Using depth estimation for better bounding box regression

S. Aklanoglu, *KIT*, Y. El himer, *KIT*, F. Retkowski, *KIT*, J. Schuck, *KIT*

I. Introduction

In this paper we summarize our findings and solutions for the advanced lab course "Cognitive Systems" at KIT (Karlsruhe Institute of Technology) and the FZI (Research Center for Information Technology). The task was to implement depth prediction for single images and use those information to improve 2D bounding box detection on vehicles using machine learning methods. An additional goal was to assess if synthetic training data from GTA V[1] was usable to accomplish this task. Therefore training the networks with synthetic data and evaluating on real images is another part in this report.

Given this task, our goal was to develop a system, which is working near real-time with at least 10 FPS. We evaluated three methods, all trained on real RGB images with depth maps and with synthetic RGB images with depth maps. First one was DeepTLR (Deep Traffic Light Recognition), which is developed at FZI and is based on Caffe[2]. The other two methods were both using two networks in different combination, the FCRN (Fully Convolutional Residual Net) for depth prediction from single RGB images and the SSD (Single Shot Detector), which can predict 2D bounding boxes.

II. Related Work

Our approaches and neural networks were based on the findings in the following four sections. To achieve state-of-the-art depth estimation results, we chose to use FCRN. To reach real-time performance for object detection we went with SSD approach. To reduce the amount of human effort required for dataset creation while simultaneously improving the performance of our models we used synthetic training data, which researches found to be beneficial in their experiments, too.

A. Fully Convolutional Residual Networks

The work in this paper is partly based on "Deeper Depth Prediction with Fully Convolutional Residual Networks"[1]. They tried to predict depth maps without stereo vision solely based on a single RGB images (monocular images). As seen in figure 1 their approach is based on a ResNet50 network which has been modified using special *deconvolutional blocks* (faster up-convolutions). They also introduced skip connections for deconvolutions and increased the time needed for a forward pass by employing *fast up-convolutions*, which use several small deconvolutions instead of a single large deconvolution. As a result the final model can predict depth maps in real-time and is at par with state-of-art solutions for precise depth map estimation.

[1] https://www.rockstargames.com/games/info/V/
[2] http://caffe.berkeleyvision.org/

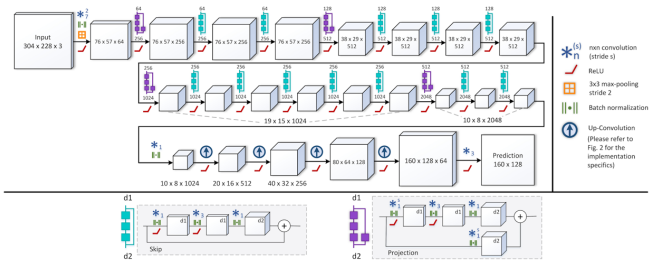
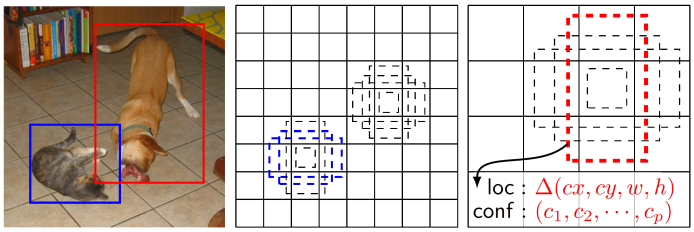


Fig. 1. Fully Convolutional Residual Net - Architecture, Source [2]

B. Single Shot Detectors

Liu et al. [3] suggest a new bounding box detector which works without an RPN (Region Proposal Network) and RoI-Pooling, making it very fast reaching almost 60 FPS. Their detector works at multiple scales, which makes it suitable for better detecting small and large objects. They achieve scores similar to Faster R-CNN [4].

The architecture is similar to Faster R-CNN using modified VGG-16 as base network to transform images to feature maps.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Fig. 2. (a) input image with ground truth (b), (c) For each default box shape offsets and confidences for each class is predicted. During training the default boxes are matched against ground truth boxes.

They predict via convolutions for each location in the feature maps:

- one confidence value per class (high confidence indicates that there is a bounding box of that class at the given location)
- x/y offsets that indicate where exactly the center of the bounding box is (e.g. a bit to the left or top of the feature map cell's center)
- height/width values that reflect the (logarithm of) the height/width of the bounding box

They also use the concept of anchor boxes: by not only generating the described values once per location, but several times for several anchor boxes (see figure 2). Each anchor box has different height/width and optionally scale.

Those predictions are also created for various feature maps in between (e.g. before pooling layers). This makes it easier for the network to detect small (as well as large) bounding boxes (multi-scale detection).

Ground truth bounding boxes have to be matched with anchor boxes (at multiple scales) to determine correct outputs. To do this, anchor boxes and ground truth bounding boxes are matched if their jaccard overlap is 0.5 or higher. Any unmatched ground truth bounding box is matched to the anchor box with highest jaccard overlap. The loss function is a mixture of classification and regression, using softmax with cross entropy for the confidence loss and smooth L1 loss for the location.

Only the anchor box with the highest loss per example image is trained (hard negative mining) and 3 in 4 boxes are negative examples. They then apply Non-Maximum-Suppression, removing bounding boxes if there is already a similar one (measured by jaccard overlap of 0.45 or more).

