Optimizing a neural network for detection of moving vehicles in video

Noëlle M. Fischer, Maarten C. Kruithof, Henri Bouma [1]

TNO, Oude Waalsdorperweg 63, 2597 AK The Hague, The Netherlands

ABSTRACT

In the field of security and defense, it is extremely important to reliably detect moving objects, such as cars, ships, drones and missiles. Detection and analysis of moving objects in cameras near borders could be helpful to reduce illicit trading, drug trafficking, irregular border crossing, trafficking in human beings and smuggling. Many recent benchmarks have shown that convolutional neural networks are performing well in the detection of objects in images. Most deep-learning research effort focuses on classification or detection on single images. However, the detection of dynamic changes (e.g., moving objects, actions and events) in streaming video is extremely relevant for surveillance and forensic applications. In this paper, we combine an end-to-end feedforward neural network for static detection with a recurrent Long Short-Term Memory (LSTM) network for multi-frame analysis. We present a practical guide with special attention to the selection of the optimizer and batch size. The end-to-end network is able to localize and recognize the vehicles in video from traffic cameras. We show an efficient way to collect relevant in-domain data for training with minimal manual labor. Our results show that the combination with LSTM improves performance for the detection of moving vehicles.

Keywords: Concept detection, recurrent network, LSTM, YOLO, moving object detection.

1. INTRODUCTION

In the field of security and defense, it is extremely important to reliably detect moving objects, such as cars, ships, drones and missiles. Detection and analysis of moving objects in cameras near borders could be helpful to reduce illicit trading, drug trafficking, irregular border crossing, trafficking in human beings and smuggling.

Object detection is one of the most relevant problems in computer vision. Detectors typically first extract robust hand-crafted features from images (SIFT [40], Haar [42], HOG [14]) and then apply classifiers in a sliding-window based approach (e.g., Viola-Jones [54] or deformable parts models (DPM) [20]). An alternative to the hand-crafted features is a data-driven approach based on deep convolutional neural networks (CNN) [37]. Recent benchmarks for object detection and object classification (e.g., Pascal VOC [19], ImageNet [15] and MS-COCO [34]) show the success of CNNs (e.g., GoogLeNet [52], AlexNet [30], Caffe [29], VGG [51] and R-CNN [22]).

Object detection methods have moved from scanning window approaches [20] to methods based on region proposals e.g., by using objectness [1], multi-scale combinatorial grouping [45], selective search [53] or EdgeBox [57]. These region proposals have the advantage that they reduce the search space [23]. An overview of region-proposal methods can be found in the paper of Hosang et al. [26][27]. For object detection, one of the most widely used approaches is R-CNN [22], or one of its faster variants where convolutions are shared over region proposals, such as Spatial Pyramid Pooling (SPPnet) [24] and Fast R-CNN [23]. These methods also have the two distinct phases of region proposals and CNN classification.

Motivated by the fact that hand-engineered features were replaced by deep neural networks for image classification, it was expected that the same trend holds for region-proposal generation. Fully CNN-based methods [36] are methods that directly perform object detection by the CNN, without a separate hand-crafted phase for proposal generation. This allows an end-to-end optimization of the system. Early attempts to take the proposal stage out of the loop led to performance degradations [41]. Examples are OverFeat [50], MultiBox [18], R-CNN minus R [33] and YOLO [46]. Different from prior work, they predict bounding boxes and class probabilities directly from full images in one evaluation. This direction has led to an new revolution in object detection and recently many novel methods have been proposed, including DenseBox [28]. Proposal-

[1] henri.bouma@tno.nl; phone +31 888 66 4054; http://www.tno.nl

Free Network [38], single-shot detection (SSD) [39], MSC-MultiBox [52], G-CNN [41], DecompNet [43], DeepBox [32], DeepMask [44] and region proposal network (RPN) for Faster R-CNN [48]. Some of these approaches perform detection in one phase [39][46] and others still have a separate phase to propose bounding boxes with a neural network [48].

Most of them have not yet been applied to car detection, except DenseBox [28] and DecompNet [43], which were both tested on the KITTI dataset [21]. And most deep-learning research effort focuses on classification or detection on single images. However, the detection of dynamic changes (e.g., moving objects, actions and events) in streaming video is extremely relevant for surveillance and forensic applications.

