MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving

Marvin Teichmann[123], Michael Weber[2], Marius Zöllner[2], Roberto Cipolla[3] and Raquel Urtasun[14]

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

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, allowing us to perform inference at more then 23 frames per second.

Training scripts and trained weights to reproduce our results can be found here: https://github.com/MarvinTeichmann/MultiNet

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 [29]. 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 [22].

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

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

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 is a deep CNN, producing rich features that are shared among all task. 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 [45] with the size-adjusting ROI-align of Faster-RCNN [17]and Mask-RCNN [21], achieving a better speed-accuracy ratio.

We demonstrate the effectiveness of our approach in the challenging KITTI benchmark [15] and show state-of-the-art performance in road segmentation. Importantly, our ROI-align 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 [46]. Our approach is able to benefit from sharing computations, allowing us to perform inference in less than 45 ms for all tasks.

All our code, training scripts and weights, required to reproduce our results, are released on Github.

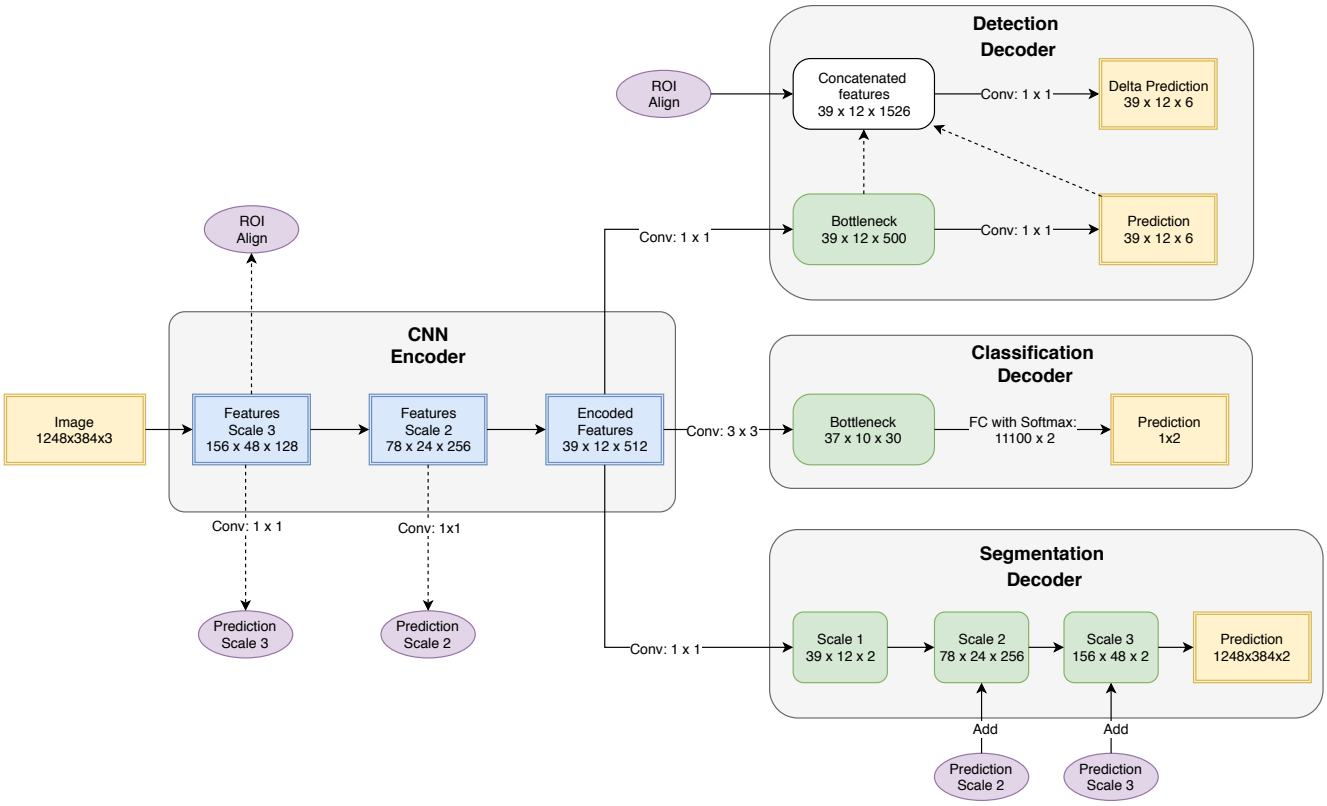

Figure 2: MultiNet architecture.