Their model was evaluated on Pascal VOC 2007. One of their finding was that by using multiple feature maps with different sizes (multi-scale architecture) to predict outputs, the results improved significantly by around 10% mAP. Though adding very coarse (high-level) feature maps seems to rather hurt than help. At a batch size of 1, SSD runs at about 46 FPS at input resolution 300x300 (74.3% mAP on Pascal VOC) and 19 FPS at input resolution 512x512 (76.8% mAP on Pascal VOC).

C. DeepTLR

A. Lesi improved classification and detection results with predicted depth maps from CNNs for single monocular images in his Master Thesis [5]. The designed network used multi-task learning, combining depth estimation, classification and bounding box regression in one loss function. It is based on layers from AlexNet [6] with slightly adjusted feature extraction (for a network overview see figure 3).

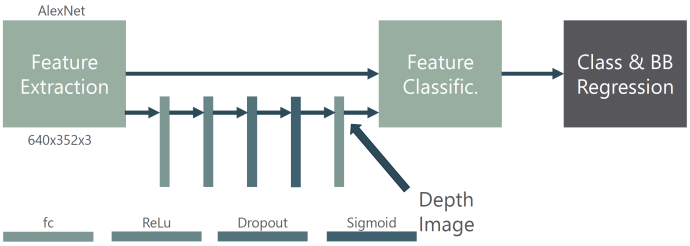


Fig. 3. DeepTLR generates a depth map from extracted features followed by convolutional blocks to improve classification and bounding box detection results.

Evaluation showed that by using depth prediction precision improved by 0.6% and recall by 1.1%, compared to the baseline without depth information. Using superpixels even improved those metrics by around 2%. Hence, joint depth prediction and object detection in one CNN is possible and object detection can profit from depth images. However, with this approach real-time performance is upheld, since precise depth images are necessary to distinct small overlapping objects.

D. Virtual Worlds Replacing Human-Generated Annotations

Until recently deep learning breakthroughs have relied upon massive amounts of human annotated training data. This time consuming process has begun impeding the progress of these deep learning efforts. Johnson Roberson et al. [7] proposed a method to incorporate photo-realistic computer images from a simulation engine to rapidly generate annotated data that can be used for the training of machine learning algorithms. Their architecture was solely trained using synthetic annotations and performs better than the identical architecture trained on human annotated real-world data, when tested on the KITTI data set for vehicle detection. By training machine learning algorithms on a rich virtual world, real objects in real scenes can be learned and classified using synthetic data.

During the same period of time [8] also showed the possibility of accelerating deep learning applications like semantic segmentation with semantic label maps extracted from DOTA 2. Using the acquired data to supplement real-world images significantly increased accuracy and reduced the amount of hand-labeled real-world data. Models trained with game data and just 1/3 of their real-world training set outperformed models trained on their complete real-world training set.

Motivated by this, we created our own synthetic dataset using GTA V to improve our detection results on the KITTI Object dataset.

III. Datasets

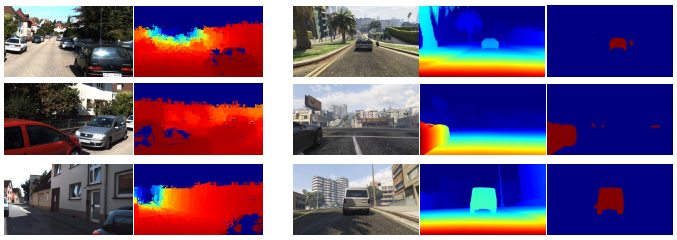


Fig. 4. Used datasets: KITTI and GTA V

For the tasks we used two datasets. The first one is from KITTI [9] [10] and from high-end graphics game GTA V as shown in figure 4. For DeepTLR and FCRN-DSSD we did convolution on KITTIs LIDAR data to close holes and to get more dense depth maps. FCRN-SSD only used discrete LIDAR data to predict depth. For GTA V we have perfect depth maps and also stencil maps, which can be used as segmentation for vehicles. KITTI consists of varying number of images, since we used two different KITTI datasets, GTA V consists of 16.000 images and 80.000 ground truth bounding boxes.

Unsurprisingly the color distribution of 15.000 analyzed images from KITTI and GTA V are differing as visualized in figure 5. Since we were in information exchange with other groups, we did not try to do some sort of histogram adjustment to match the color distribution of GTA V images with that of KITTI, since it did not significantly improve performance.

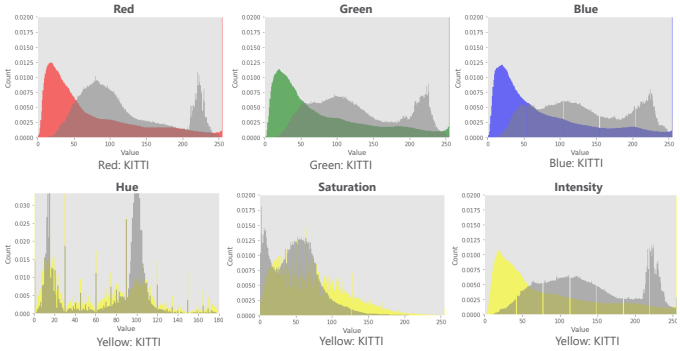


Fig. 5. Differences in color distribution are clearly

What's more interesting and has a higher impact on detection performance are sizes and distributions of the ground truth bounding boxes. In figure 6 we can see in the upper row, that the sizes of bounding boxes in KITTI are more evenly distributed than in the GTA V dataset. The GTA V dataset has a higher percentage of smaller boxes. Looking at the overlapping ratio in the second row also shows that in KITTI on average 75% of boxes overlap by 20% their area, in GTA V whereas 75% of boxes overlap at over 30%. This can have an impact on the vehicle detection task.