In this paper, we combine the YOLO end-to-end feedforward neural network for static detection with a recurrent Long Short-Term Memory (LSTM) network for multi-frame analysis. We present a practical guide with special attention to the selection of the optimizer and batch size. The end-to-end network is able to localize and recognize the vehicles in video from traffic cameras. We show an efficient way to collect relevant in-domain data for training with minimal manual labor. Our results show that the combination with LSTM improves performance for the detection of moving vehicles.

The outline of this paper is as follows. Section 2 gives an overview of related work. Section 3 presents the method. Section 4 describes the experimental setup and it the results. Finally, Section 5 summarizes the conclusions.

2. RELATED WORK

OverFeat [50] was one of the first CNNs to perform classification and localization together. However, it disjoints classification and localization in training, and it optimizes for localization, not detection performance. The localizer only sees local information when making a prediction. Therefore, OverFeat cannot reason about global context and thus requires significant post-processing to produce coherent detections [28][46].

The MultiBox [18] trains CNNs to generate proposals instead of selective search. The MultiBox method generates 800 non-translation-invariant anchors [28][48]. The MultiBox approach produces bounding-box coordinates, which are independent of the number of classes [52].

'R-CNN minus R' [33] challenges the role of object proposal algorithms, such as Selective Search (SS), in CNN-based object detection systems. The proposals were replaced by a fixed set of static bounding-boxes. This approach is much faster than R-CNN, but it is not real-time (it takes 160 ms per image excl. SS) and it takes a significant accuracy hit from not having good proposals [46]. A bounding-box set with a distribution close to the original proposed objects are selected using a clustering technique. However, for achieving comparable results, even more boxes need to be used compared to R-CNN [41].

YOLO [46] source code is available at github under the name 'darknet'. YOLO has a single convolutional network to predict bounding boxes and class probabilities in an end-to-end network. It is a complete detection system for multiple classes in a single shot. YOLO is very fast at test time (45 fps and 155 fps for Fast-YOLO) since it only requires a single network evaluation. The approach takes a 448x448 image as input and outputs coarse 7x7 grid cells. Each cell can only produce one or two boxes and only one class. Therefore, it is unable to detect small objects or highly overlapped objects. It is also unclear to which extent these results can translate to good performance on data sets with significantly more objects, such as the ILSVRC detection challenge. This fast approach of YOLO does not obtain the best detection accuracy [41]. More recently, an improved version was proposed (YOLO9000 or YOLOv2) [47].

DenseBox [28] is a multi-task jointly learned end-to-end detection network. DenseBox does not require predefined anchors. It is trained on one scale with jitter augmentation, which means that the method needs to evaluate features at multiple scales. DenseBox will densely generate one bounding box with score at every 4 pixels. The bounding boxes are translation-invariant like Region Proposal Network (RPN) [48]. DenseBox uses up-sampling layers to keep a relative high-resolution output, with a down-sampling scale factor of 4 in the model. This enables the network capable to detect very small objects and highly overlapped objects.

Proposal-Free Network [38] uses source code that is based on DeepLab and Caffe. The pixel-wise instance locations and the instance number of each category are simultaneously optimized in one network. Finally, the fine-grained segmentation mask of each instance can be produced with PFN instead of the coarse outputs depicted by the bounding boxes from YOLO. PFN can process one 300x500 image in about one second.