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.

Classification: After the development of AlexNet [29], most modern approaches to image classification utilize deep learning. Residual networks [22] 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 [37]. Sensor fusion has also been exploited in this context [50]. 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 [31, 25, 24] are first generated and then scored using a convolutional network [18, 46]. Additional performance improvements can be gained by using convolutional neural networks (CNNs) for the proposal generation step [10, 46] or by reasoning in 3D [6, 5]. Recently, several methods have proposed to use a single deep network that is trainable end-to-end to directly perform detection [51, 33, 53, 33]. 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 ROI pooling implementation.

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 [19, 32]. FCN were proposed to model semantic segmentation using a deep learning pipeline that is trainable end-to-end. Transposed convolutions [58, 9, 26] are utilized to upsample low resolution features. A variety of deeper flavors of FCNs have been proposed since [1, 40, 47, 42]. Very good results are archived by combining FCN with conditional random fields (CRFs) [60, 3, 4]. [60, 49] 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 [56] to augment the receptive field size without losing resolution. The aforementioned techniques in conjunction with residual networks [22] are currently the state-of-the-art.

Multi-Task Learning: Multi-task learning techniques aim at learning better representations by exploiting many tasks. Several approaches have been proposed in the context of CNNs [36, 34]. An important application for multi-task learning is face recognition [59, 55, 44].

Learning semantic segmentation in order to perform detection or instance segmentation has been studied [16, 7, 43]. In those systems, the main goal is to perform an instance level task. Semantic annotation is only viewed as an intermediate result. Systems like [51, 54] and many more design one system which can be fine-tuned to perform tasks like classification, detection or semantic segmentation. In this kind of approaches, a different set of parameters is learned for each task. Thus, joint inference is not possible in this models. The system described in [20] is closest to our model. However this system relies on existing object detectors and does not fully leverage the rich features learned during segmentation for both tasks. To the best of our knowledge our system is the first one proposed which is able to do this.

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 45ms. We start our discussion by introducing our joint encoder, followed by the task-specific decoders.

3.1. Encoder

The task of the encoder is to process the image and extract rich abstract features [57] that contain all necessary information to perform accurate segmentation, detection and image classification. The encoder consists of the convolutional and pooling layers of a classification network. The weights of the encoder are initialized using the weights pretrained on ImageNet Classification Data [48]. As encoder any modern classification network can be utilized.

We perform experiments using versions of VGG16 [57] and ResNet [22] architectures. Our first VGG encoder uses all convolutional and pooling layers of VGG16. but discards the fully-connected softmax layers. We call this version *VGG-pool5*, as pool5 is the last layer used from VGG16. The second implementation only discards the final fully-connected softmax layer. We call this architecture *VGG-fc7*, as fc7 is the last layer used from VGG16. VGG-fc7 utilizes two fully-connected layers from VGG, namely *fc6* and *fc7*. We replace those layers with equal 1×1 convolutions as discussed in [51, 35]. This trick allows the encoder to process images with arbitrary input size. In particular we are not bound to the original VGG input of 224×224, which would be to small to perform perception in street scenes.

For ResNet we implement the 50 and 101 layer Version of the Network. As encoder we utilize all layers apart from the layers fully-connected softmax.