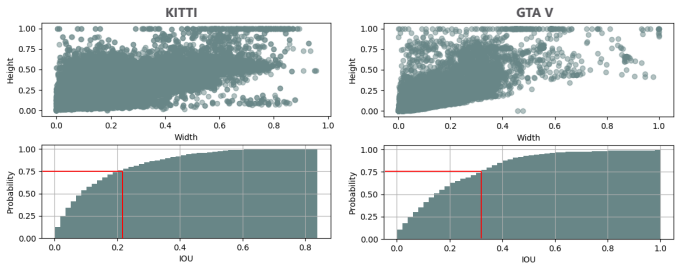


Fig. 6. Differences in ground truth bounding box sizes and overlaps

Another difference is shown in figure 7. The position of bounding box centers are normalized and plotted. Most of the boxes are on the upper left corner. This is due to the fact that most cars are on the oncoming traffic side of the road. On the other side, KITTI has more boxes and therefore cars in front of the camera on same lane or road side and therefore better distributed boxes. GTA V has 80.000 boxes, whereas KITTI had 50.000 boxes in this analyze. So a lot more boxes in GTA V are on the upper left corner than in KITTI.

IV. DeepTLR+Conv

A. Network Architecture

The architecture of this model is based on DeepTLR [11]. The CNN consists mainly of three blocks. It contains first a contractive block that progressively decreases the input image resolution through a series of convolutions and pooling layers. Giving higher level neurons large receptive fields, thus capturing more global information. This is shown in figure 8 and table I, where the first eight layers represent the contractive block of the CNN. The extracted features are then forwarded to the second main block of the network, which

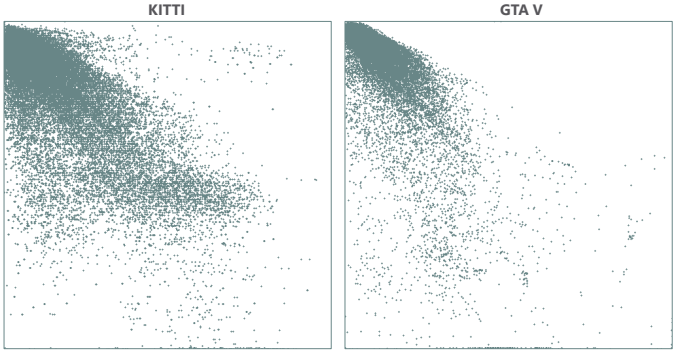


Fig. 7. Differences in ground truth bounding box positions

Layer	Channels	Kernel size	Stride	Padding
Conv1	96	11x11	4	0
Pool1	96	3x3	2	0
Conv2	256	5x5	1	2
Pool2	256	3x3	2	0
Conv3	384	3x3	1	1
Conv4	384	3x3	1	1
Conv5	384	3x3	1	1
Pool5	384	3x3	2	0
fc9	4096	1x1	1	0
fc10	4096	1x1	1	0
fc11	4096	1x1	1	0
conv6	4096	6x6	1	3
conv7	4096	1x1	1	0
bb-output	256	1x1	1	0
pixel-conv	192	1x1	1	0

TABLE I
THE LAYERS OF DEEPTLR+CONV WITH THEIR CORRESPONDING NUMBER OF CHANNELS, KERNEL SIZE, STRIDE AND PADDING

is consisting of three fully connected layers that combine the ground truth depth values and the extracted features succeeded by a concatenation layer to prepare the input of the third part. Finally The third part guides the CNN into learning the bounding box regression using the extracted features and the depth values yielding the prediction to calculate the loss functions.

B. Training

Some primordial preprocessing routines were performed before the training was started. The dataset instances were cropped, randomly mirorred and warped. A data split was performed aswell (80% Train set, 20% Validation set). During the training process, the model was given a pair of RGB image (640x352) and Depth map (640x352) as ground truth for each instance in the dataset. It was trained using three different datasets (GTA V, KITTI [12], GTA V+KITTI). In doing this, we could investigate the model's ability to transfer its learned concept from a synthetic dataset into a real world dataset. The training was performed using a GeForce GTX 970 (4042 MB of Memory). The Adam solver adapts the learning rate during the training using a step policy $\eta\gamma^{iter/stepsize}$ where η is the base learning rate being equal to 10^{-4}, $\gamma = 0.4$, $stepsize = 10^{-4}$ and $iter$ being the current iteration. The momentum is set to 0.9 and the weight decay factor is set to the value $5 \cdot 10^{-4}$.