Single-shot detection (SSD) [39] has its code available on github by WeiLiu89 under an 'ssd' version of 'caffe' and more recently also as 'ssd_keras'. SSD is based on a single neural network that is capable of directly generating object bounding boxes and their confidences for a large number of object categories. In SSD, the offset adjustment and confidences for multiple categories of each prior are predicted from the underlying 1x1 feature at each location on a feature map, as opposed to the whole feature map as done in MultiBox [18] and YOLO [46]. This results in a more compact network, which is crucial for efficient detection of a large number of object categories. Additionally, this adds translational invariance in the model output layers, and reduces overfitting and improves detection performance [39]. The SSD priors are of different aspect ratios and scales, densely spread out on all locations from a feature map, and thus can cover various object shapes as opposed to YOLO [46] or OverFeat [50], which use a more rigid grid-based tiling. SSD associates bounding-box priors with features maps of different spatial resolution in the network. This naturally handles objects of different scales and improves detection accuracy with negligible computation overhead, as opposed to OverFeat [50] and SPPnet [24], which requires resizing images to different resolutions and individual processing. The overall SSD design is efficient and provides a unified framework for both training and inference, even for hundreds of object categories. In contrast to Faster-RCNN, SSD directly learns to predict both the offsets over the priors and the confidences for multiple categories, instead of treating all categories as class-agnostic object. Thus, the SSD approach avoids the complication of merging RPN with Fast R-CNN and is much easier to train and straightforward to integrate in other tasks. The SSD approach is more flexible than OverFeat or YOLO because SSD can impose priors of different aspect ratios and scales on each feature location from multiple feature maps. For one prior per location, it is similar to OverFeat and if the topmost feature map is used for predictions instead of convolutional priors it is similar to YOLO [39]. YOLO [46], removes the region proposal step and directly predicts multiple objects and their locations with a single CNN evaluation. However that method is restricted to a predefined number of outputs (e.g. 49 in a 7x7 grid) even when there are fewer or more objects present, and its performance falls short compared to previous region-based detectors [43].

MSC-MultiBox [52] has its source code available at github under the names 'Multibox', 'Inception' and 'Tensorflow'. MSC-MultiBox increases recall of object locations by increasing the number of potential proposals with a fixed budget of evaluated proposals. MultiBox and SPP-net [24] show a comparable efficiency improvement and they are complementary in the sense that spatial pyramid pooling can be added to the underlying ConvNet, and post-classification of proposals can be sped up in the same way with no change to the MultiBox objective [52]. The RPN [48] is quite similar to the MSC-Multibox approach [52]. They both use priors (called "anchors" in Fast R-CNN) that are designed to be translation-invariant and that are predicted from the top layer feature map. The MultiScale priors are different in that multiple tapering layers are used, while the Fast R-CNN approach is predicting boxes of many scales from a single feature map. The two-stage setup of MSc-MultiBox scales to a higher number of classes: the Faster R-CNN work uses many thousands of priors and scalings that increasing towards thousands of classes is not obvious [52]. MSc-MultiBox can also perform single object detection by replacing the confidence prediction with a single-class prediction. However, MSc-MultiBox cannot perform general object detection and is still just a piece in a larger detection pipeline, requiring further image patch classification. MultiBox uses a convolutional network to predict bounding boxes in an image, but it is not a complete detection system [46]. MultiBox results in a somewhat complex setup, requiring the training of two neural networks with a dependency between them [39].

G-CNN [41] trains a CNN to move and scale a fixed grid of bounding boxes towards actual objects. The backbone of the network architecture of this regressor can be any CNN network (e.g. AlexNet [30], VGG [51], etc). 180 boxes is enough for G-CNN to perform better than Fast R-CNN, which uses around 2K selective-search boxes. G-CNN has a higher mAP than YOLO. The authors claim that their result is the best-reported result among methods without an object-proposal stage. G-CNN runs at 3 fps, which is 5 times faster than "Fast R-CNN" and it achieves comparable results to state-of-the-art detectors.

The aim of DecompNet [43] is to produce a network that generates the correct number of object instances and their bounding boxes (or segmentation masks) given an image, using only a single network evaluation without any pre- and post-processing steps, such as region proposals and non-maximal suppression. Building these together into a single framework enables much more straightforward end-to-end training of multiple-class multiple-object detectors, potentially allowing easier and wider application, as well as better performance. PFN [38] is related to the work of DecompNet because it classifies category first, and then divides it into multiple instances. It regresses the number of instance and instance location maps per category. With this information, they applied spectral clustering to separate multiple instances as a post-processing step. In contrast, DecompNet implicitly predicts the number of instances and learns how to cluster response maps, and it is end-to-end trainable. Furthermore, PFN is only applicable when there is a segmentation label while DecompNet will also work when label is not available [43]. The authors admit that DecompNet does not achieve very good results for large number of objects in an image. In KITTI dataset, some images have more than 10-20 cars in an image. In this case their network performs poorly.

