raw scores already reach values of about 95 % after only about 4000 iterations. Training is conducted for 16,000 iterations to obtain a meaningful median score.

Table 2 shows the scores of our segmentation decoder after 16,000 iterations. The scores indicate that our segmentation decoder generalizes very well using only the data given by the KITTI Road Benchmark. No other segmentation dataset was utilized. As shown in Fig. 5, our approach is very effective at segmenting roads. Even difficult areas,

Method	MaxF1	AP	Place
FCN_LC [32]	90.79 %	85.83 %	5th
FTP [24]	91.61 %	90.96 %	4th
DDN [33]	93.43 %	89.67 %	3th
Up_Conv_Poly [35]	93.83 %	90.47 %	2rd
MultiNet	**94.88%**	**93.71%**	1st

Table 3: Summary of the URBAN_ROAD scores on the public KITTIRoad Detection Leaderboard [11].

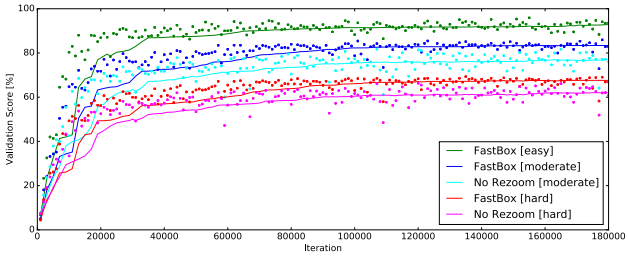

Figure 6: Validation scores of the detection decoder. Performance of FastBox with and without rezoom layer is shown for comparison.

corresponding to sidewalks and buildings are segmented correctly. In the confidence plots shown in top two rows of Fig. 5, it can be seen that our approach has confidence close to 0.5 at the edges of the street. This is due to the slight variation in the labels of the training set. We have submitted the results of our approach on the test set to the KITTI road leaderboard. As shown in Table 3, our result achieve first place.

Detection: Our detection decoder is trained and evaluated on the data provided by the KITTI object benchmark [13]. Fig. 6 shows the convergence rate of the validation scores. The detection decoder converges much slower than the segmentation and classification decoders. We therefore train the decoder up to iteration 180,000.

FastBox can perform evaluation at very high speed: an inference step takes 37.49 ms per image. This makes FastBox particularly suitable for real-time applications. Our results indicate further that the computational overhead of the rezoom layer is negligible (see Table 5). The performance boost of the rezoom layer on the other hand is quite substantial (see Table 4), justifying the use of a rezoom layer in the final model. Qualitative results are shown in Fig. 7 with and without non-maxima suppression.

MultiNet: We have experimented with two versions of MultiNet. The first version is trained using two decoders, (detection and segmentation) while the second version is

Figure 5: Visualization of the segmentation output. Top rows: Soft segmentation output as red blue plot. The intensity of the plot reflects the confidence. Bottom rows hard class labels.