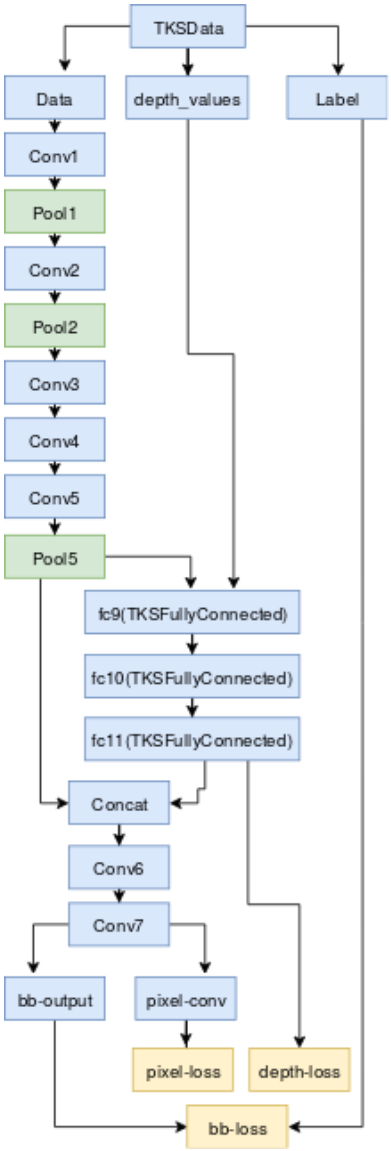


Fig. 8. A simplified graphic representation of DeepTLR+Conv.

The loss function of the depth prediction is an Euclidean loss defined as follows:

$$\frac{1}{2N}\sum_{n=1}^{N}\|(p_n - g_n)\|_2^2$$

where p_n is the predicted depth map vector and g_n the ground truth depth map vector.

The loss function of the bounding box regression is the L_1 loss, while the pixel loss is calculated using a softmax loss layer that computes the multinomial logistic loss of the softmax of its inputs. This guarantees a numerically stable gradient.

The Depth prediction and the bounding box losses are weighted respectively with the scalars 5 and 10. The progress of the loss function (see figure 9) during the training changes slightly from a dataset to another. But generally the model showed very similar behaviors, which proves its consistency and robustness.

Fig. 9. the loss function during the training

Training Dataset	Evaluation Dataset	TP	FP	FN	P%	F1%
KITTI	KITTI	1862	1096	2450	90.48	58.46
GTAV	KITTI	9801	7920	12573	55.31	48.89
GTAV	KITTI+GTAV	1467	516	1737	73.98	56.56
KITTI+GTAV	KITTI	1694	154	2464	91.67	56.41
KITTI+GTAV	KITTI+GTAV	3400	3000	4200	91.89	60.18

TABLE II
EVALUATION RESULTS OF THE MODEL TRAINED AND EVALUATED ON KITTI,GTAV,KITTI+GTAV. THE INPUT SHAPE OF BOTH RGB AND DEPTH MAP IS (640X352)

C. Results

The Model was evaluated as shown in table II. The average inference time on a GeForce GTX 970 (4042 MB of Memory) is $21ms$. Different combinations of Training/Validation datasets were evaluated. The results in table II show clearly that learning only from synthetic data leads to a bad generalization when inferring on unseen real world data. The model generalizes better when being trained on a homogeneous dataset. This leads us to use synthetic data as a supplement to the real world data. KITTI+GTA V data set contains 30.000 KITTI instance and 13.000 GTA V instance. This combination improved the detection of the vehicles in the top left corner of a given frame. KITTI+GTA V improved also the detection of very far vehicles. Another improvement were observed regarding the overlap of the bounding boxes. While the KITTI depth values refined slightly the probability maps in some cases, where two or more vehicles are positioned side by side in the nearest distances, GTA V refined also slightly the probability maps in the furtherest distances (see figures 12, 11). See also 10 for depth prediction results.

V. FCRN-SSD

A. Network Design

The main idea is to use two separate networks (see figure 13). The first part is a Fully Convolutional Residual Net which predicts a depth map from a monocular image (RGB) and adds a fourth channel to the input data, resulting in RGBD images. As the original FCRN network outputs a image roughly half

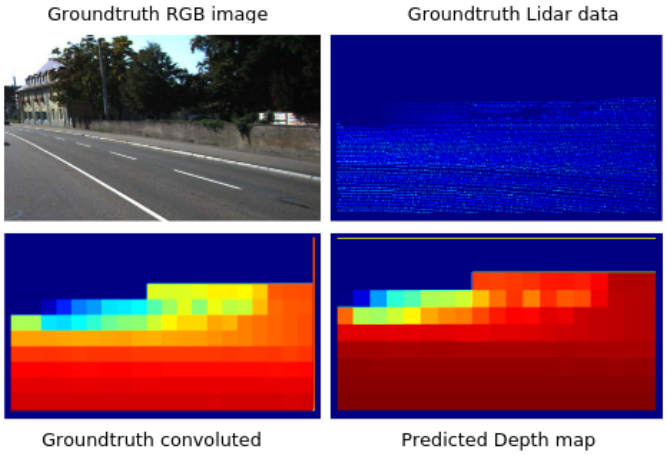


Fig. 10. An examlpe from the evaluation results. The top images are the actual ground truth. the bottom left image is the result of a convolution over the ground truth LIDAR map (with a 32x32 kernel size) which is used as the model's input. The bottom right image is the model's predicted depth map.

Fig. 11. Example of bounding box regressions with probability maps using DeepTLR+conv trained only on GTAV

the size of the input image, the architecture was adjusted to return depth maps of the same size. These RGBD images are used to train a Single Shot Detector, which finally returns two dimensional bounding boxes. The main advantage of this approach is the possibility to deal with two separate and stable networks which can be evaluated individually. This allows to examine the impact of synthetic data usage separately, because no combined objective function for multi-task learning is involved.