The DeepBox [32] implementation runs in 260 ms per image, which is comparable to EdgeBoxes (250 ms). The Fast R-CNN detection system [23], using 500 DeepBox proposals per image, is 4.5 points better than the same object detector using 500 EdgeBox proposals.

DeepMask [44] generates class-agnositc segmentation masks instead of the less informative bounding boxes that are generated by MultiBox, DeepBox and Faster RCNN. DeepMask is able to generate the proposals and rank them in one shot from the test image, directly from the pixel space, in contrast to e.g., DeepBox that reranks proposals from EdgeBox. Detection efficiency can be improved by unifying the detection and classification models, reusing as much computation as possible and in the process abandoning the idea of data-independent region proposals. DeepMask [44] is a convolutional neural network model with two branches: one that can generate class-agnostic segmentation masks, and a second branch predicting the likelihood of a given patch being centered on an object [52]. Inference is efficient since the model is applied convolutionally on an image and one can get the class scores and segmentation masks using a single model. Inference takes 1.2 to 1.6 seconds per image. The performance of DeepMask is compared to other region-proposal methods, such as EdgeBoxes and Selective Search. Fast R-CNN performs better with 100 DeepMask proposals than with 2000 Selective-Search proposals.

Region proposal network (RPN) for Faster R-CNN (or: Faster-RCNN) [48] has its source code available at github under the name 'faster_rcnn'. Faster-RCNN still use region proposals to find objects in an image. Unlike its former variants, the region proposals in Faster-RCNN are produced by RPN's, sharing convolutional feature computation with classifiers in the second stage. However, the RPN needs predefined anchors [28]. RPN is trained on multi-scale objects. The RPN output translation-invariant boundingboxes like DenseBox [28]. The priors of MSC-Multibox are different from Faster-RCNN in that multiple tapering layers are used, while the Faster-RCNN approach is predicting boxes of many scales from a single feature map. The two-stage setup of MSc-MultiBox scales to a higher number of classes: the Faster-RCNN work uses many thousands of priors and scaling that approach to thousands of classes is not obvious [52]. The most accurate Faster-RCNN achieves 7 fps while a smaller, less accurate one runs at 18 fps [22][23]. Faster-RCNN replaces selective search proposals by ones learned from a region proposal network (RPN), and introduces a method to integrate the RPN with Fast R-CNN by alternating between fine-tuning shared convolutional layers and prediction layers for these two networks [39]. Some work, like Faster-RCNN [48] use a deep CNN to generate a relatively small number of high-quality candidates, but still require several hundreds of box evaluations [43].

3. METHOD

3.1 Network structure

We used the TinyYOLO model to be able to run approximately in real-time (we achieved 21 frames per second) in the Keras library with a Theano backend. YOLO is an end-to-end network that can learn and detect multiple classes in a single pass, which makes it very efficient. The modification we made can be seen in Figure 1. We added a Time Distributed layer over the convolutional layers with a length of N and combined these layers with an LSTM layer of size 4096 before the last connected layer. TinyYOLO performs the analysis in N separate frames and the LSTM combines the information from these frames.

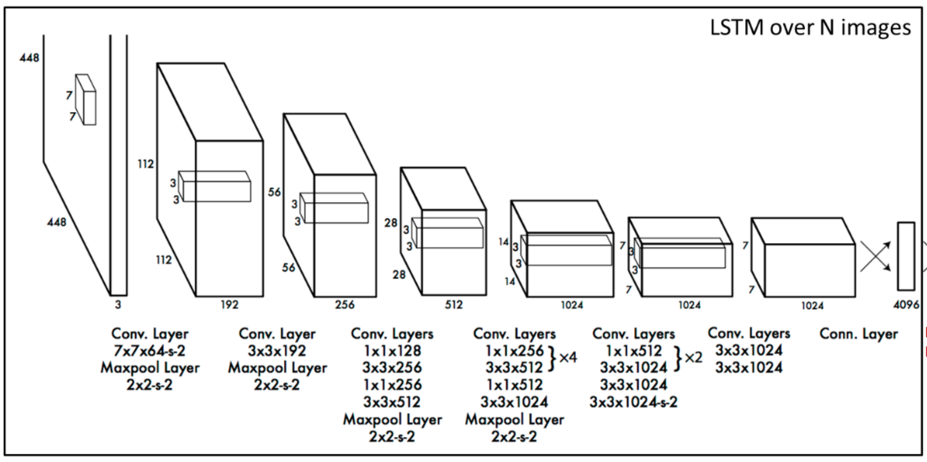

Figure 1: Overview of the YOLO model used in our experiments (based on figure from [46]). The location of the LSTM layer is added to the figure and shown in red.