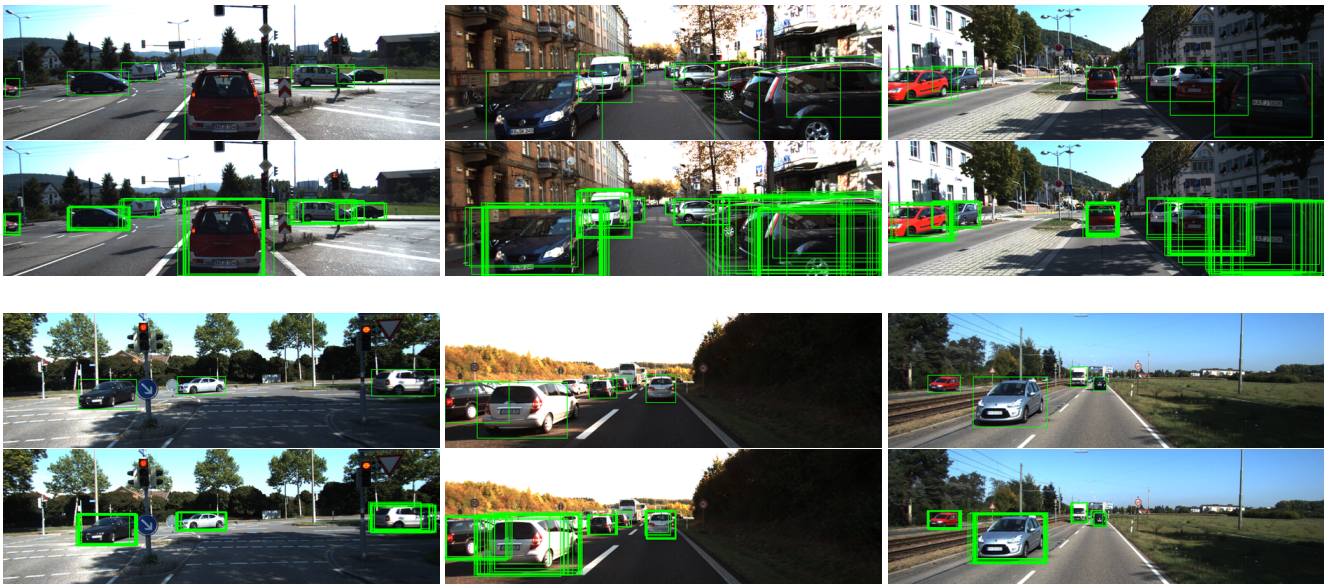

Figure 7: Visualization of the detection output. With and without non-maximal suppression applied.

Task: Metric	moderate	easy	hard
FastBox with rezoom	83.35 %	92.80 %	67.59 %
FastBox no rezoom	77.00 %	86.45 %	60.82 %

Table 4: Detection performance of FastBox.

	FastBox	FastBox (no rezoom)
speed [msec]	37.49 ms	35.75 ms
speed [fps]	26.67 Hz	27.96 Hz
post-processing	2.10 ms	2.46 ms

Table 5: Detection speed of FastBox. Results are measured on a Pascal Titan X.

trained with all three decoders. Training with additional decoders significantly lowers the convergence speed of all decoders. When training with all three decoders it takes segmentation more than 30.000 and detection more than

Task: Metric	seperate	2 losses	3 losses
Segmentation: MaxF1	**95.83%**	94.98 %	95.13 %
Detection: Moderate	83.35 %	83.91 %	**84.39%**
Classification: Accuracy	92.65 %	—	**94.38%**

Table 6: MultiNet performance: Comparison between united and seperate evaluation on the validation set.

MultiNet	Segmentation	Detection	Classification
98.10 ms	94.6 ms	37.5 ms	35.94 ms
10.2 Hz	10.6 Hz	27.7 Hz	27.8 Hz

Table 7: MultiNet inference speed: Comparision between united and seperate evaluation.

150.000 iterations to converge, as shown in Fig. 8. Fig. 8 and Table 6 also show, that our combined training does not harm performance. On the contrary, the detection and classification tasks benefit slightly when jointly trained. This effect can be explained by transfer learning between tasks: relevant features learned from one task can be utilized in a different task.

MultiNet is particularly suited for real-time applications. As shown in Table 7 computational complexity benefits significantly from a shared architecture. Overall, MultiNet is able to solve all three task together in real-time.

6. Conclusion

In this paper we have developed a unified deep architecture which is able to jointly reason about classification, detection and semantic segmentation. Our approach is very simple, can be trained end-to-end and performs extremely well in the challenging KITTI, outperforming the state-of-the-art in the road segmentation task. Our approach is also very efficient, taking 98.10 ms to perform all tasks. In the future we plan to exploit compression methods in order to further reduce the computational bottleneck and energy consumption of MutiNet.

Acknowledgements: This work was partially supported by Begabtenstiftung Informatik Karlsruhe, ONR-N00014-14-1-0232, Qualcomm, Samsung, NVIDIA, Google, EPSRC and NSERC. We are thankful to Thomas Roddick for proofreading the paper.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015. 2

[2] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014. 2

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016. 2

[4] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016. 2

[5] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2015. 2

[6] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016. 2, 3

[7] M. Elhoseiny, T. El-Gaaly, A. Bakry, and A. M. Elgammal. Convolutional models for joint object categorization and pose estimation. *CoRR*, abs/1511.05175, 2015. 3

[8] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. *CoRR*, abs/1312.2249, 2013. 2

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 4

[10] J. Fritsch, T. Kuehnl, and A. Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013. 4, 5

[11] A. Geiger. Kitti road public benchmark, 2013. 5

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 4

[13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 4, 5

[14] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. 1

[15] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. 2

[16] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. *CoRR*, abs/1302.1700, 2013. 2