Fig. 12. Example of bounding box regressions with probability maps using DeepTLR+conv trained on KITTI+GTAV and compared to a model trained only on KITTI

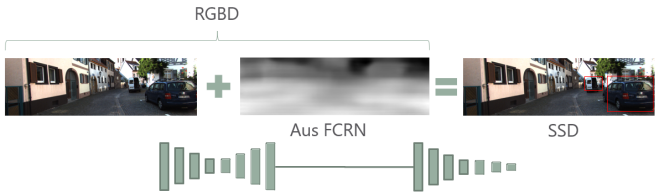


Fig. 13. Main idea, linkage between FCRN and SSD

B. Depth Training

The training of FCRN is done with the Adam optimizer and a learning rate of 0.001. The l2-norm is used as loss function. For the loss function, the black pixels in the ground truth are masked out and aren't contribute to the error. Input images are 300x300 pixels, the output, the depth map, has the same size. Batch size is chosen to be 8. These values are oriented towards the recommendations of [1].

C. Depth Results

In table III metrics are presented for two FCRN models trained on the KITTI respectively GTA V training dataset. Both models are evaluated on both test datasets. The metrics for GTA V are better than for KITTI, which is probably because of the better data quality, but as expected the metrics are far better when evaluated on the same dataset. Still the GTA V data can be useful when looking at figure 14, which proves that the prediction of the GTA V model on KITTI data is far from random, it seems to recognize near objects while the model has never seen KITTI data. Anyway the results for the KITTI model on the KITTI test data seems much more fine-grained as it recognized filigree objects like street lights, see figure 15.

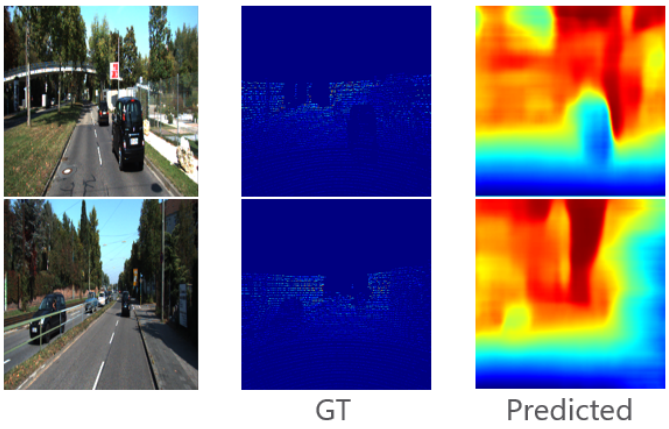


Fig. 14. Qualitative evaluation of depth prediction performance on KITTI data, model trained on GTA

D. Detection Training

The SSD detector was trained and tested on the KITTI Object Detection Evaluation 2012 [12] training dataset, using a 60/40 split for training and testing, respectively.

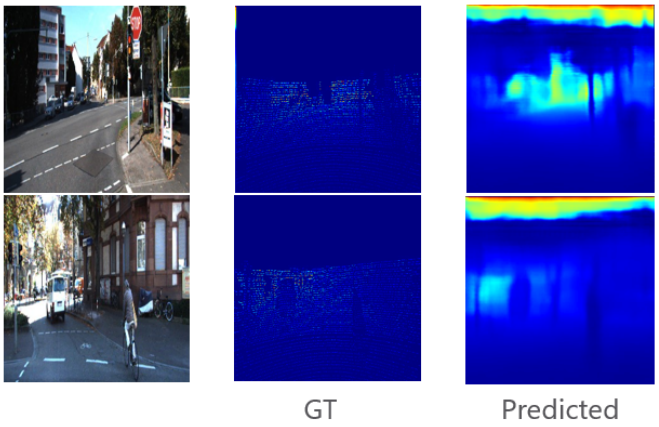


Fig. 15. Qualitative evaluation of depth prediction performance on KITTI data, model trained on KITTI

The training routine was adjusted to cope with 3D and 4D RGBD input images. We used a batch size of 8 images due to memory limitations and a learning rate of 10^{-3}, which had stable convergence. The RMSProp optimizer was used with a weight decay of 5×10^{-4}. As mentioned, the loss function is a mixture of classification and regression and is calculated the following way: Let $x_{ij}^p = \{1, 0\}$ be an indicator for matching the i-th default box to the j-th ground truth box of category p. In the matching strategy above, we can have $\sum_i x_{ij}^p \geq 1$. The overall objective loss function is a weighted sum of the localization loss (loc) and the confidence loss (conf):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

where N is the number of matched default boxes. If $N = 0$, wet set the loss to 0. The localization loss is a Smooth L1 loss between the predicted box (l) and the ground truth box (g) parameters. Offsets are regressed for the center (cx, cy) of the default bounding box (d) and for its width (w) and height (h).

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

The confidence loss is the softmax loss over multiple classes confidences (c).

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0)$$

$$\text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

and the weight term α is set to 1 by cross validation.

E. Detection Evaluation

For evaluation we chose bounding boxes with a classification confidence threshold of at least 0.8 and limited the results to 7 bounding boxes (top-k). Bounding Boxes that had an Intersection over Union (IoU) of at least 70% were considered as valid object proposals. Additionally, non-maximum suppression was performed, filtering all bounding boxes of the same class in the same area with a lower confidence value. The network achieved 25% mAP when trained on RGBD images. Only vehicle and pedestrians related classes were considered for evaluation, other classes were omitted due to their relatively low occurrence frequency within the KITTI Object dataset. For a sample detection result see figure 16.