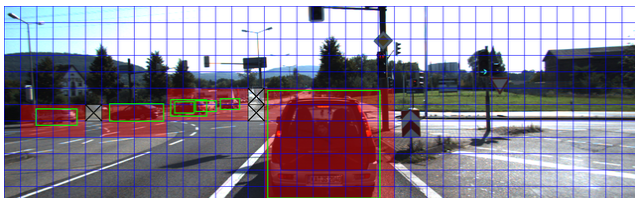

Figure 3: Visualization of our detection encoding. Blue grid: cells, Red cells: cells with positive confidence label. Transparent Cells: cells with negative confidence label. Grey cells: cells in don't care area. Green boxes: ground truth boxes.

3.2. Classification Decoder

We implement two classification decoders. One version is a vanilla fully-connected layer with softmax activation. This encoder is used in conjunction with an input size of 224×224. Thus, the overall network is equal to the original VGG or ResNet respectively, when used with the corresponding encoder. The purpose of this encoder is to serve as high quality baseline to show the effectiveness of our scene classification approach. This first classification encoder cannot be used for joint inference with segmentation and detection. Both approaches require a larger input size. Increasing the input size on this classification encoder however, yields into an unreasonable high amount of parameters for the final layer.

The second classification decoder is designed to take advantage of the high resolution features our encoder generates. In typical image classification tasks (e.g. [48, 28]) the input features one object, usually centred prominently in the image. For this kind of task it is reasonable to use a very small input size. Street scenes on the other hand contain a large amount of smaller scale objects. We argue that it is vital to use high-resolution input in order to utilize features those objects provide. By increasing the input size of our image to 1248×348, we effectively apply our feature generator to each spatial location of the image [51, 35]. The

result is a grid of 39×12 features, each corresponding to a spatial region of size 32×32 pixels. In order to utilize this features, we first apply a 1×1 convolution with 30 channels. This layer serves as *BottleNeck*. The main purpose is to greatly reduce dimensionality.

3.3. Detection Decoder

The detection decoder is designed to be a proposal free approach similar to ReInspect [53], Yolo [45] and Overfeat [51]. By omitting and artificial proposal generator step much faster inference can be obtained. This is crucial towards our goal of building a real-time capable detection system.

Proposal based detection systems have a crucial advantage over non-proposal based. They internally rescale the rich features utilized for detection. This makes the CNN internally invariant to scale. This is a crucial feature, as CNN are naturally not able to generalize over different scales. We argue, that the scale invariance is the main advantage of proposal based systems.

Our detection decoder tries to close the marry the good detection performance of proposal based detection systems with the fast speed of non-proposal based systems. To archive this, we include a rescaling layer inside the decoder. The rescaling layer consists of RoI align [21] and provides the main advantage of proposal based systems. Unlike proposal based systems, no non-differential operations are done and the rescaling can be computed very efficiently.

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 $39 \times 12 \times 500$. Those features serve as *bottleneck*. 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.

Those prediction are then utilized to introduce scale invariance. A rescaling approach, similar to the ones found in proposal based systems is applied on the initial coarse prediction. The rescaling layer follows the RoI align strategy of [21]. It uses however the prediction of each cell to produce a RoI align. This makes the operation differentiable. Thus it can be implemented insight the CNN pooling. The result is an end-to-end trainable system which is faster. The features pooled by the RoI align are concatenated with the initial prediction and used to produce a more accurate prediction. The second prediction is modeled as offset, its output is added to the initial prediction.

3.4. Segmentation Decoder

The segmentation decoder follows the main ideas of the FCN architecture [35]. Given the features produced by the encoder, we produce a low resolution segmentation of size 39×12 using a 1×1 convolutional layer. This output is then upsampled using three transposed convolution layers [9]. Skip connections 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 upsampled 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.

MultiNet Training Strategy: MultiNet training follows a fine-tuning approach. First the encoder network is trained to perform classification on the ILSVRC2012 [8] data. In practice, this step is omitted. Instead we initialize the weights of all layers of the encoder with weights published by the authors whose network architecture we are using.