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1, 2

[18] M. Hoai, Z.-Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3265–3272. IEEE, 2011. 3

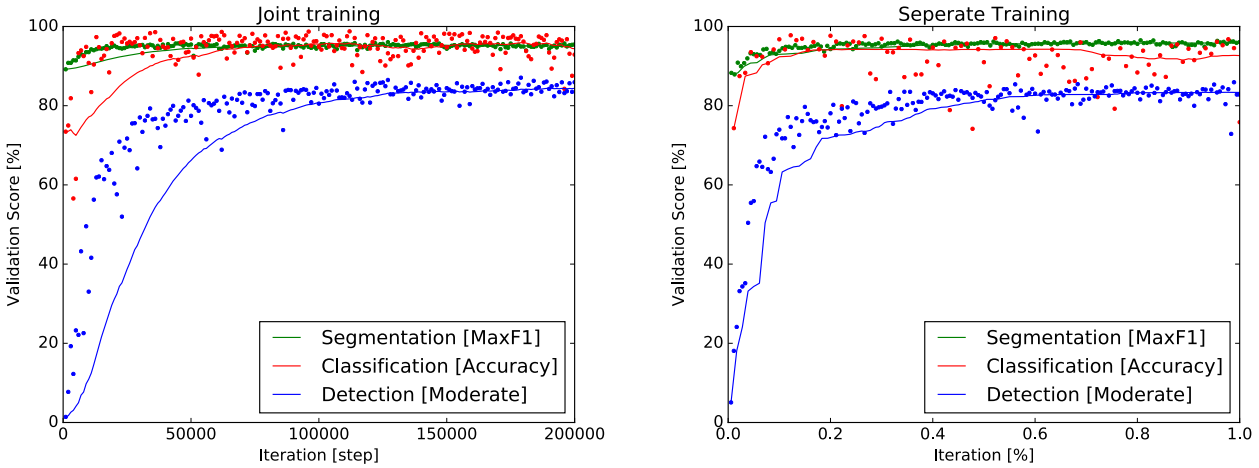

Figure 8: MultiNet: Comparison of Joint and Separate Training.