Fig. 16. Detection result of the SSD network. All cars and trucks are detected with relatively precise bounding boxes. False positives are a problem as seen on the right side and can be reduced with a higher confidence threshold, but not without the drawback of having more false negatives.

We observed that the loss during training with depth data was converging faster. We yet have to perform crossvalidation with different IoU tresholds (mAP@[.5:.95]), simple tests suggested an IoU optimum around 0.7 which was used in our evaluation routines.

VI. FCRN-DSSD

A. Network Design

The 3rd solution was to combine FCRN with SSD as a real-time capable multi-task learning approach in Tensorflow as shown in figure 17. As a base network we used ResNet50 v2 [2] with input dimensions of 384x384x3 pixel in fully convolutional mode. Since the FCRN from [1] is based on ResNet50, we adopted this network also for SSD and changed the dimensions of the feature layers, from which feature maps were used for bounding box prediction and classification. SSD has difficult times in detecting small objects [3], therefore we extended SSD with so called deconvolutional modules (DM) and prediction modules (PM), which is called DSSD (Deconvolutional Single Shot Detector). The implementation of those modules were same as described in [13]. According to [13] this improves recognition of small objects significantly. But the DSSD in [13] was using ResNet101 instead of ResNet50. We decided to leave the basenet with ResNet50, since FCRN was working good with that network. Another difference is, that we extended the count of feature layers. We use one additional feature map size from ResNet50, since the dimensions of feature layers and deconv layers did not match. The network was extended to support any number of feature layers from the basenet, which can be defined as dictionary with each layer as comma separated list, in this case it was 'resnet_v2_50': 'resnet_v2_50/block2/unit_3/bottleneck_v2/conv3, resnet_v2_50/block3/unit_5/bottleneck_v2/conv3'.

The network is in a modular design. For training it is possible to activate or deactivate DSSD or the depth prediction, so that you can just use SSD. The framework is using Tensorflow Slim[3] for building the networks. This makes it easy to extend the network with other basenets. Besides ResNet50, there are VGG-16 [14] and MobileNet [15] already available. Furthermore, SSD can be exchanged easily for another detector if desired.

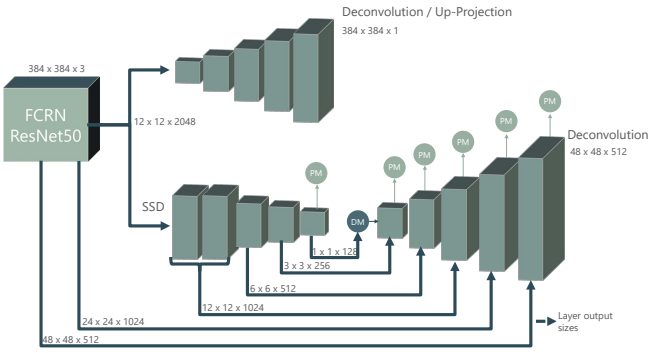


Fig. 17. FCRN with DSSD network as multi-task learning network

B. Multi-task loss function

The total loss function for multi-task learning is a weighted sum of FCRN loss and SSD loss plus the regularization loss:

$$L_{total} = \alpha L_{FCRN} + \beta L_{SSD} + L_{regularization} \quad (1)$$

For FCRN [1] suggests to use a modified version of the Huber loss [16], which is called berHu loss and defined as:

$$Threshold\ c = \frac{1}{5} \max_i(|\tilde{y}_i - y_i|) \quad (2)$$

c is calculated for every pixel in the current batch. The maximum error between predicted depth and ground truth depth is taken from the batch and weighted with 20%. Depending on the absolute error x, L_1 or L_2 norm is chosen based on the threshold:

$$B(x) = \begin{cases} |x| & |x| \leq c \\ \frac{x^2+c^2}{2c} & |x| > c \end{cases} \quad (3)$$

We do not consider invalid pixels from the ground truth depth map like sky. They will be masked out during loss calculation.

For SSD smoothed L_1 loss for localization and softmax loss over multiple classes (cross entropy) for confidence is used as defined in [3].

[3]https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/slim

C. Preprocessing Pipeline

The preprocessing pipeline was taken from Google's Inception v3[4] and uses only Tensorflow functionality. The images are augmented online when training with bounding box adjustments, random horizontal flips, random color distortions and patch sampling shown in figure 18. From the original image with size 1280x720 pixels a random ground truth bounding box will be selected (first image) and enlarged with some constrains (second image), for example that 60% of all ground truth bounding boxes have to be in that enlarged box or that it has to have a certain aspect ratio etc. The second image shows the randomly selected ground truth bounding box in red and the enlarged box in blue. The blue box will be cut out (third image) and resized to the network input dimension of 384x384 pixels. The same transformation will be applied to the ground truth depth map, too.

Before using this pipeline we did convolution on KITTIs LIDAR data, so that holes and empty areas were closed and saved the images to disk.