In a second step, the final fully connected layers are removed and replaced by our decoders. Then the network is trained end-to-end using KITTI data. Thus MultiNet training follows a classic fine-tuning pipeline.

Our joint training implementation computes the forward passes for examples corresponding to each of the three tasks independently. The gradients are only added during the back-propagation steps. This has the practical advantage that we are able to use different training parameters for each decoder. Having this degree of freedom is an important feature of our joint training implementation. The classification task for example requires a relative large batch size and more aggressive data-augmentation than the segmentation task to perform well.

Loss function: Classification and segmentation are trained using a softmax cross-entropy loss function.

For the detection, the final prediction is a grid of 12×39 cells. Each cell gets assigned a confidence label as well as a box label. The box label encodes the coordinates of the box and is parametrized relative to the position of a cell. A cell c gets assigned a positive confidence label if and only if it intersects with at least one bounding box. If this is the case the cell also gets assigned to predict the coordinates of the box it intersects with. If multiple boxes intersect with a cell, the box whose centre is closest to the centre of c is chosen. Note that one box can be predicted by multiple cells.

If a box b is assigned to a cell c the following values are stored in c:

$$c_x = (x_b - x_c)/w_c \qquad c_y = (y_b - y_c)/h_c \quad (1)$$
$$c_w = w_b/w_c \qquad c_h = h_b/h_c \quad (2)$$

where x_b, y_b and x_c y_c correspond to the center coordinates of b and c and w and h denote width and hight. Note, that w_c and h_c are always 32, as the cells of our model have a fixed width and height. We use L1 as our loss

$$\text{loss}_{\text{cell}}(c, \hat{c}) := \delta c_p \cdot (|c_x - \hat{c}_x| + |c_y - \hat{c}_y| + |c_w - \hat{c}_w| + |c_w - \hat{c}_w|) \quad (3)$$

where $\hat{c}$ is the prediction of a cell and c its ground-truth, and c_p denotes whether a positive label has been assigned to a cell. The δc_p term ensures that the regression loss is zero if no object is present. We train the confidence labels using cross-entropy loss. The loss per cell is given as the weighted sum over the confidence and the regression loss. The loss per image is the mean over the losses of all cells. The KITTI Dataset also contains 'don't Care areas'. Those areas are handled by multiplying the loss of the corresponding cells with zero. We note, that our label representation is much simpler than Faster-RCNN or ReInspect. This is an additional feature of our detection system. The loss for MultiNet is given as the sum of the losses for segmentation, detection and classification.

The loss for the joint training is given as the sum of the losses for segmentation, detection and classification.

Initialization: The weights of the encoder are initialized using weights trained on ImageNet [8] data. The weights of the detection and classification decoder are initialized using the initialization scheme of [23]. The transposed convolution layers of the segmentation decoder are initialized to perform bilinear upsampling. The skip connections of the segmentation decoder are initialized to very small weights. Both these modifications greatly improve segmentation performance.

Optimizer and regularization: We use the Adam optimizer [27] with a learning rate of 10^{-5} to train our MultiNet. A weight decay of $5 \cdot 10^{-4}$ is applied to all layers and dropout with probability 0.5 is applied to the 3×3 convolution of the classification and all 1×1 convolutions of the detection decoder.

Standard data augmentation are applied to increase the amount of effective available training data. We augment colour features by applying random brightness and random contrast. Spatial feature are distorted by applying random flip, random resize and random crop.

Method	MaxF1	AP	Place
FTP [30]	91.61 %	90.96 %	6th
DDN [38]	93.43 %	89.67 %	5th
Up_Conv_Poly [41]	93.83 %	90.47 %	4rd
DEEP-DIG [39]	93.83 %	90.47 %	3th
LoDNN [2]	94.07 %	92.03 %	2rd
MultiNet	**94.88%**	**93.71%**	1st

Table 1: Summary of the URBAN ROAD scores on the public KITTIRoad Detection Leaderboard [13].