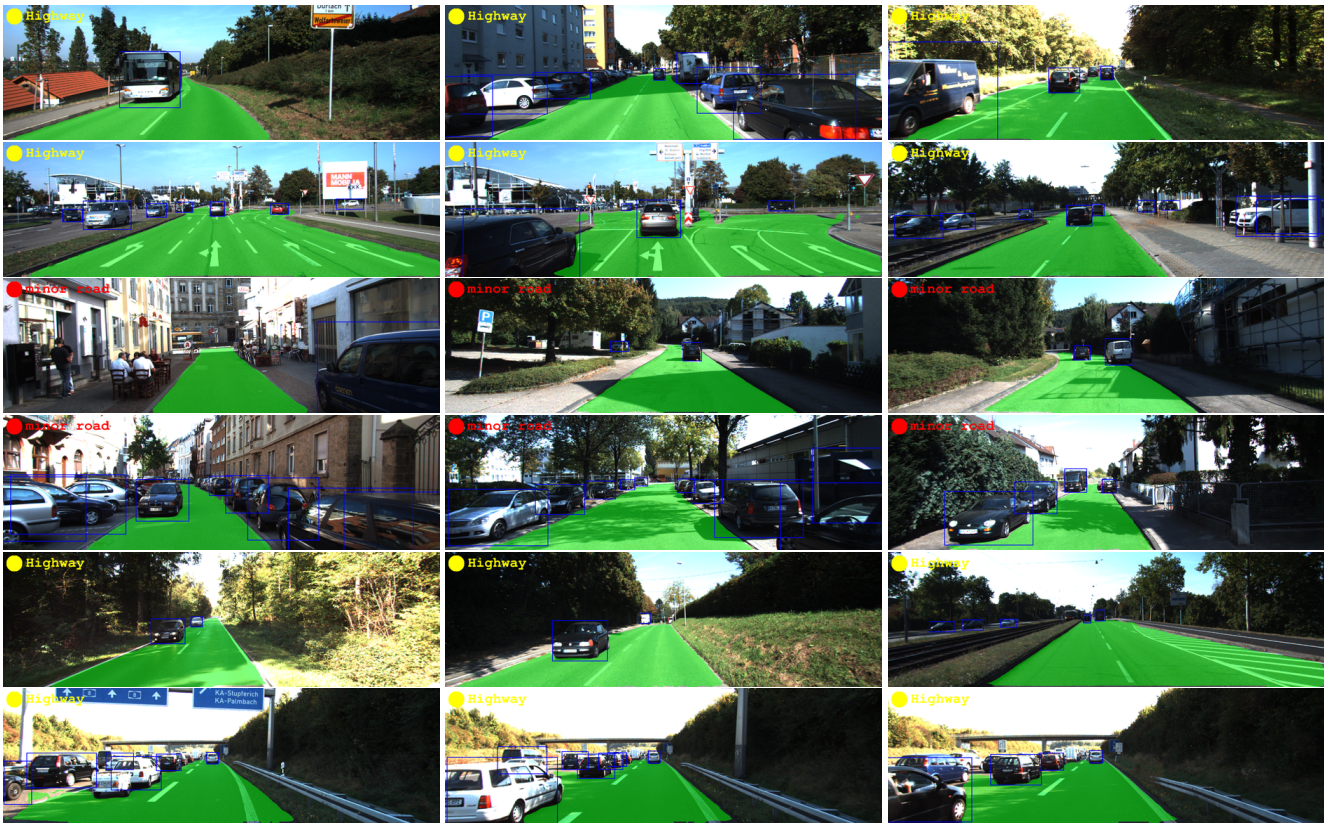

Figure 9: Visualization of the MultiNet output.

[19] J. H. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *CoRR*, abs/1502.05082, 2015. 2

[20] J. H. Hosang, R. Benenson, and B. Schiele. How good are detection proposals, really? *CoRR*, abs/1406.6962, 2014. 2

[21] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic. Generating images with recurrent adversarial networks. *CoRR*, abs/1602.05110, 2016. 2, 3

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 4

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In

F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 1, 2

[24] A. Laddha, M. K. Kocamaz, L. E. Navarro-Serment, and M. Hebert. Map-supervised road detection. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 118–123, June 2016. 5

[25] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2

[26] H. Li, R. Zhao, and X. Wang. Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification. *CoRR*, abs/1412.4526, 2014. 2

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. 2

[28] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proc. NAACL*, 2015. 3

[29] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR (to appear)*, Nov. 2015. 3, 4

[30] M. Long and J. Wang. Learning multiple tasks with deep relationship networks. *CoRR*, abs/1506.02117, 2015. 3

[31] W.-C. Ma, S. Wang, M. A. Brubaker, S. Fidler, and R. Urtasun. Find your way by observing the sun and other semantic cues. *arXiv preprint arXiv:1606.07415*, 2016. 2, 4

[32] C. C. T. Mendes, V. Frmont, and D. F. Wolf. Exploiting fully convolutional neural networks for fast road detection. In *IEEE Conference on Robotics and Automation (ICRA)*, May 2016. 5

[33] R. Mohan. Deep deconvolutional networks for scene parsing, 2014. 5

[34] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. 2015. 2

[35] G. Oliveira, W. Burgard, and T. Brox. Efficient deep methods for monocular road segmentation. 2016. 5

[36] G. Papandreou, L. Chen, K. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a DCNN for semantic image segmentation. *CoRR*, abs/1502.02734, 2015. 2

[37] 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. *CoRR*, abs/1603.01249, 2016. 3

[38] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 1, 2, 3

[39] M. Ren and R. S. Zemel. End-to-end instance segmentation and counting with recurrent attention. *CoRR*, abs/1605.09410, 2016. 2, 3

[40] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 2, 3

[41] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 2

[42] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *CoRR*, abs/1503.02351, 2015. 2

[43] C. Seeger, A. Müller, L. Schwarz, and M. Manz. Towards road type classification with occupancy grids. *IVS Workshop*, 2016. 2

[44] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013. 2, 3

[45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1, 3

[46] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 702–709. IEEE, 2012. 3

[47] J. Yim, H. Jung, B. Yoo, C. Choi, D. Park, and J. Kim. Rotating your face using multi-task deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 676–684, 2015. 3

[48] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015. 2

[49] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014. 3

[50] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010. 2

[51] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014. 3

[52] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. *CoRR*, abs/1502.03240, 2015. 2