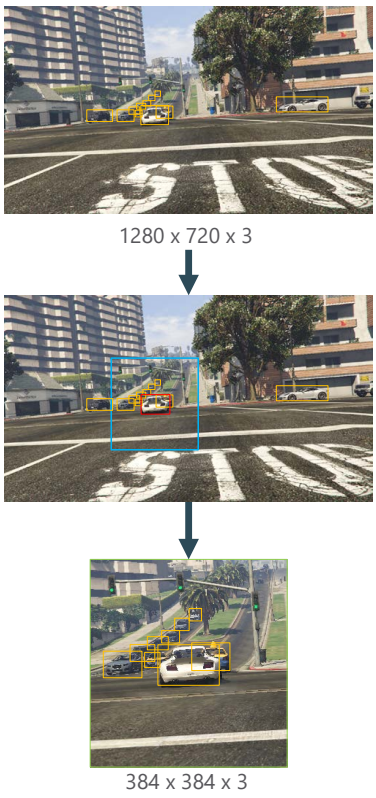


Fig. 18. Example on how patch sampling is working

D. Results

For training we used 15.000 GTA V and KITTI RGB images with depth maps. We trained the model with different constant learning rates and loss weights. As optimizer we chose Adam, 0.9 momentum and a weight decay of 0.0005 with batch size 32.

[4]https://github.com/tensorflow/models/blob/master/research/inception/inception/image_processing.py

Figure 19 is showing the qualitative results of the trained networks. The upper row shows the network trained with KITTI dataset and inferenced also on KITTI, with the RGB image, the ground truth (GT) and predicted depth map. One can see that prediction for the ground is pretty similar as in ground truth and also the orange blob for the car is visible. The erroneous predictions for the sky are due to ignored invalid pixels when calculating the losses and should be ignored for now.

The lower row is showing the results of the network trained on GTA V and evaluated on GTA V testset. We cannot clearly see if the depth prediction is similar to the ground truth. The street and the angle seem to be predicted but details like the stop sign and the car in the distance is not visible at all.

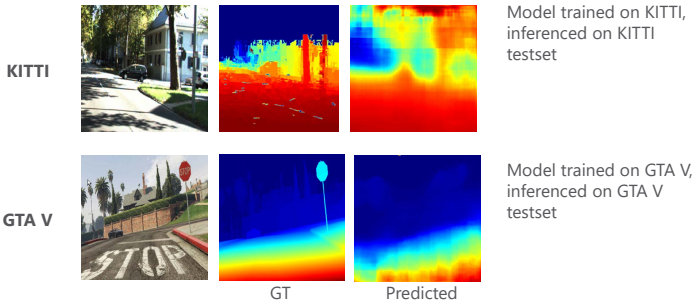


Fig. 19. Qualitative evaluation of depth prediction performance both on KITTI and GTA V dataset

In figure 20 we see the same upper row as in figure 19 but the lower row now shows the model trained on GTA V and evaluated on KITTI testset. Compared to the model trained on KITTI, it performs very poor on KITTI testset. This is due to the fact that the depth prediction performs poorly in this network.

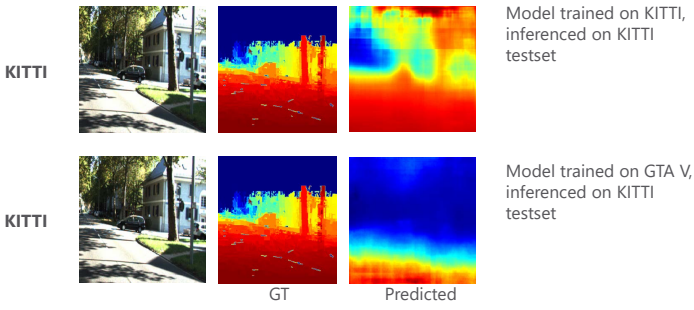


Fig. 20. Qualitative evaluation of depth prediction performance with GTA V trained network applied on KITTI dataset

Table IV is showing some error metrics for depth prediction (REL, RMSE, LOG10, d1, d2, d3) and 2D bounding box regression (mAP) for models trained on KITTI and on GTA V. Both models are evaluated on KITTI and GTA V testset. Depth prediction in model trained and evaluated on KITTI performs much worse than in model trained and evaluated on GTA V. This is due to 'perfect' depth map from GTA V. The depth map from KITTI is very sparse in contrast and therefore inaccurate. But model with KITTI reaches 37% mAP for vehicle detection, whereas model with GTA V only reaches 13% mAP. This is due to SSDs inability to deal with many small objects and other reasons mentioned in section III w.r.t. ground truth bounding boxes.

When evaluating model trained with KITTI on GTA V testset, the depth prediction performs better than evaluating same model on KITTI testset. But looking at mAP, we see that performance drops over 80%. Detecting vehicles seems not to influence depth prediction. In other words, we assume that features learned for depth prediction are not used for vehicle detection.

Looking at model trained on GTA V and evaluated on KITTI testset shows a similar result. Depth prediction performs slightly better than same model evaluated on GTA V testset. Ironically vehicle detection learned on GTA V and applied on KITTI performs better with about 30% higher mAP. This result needs to be investigated further.

In total, the network performs poorly at the current state. Training for single model took about 6 days, due to big network size. Inference speed is at 62,5 ms (∼16 FPS), which is considered as real-time. One major problem is the appropriate weighting of the losses, which influences the results greatly. We chose 4:1 or 8:1, but also 1:2 weighting between DSSD and FCRN. Another issue was SSD itself. There are lots of model parameters to tune and try, which consumes more time to find the right default box sizes and ratios for the datatsets.

Even if the numbers are poor, we can see that depth prediction and vehicle detection can profit from synthetic data like GTA V. Due to technical problems and short time we could not evaluate how the network performs when training on both datasets at the same time.