5. Experimental Results

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

5.1. Dataset

We evaluate MultiNet on the KITTI Vision Benchmark Suite [14]. The Benchmark contains images showing a variety of street situations captured from a moving platform driving around the city of Karlsruhe. 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 [12] to evaluate the performance of our semantic segmentation decoder and the object detection benchmark [15] for the detection decoder. We exploit the automatically generated labels of [37], 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 [11]. For evaluation, objects are divided into three categories: easy, moderate and hard to detect. The segmentation performance is measured using the MaxF1 score [12]. In addition, the average precision score is given for reference. Classification performance is evaluated by computing the mean accuracy, precision and recall.

5.2. Experimental evaluation

The section is structured as fellows. We first evaluate the performance of the three decoders individually. To do this we fine-tune the encoder using just one of the three losses segmentation, detection and classification and compare their performance with a variety of baseline. In the second part we compare joint training of all three decoders with individual inference and show, that the performance of joint training can keep up with the performance of individual inferences. Overall we show, that our approach is competitive with individual inference. This makes our approach very relevant. Joint training has many advantages in robotics application, such as a fast inference time.

Figure 4: Visualization of the segmentation output. Top row: Soft segmentation output as red blue plot. The intensity of the plot reflects the confidence. Bottom row hard class labels.

Task: Metric	MaxF1	AP
VGG-pool5	95.80 %	92.19 %
ResNet50	95.89 %	92.10 %
VGG-fc7	95.94 %	92.24 %
ResNet101	**96.29 %**	**92.32 %**

Table 2: Performance of the segmentation decoder.

Task: Metric	moderate	easy	hard
VGG no RIO pool	77.00 %	86.45 %	60.82 %
Faster-RCNN	78.42 %	91.62 %	66.85 %
VGG-pool5	84.76 %	92.18 %	68.23 %
ResNet50	86.63 %	95.55 %	74.61 %
ResNet101	**89.79 %**	**96.13 %**	**77.65 %**

Table 3: Performance of our detection decoder.

	speed [msec]	speed [fps]
VGG-pool5	**42.14 ms**	**23.73 Hz**
ResNet50	39.56 ms	25.27 Hz
VGG-fc7	96.84 ms	10.32 Hz
ResNet101	69.91 ms	14.30 Hz

Table 4: Inference speed of our segmentation.

	speed [msec]	speed [fps]	processing
VGG no RIO	**35.75 ms**	**27.96 Hz**	2.46 ms
Faster-RCNN	78.42 ms	12.75 Hz	5.3 ms
VGG-pool5	37.31 ms	26.79 Hz	3.61 ms
ResNet50	40.09 ms	24.93 Hz	3.19 ms
ResNet101	65.89 ms	15.17 ms	3.11 ms

Table 5: Inference speed of our detection decoder.

Segmentation: The segmentation decoder encoder is trained using the four different encoders discussed in Section 3.1. The scores, computed on a halt-out validation set is reported in Table 1.

To compare my approach against the state-of-the-art we trained a segmentation network with VGG-fc7 encoder on the whole training set and submitted the results to the KITTI road leaderboard. At submission time my approach archived first place in the benchmark. Recently my approach was overtaken by newer submissions. All non-anonymous submissions to the benchmark are shown in Table 2.

Qualitative results are shown in Fig. 4 both as red blue plot showing the confidence level at each pixel as well as a hard prediction using a threshold of 0.5.

Detection: The detection decoder is trained and evaluated on the data provided by the KITTI object benchmark [15]. We train the detection decoder on a VGG [52] and ResNet [22] decoder and evaluate on a validation set. Table 3 shows the results of our decoder compared to a Faster-RCNN baseline, evaluated on the same validation set. The results show that our rescaling approach is very efficient. Training the detection decoder with rescaling is only marginality slower then training it without. However it offers a significant improvement in detection performance. Overall our approach archives is speed-up over faster-rcnn of almost a factor 2 and outperforms its detection accuracy. Qualitative results of the detection decoder can be seen in 5.