3.2 Optimizer, learning rate and batch size

We found the Stochastic Gradient Descend (SGD) optimizer to be quite unstable so we designed an experiment to determine the optimal learning rate and optimizer combination. We chose 6 different combinations as shown in Figure 2. We trained a small subset of 500 images for 200 epochs with a batch size of 32 and noted the train loss at 10, 50, 100 and 200 epochs. From Figure 2 three combinations (of optimizer and learning rate) obtained the smallest loss after 200 epochs. The RMSProp with a learning rate of 10^{-4} showed some instabilities and was disregarded. From the two remaining we chose the RMSProp with a learning rate of 10^{-5} since it showed a slightly faster loss decay. We chose a batch size of 8 for the next experiments which gave us a good balance between final loss and speed.

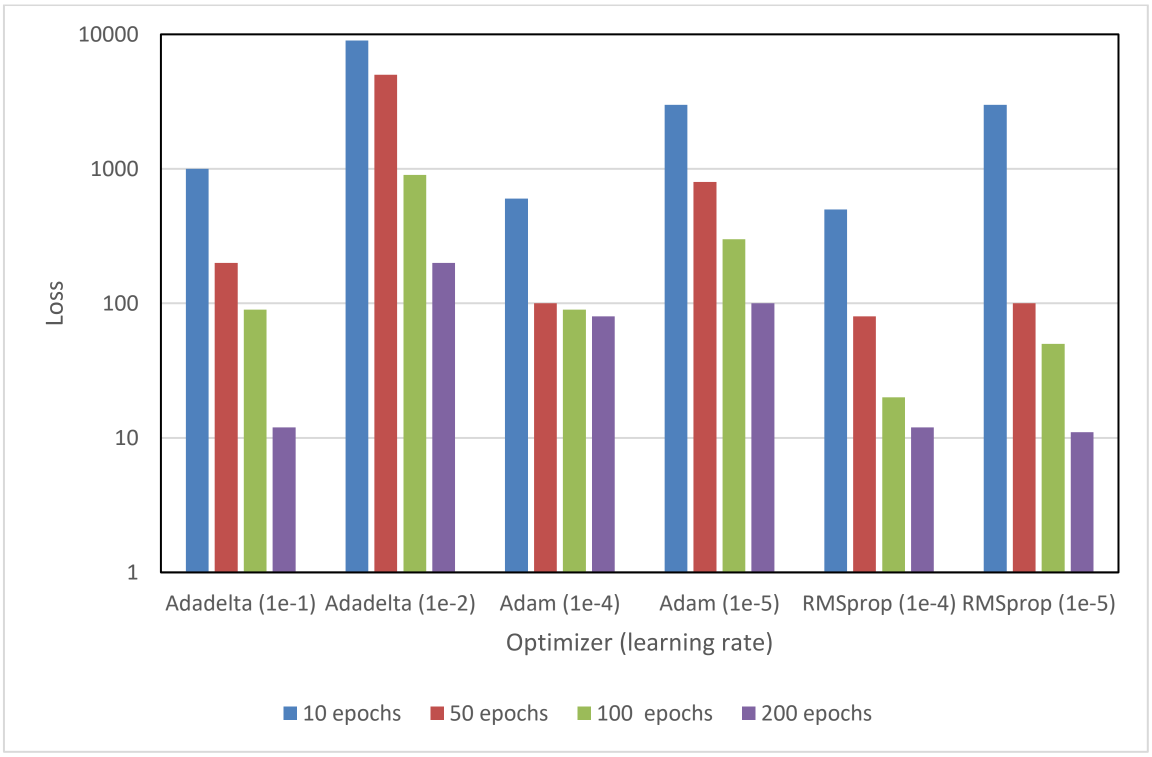

Figure 2: Loss after 10, 50, 100 and 200 epochs for different optimizers and learning rates.