VII. Conclusion

In our evaluation we were able to observe that by using synthetic GTAV V data, depth prediction works well due to perfect training data. DeepTLR and the FCRN-SSD approaches especially profit from it. FCRN-DSSD yet has problems with its multi-task learning objective and lacks good performance. Bounding box prediction with GTA V data is difficult and requires careful parameter tuning for all approaches.

Increasing GTA V images number to ∼200.000 images [7], should give better results since variation of images is very low. Further fine tuning of images can be done by adjusting color ranges of GTA V data to match KITTI or a more natural color distribution, but first tests did not give significant better results. Also improved in-game traffic flow control would help to enhance inter-scene and bounding box size variation. To increase variance in images, in-game weather control should be exploited. Another improvement would be to use stencil maps to create ground truth bounding boxes, since in-game occlusion checks are insufficient and therefore often times bounding boxes are drawn even if the vehicle is not visible.

FCRN depth estimation results could most probably be improved by using the berHu loss instead of L2 norm, as the original paper suggests.

The SSD networks loss has problems to converge, especially

for positive bounding box examples and localizations. Also extending it with deconvolution layers should improve detection on small objects. Better default box sizes, ratios and tiling adjusted for GTA V training data are needed.

Loss weights with a ratio of 4:1 or 8:1 (DSSD:FCRN) where chosen for multi-task training on GTA V or 1:2 when training on KITTI. FCRN-DSSD would profit from adaptive loss balancing, where weights for loss functions are learned by the network [17] or gradients are normalized [18]. Since the current FCRN-DSSD pipeline is using Tensorflow-only functions, it is inflexible. To improve GTA V dataset quality, offline preprocessing to handle ground truth bounding boxes out of range still has to be done and more sophisticated image augmentation libraries[5]. In addition, if the learning rate is too high the SSD loss oscillates while the depth loss converges quickly. If the learning rate is too low, the depth loss does not converge while the SSD loss oscillates less. Different learning rates should be used for different task, since some tasks are easier to learn as suggested in [19]. Finally, FCRN-DSSD is in modular architecture, it's easy to change certain parts or networks. Using a different detector may give a hint if the performance problem is rooted in the multi-task approach or not.

REFERENCES

[1] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," *CoRR*, vol. abs/1606.00373, 2016. [Online]. Available: http://arxiv.org/abs/1606.00373

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[5] A. Lesi, "Learning from depth images for monocular object detection with convolutional neural networks," Master's thesis, Karlsruhe Institute of Technology, 2016.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[7] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" *CoRR*, vol. abs/1610.01983, 2016. [Online]. Available: http://arxiv.org/abs/1610.01983

[8] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," *CoRR*, vol. abs/1608.02192, 2016. [Online]. Available: http://arxiv.org/abs/1608.02192

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

[10] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *International Conference on 3D Vision (3DV)*, 2017.

[11] M. Weber, P. Wolf, and J. M. Zllner, "Deeptlr: A single deep convolutional network for detection and classification of traffic lights," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, June 2016, pp. 342–348.

[12] "Kitti object detection evaluation 2012," http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark, accessed: 2018-15-04.

[13] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional single shot detector," *CoRR*, vol. abs/1701.06659, 2017. [Online]. Available: http://arxiv.org/abs/1701.06659

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[16] P. J. Huber, *Robust Estimation of a Location Parameter*. New York, NY: Springer New York, 1992, pp. 492–518. [Online]. Available: https://doi.org/10.1007/978-1-4612-4380-9_35

[17] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," *CoRR*, vol. abs/1705.07115, 2017. [Online]. Available: http://arxiv.org/abs/1705.07115

[18] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," 2018. [Online]. Available: https://openreview.net/forum?id=H1bM1fZCW

[19] T. van Erven and W. M. Koolen, "Metagrad: Multiple learning rates in online learning," *CoRR*, vol. abs/1604.08740, 2016. [Online]. Available: http://arxiv.org/abs/1604.08740

[5] https://github.com/aleju/imgaug

APPENDIX A

TABLE III
METRICS FOR EVALUATION OF FCRN AS PART OF FCRN-SSD

	Trained on KITTI		**Trained on GTA V**	
	Evaluated on KITTI testset	Evaluated on GTA V testset	Evaluated on KITTI testset	
REL	**-**	**-**	**-**	**-**
RMSE	**13.78**	**75.86**	**15.60**	**156.47**
LOG10	**2.02**	**2.03**	**0.48**	**4.27**
d1	**0.04**	**0.01**	**0.65**	**0.002**
d2	**0.09**	**0.02**	**0.80**	**0.004**
d3	**0.17**	**0.03**	**0.88**	**0.007**

TABLE IV
METRICS FOR EVALUATION OF FCRN-DSSD; MISSING VALUES NOT COMPUTED DUE TO NUMERICAL ERROR (INF, NAN)

	Trained on KITTI		**Trained on GTA V**	
	Evaluated on KITTI testset	Evaluated on GTA V testset	Evaluated on KITTI testset	
REL	**199.94**	**188.51**	**65.82**	**62.57**
RMSE	**205.92**	**195.99**	**92.71**	**81.22**
LOG10	**-**	**-**	**5.32**	**-**
d1	**-**	**-**	**-**	**-**
d2	**-**	**-**	**-**	**-**
d3	**-**	**-**	**-**	**-**
mAP	**0.37**	**0.07**	**0.13**	**0.17**