All in all my results indicate that utilizing a rescaling layer in order to archive scale invariance is a good idea. A rescaling layer might be the key to closing the gap between proposal and non-proposal based approaches.

Our detection decoder is trained and evaluated on the data provided by the KITTI object benchmark [15]. We train our detection decoder on a VGG [52] and ResNet [22] decoder and evaluate on a validation set. Table 3 shows the results of our decoder compared to a Faster-RCNN baseline, evaluated on the same validation set. We report the inference speed in Table 5. We observe that our approach archives is speed-up over faster-rcnn of almost a factor 2 and outperforms its detection accuracy. This makes our de-

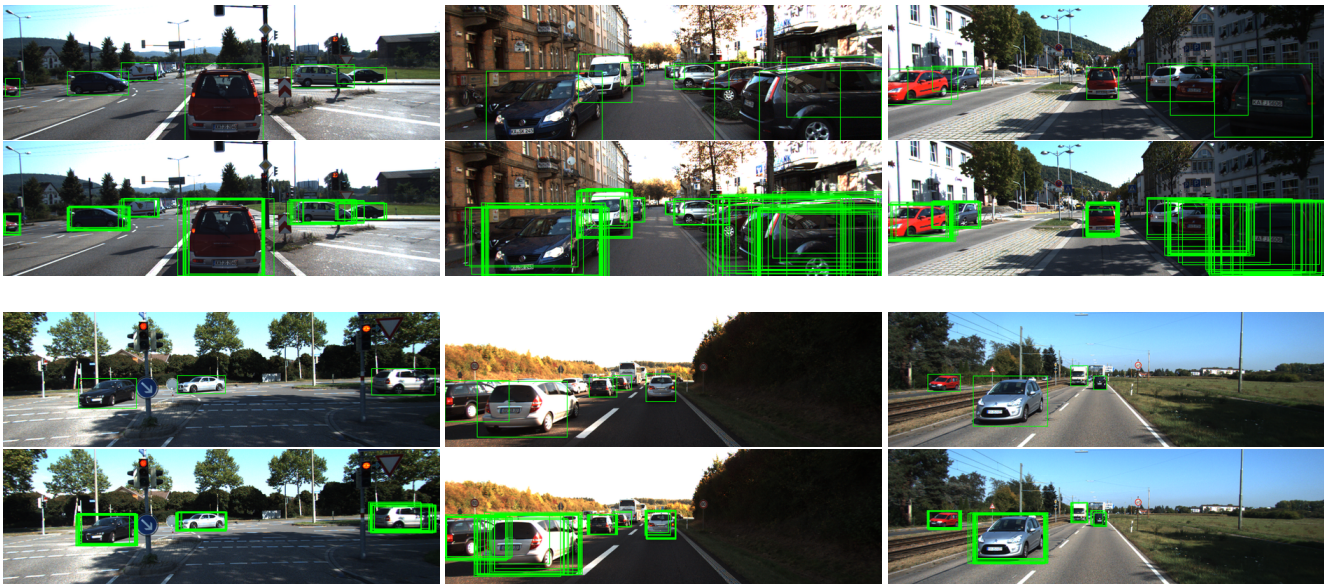

Figure 5: Visualization of the detection output. With and without non-maximal suppression applied.

	mean Acc.	Precision	Recall
VGG pool5 [our]	97.34 %	98.52 %	87.58 %
ResNet50 [our]	98.86 %	**100.00 %**	94.11 %
ResNet101 [our]	**99.84 %**	98.70 %	**100.00 %**
VGG16 [base]	93.04 %	91.61 %	87.90 %
ResNet101 [base]	93.83 %	91.94 %	89.54 %

Table 6: Classification performance of our decoder compared to baseline classification.

	speed [msec]	speed [fps]
VGG pool5 [our]	**37.83 ms**	**26.43 Hz**
ResNet50 [our]	44.27 ms	27.96 Hz
ResNet101 [our]	71.62 ms	22.58 Hz
VGG16 [base]	7.10 ms	140 Hz
ResNet101 [base]	33.06 ms	30.24 Hz

Table 7: Inference speed of our classification.

coder particularly suitable for real-time applications. Qualitative results of our detection decoder can be seen in 5.

Classification: The classification data is not part of the official KITTI Benchmark. To evaluate the classification decoder we first need to create our own dataset. This is done using the method descriped in [37]. To obtain a meaningful task all images of one scene ether fully in the train or fully in the validation set. This is important as the images of one scene are usually visually very similar.

We use a vanilla ResNet and VGG classification approach as baseline and compare this to a VGG and ResNet approach with my classification decoder. The differences between those two approaches are discussed in more detail in Section 3.2. The results are reported in Table 6 and Table 7. Our customised classification decoder clearly outperforms vanilla decoders, showing the effectiveness of my approach.

MultiNet: We ran a series of experiments comparing VGG and ResNet as encoder. Table 8 and Table 9 compare performance of VGG and ResNet. We observe, that both ResNet-based encoders are able to outperform VGG slightly. There is however a trade-off, as the VGG encoder is faster.

The speed gap between VGG pool5 and ResNet50 is much larger when performing joint inference compared to the individual task. This can be explained by the fact that ResNet computes features with 2048 channels, while VGG features have only 512 channels. Thus, computing the fist layer of each decoder is significantly more expensive.

Overall we conclude, that MultiNet using a VGG decoder offers a very good trade-off between performance and speed.

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

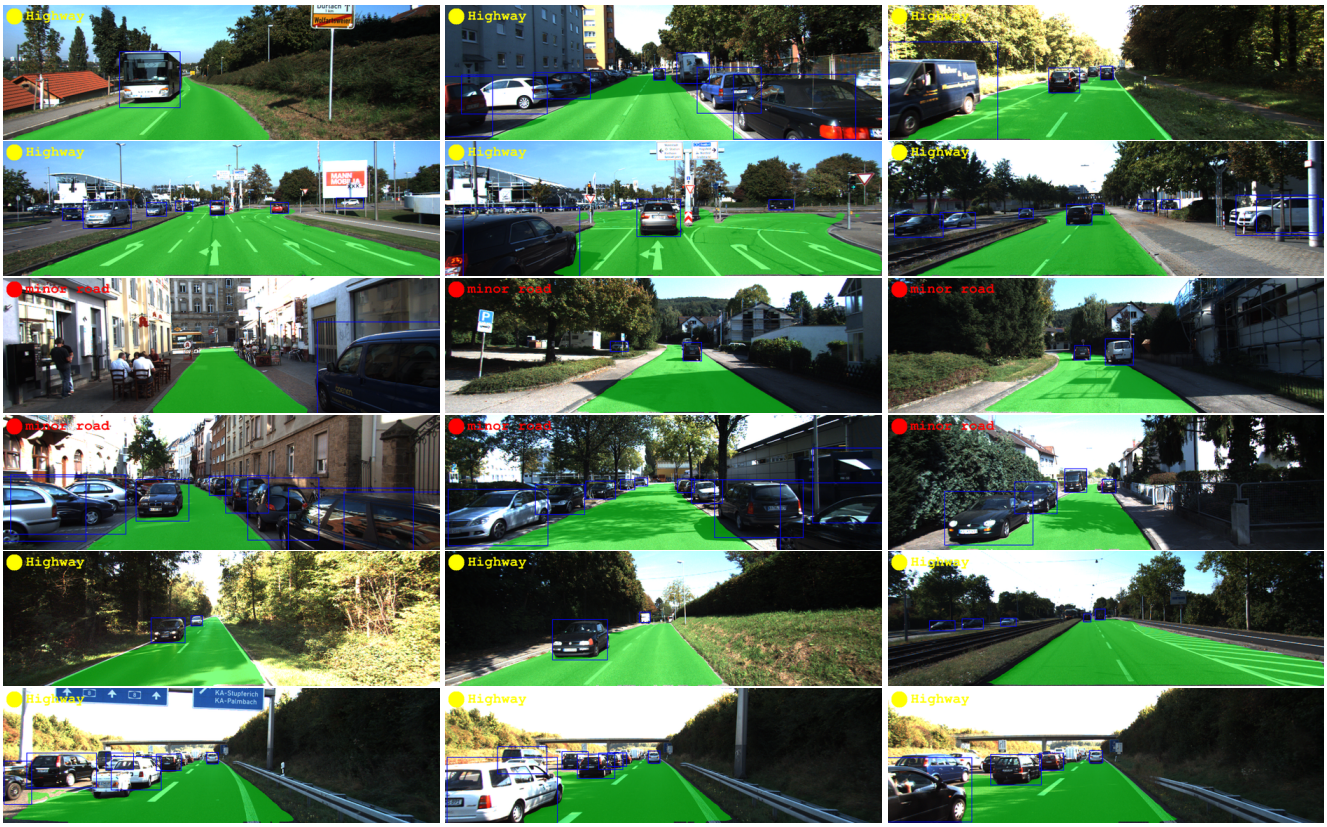

Figure 6: Visualization of the MultiNet output.

	MaxF1	AP	moderate	easy	hard	m. Acc.	Precision	Recall
VGG pool5	95.99 %	92.31 %	84.68 %	92.06 %	72.08 %	95.75 %	100 %	91.50 %
ResNet50	**96.35 %**	92.13 %	86.92 %	**96.84 %**	72.75 %	98.36 %	100 %	96.73 %
ResNet101	95.99 %	**91.99 %**	**89.30 %**	96.31 %	**75.42 %**	**98.61 %**	99.33 %	**97.38 %**

Table 8: Results of joint training

	speed [msec]	speed [fps]
VGG pool5	**42.48 ms**	**23.53 Hz**
ResNet50	60.22 ms	16.60 Hz
ResNet101	79.70 ms	12.54 Hz

Table 9: Speed of joint inference.

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 42.48 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. Caltagirone, S. Scheidegger, L. Svensson, and M. Wahde. Fast lidar-based road detection using convolutional neural networks. *arXiv preprint arXiv:1703.03613*, 2017. 5

[3] 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

[4] 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

[5] 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

[6] 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

[7] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016. 3

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255, June 2009. 4, 5

[9] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016. 2, 4

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

[11] 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. 5

[12] 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. 5

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

[14] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 5

[15] 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, 5, 6

[16] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1134–1142, 2015. 3

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

[18] 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

[19] 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

[20] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014. 3

[21] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. 1, 4

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1, 2, 3, 6

[23] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. 5

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

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

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

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[28] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research). 3

[29] 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

[30] 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

[31] 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

[32] 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

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

[34] 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

[35] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 3, 4

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

[37] 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, 5, 7

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

[39] J. Muoz-Bulnes, C. Fernandez, I. Parra, D. Fernndez-Llorca, and M. A. Sotelo. Deep Fully Convolutional Networks with Random Data Augmentation for Enhanced Generalization in Road Detection. In *Submitted to the Workshop on Deep Learning for Autonomous Driving on IEEE 20th International Conference on Intelligent Transportation Systems*, Yokohama, Japan, Oct. 2017. 5

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

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

[42] 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

[43] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016. 3

[44] 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

[45] 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, 4

[46] 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. 1, 2

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

[48] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 3

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

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

[51] 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, 4

[52] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 6

[53] R. Stewart, M. Andriluka, and A. Y. Ng. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2325–2333, 2016. 2, 4

[54] Z. Wu, C. Shen, and A. van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *CoRR*, abs/1611.10080, 2016. 3

[55] 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

[56] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015. 3

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

[58] 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

[59] 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

[60] 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