4. EXPERIMENTS AND RESULTS

4.1 Dataset

The dataset consists of one week of video data from 43 cameras on a road. For an example of how the images from the cameras look, see Figure 3. The ground-truth (GT) detections are generated in two steps. First a Viola-Jones detector (VJ-detector) is applied and then the detections are filtered with a tracker (VJ-tracker) to remove false positives and false negatives. The VJ detector was trained with 5000 car images of 12x12 pixels and 10000 background images of 12x12 pixels taken from random pictures. We checked manually to observe the quality of the GT data. We observed that there are hardly false positives in GT data after tracking. There are several false negatives (misses) in GT data, which are very near, very far, or they lack contrast.

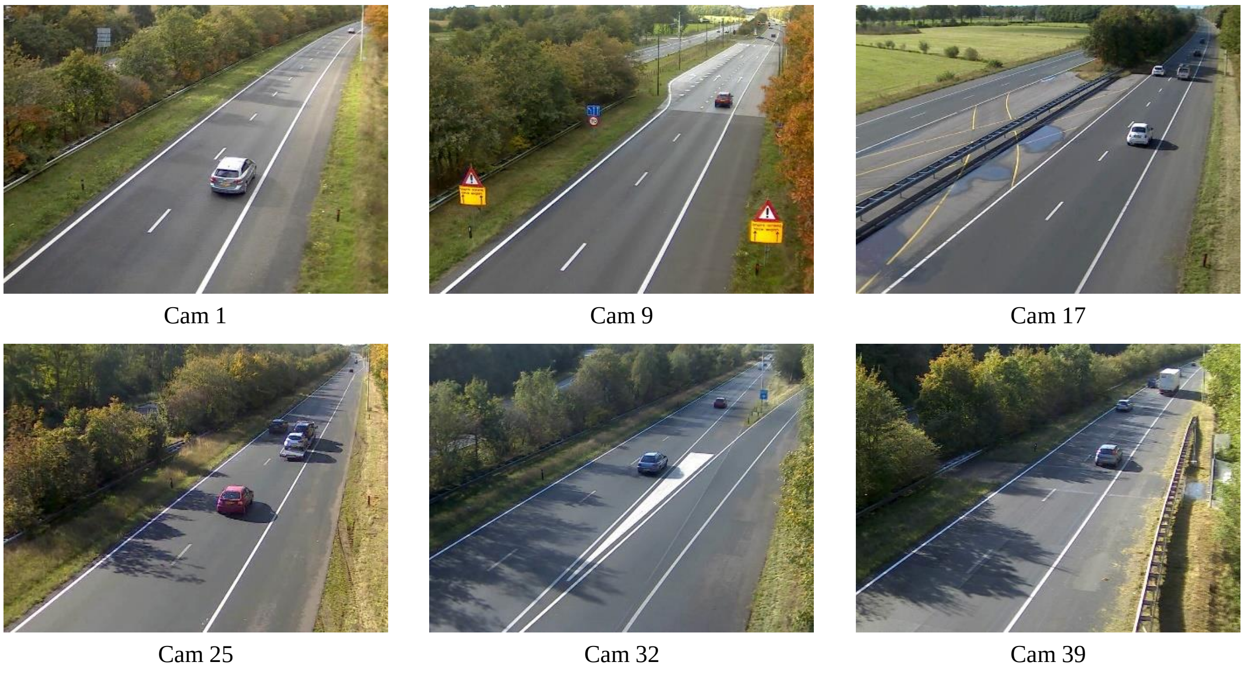

Figure 3: Example images from the dataset.

4.2 Sampling strategy

Sampling training samples can be performed in two ways. One way is to sample random frames from the dataset and use all detections in those frames. However, due to the perspective effect, there are only few 'large' detections nearby and many 'small' detections further away. Another way to sample first selects a random set of frames, then stores all detections in bins related to their size, and finally takes an equal amount of random samples from each bin. In this way, the training set is more balanced. The first strategy is called 'random sampling' and the second is 'balanced sampling'. For the experiments we chose 100 equidistant bins in the y direction of the image and put each detection in the bin containing the y value of the bottom of the detection.

4.3 Experimental setup

For the training, we trained with 20,000*N or 100,000*N frames with detections generated with VJ-tracker (multiple cameras, multiple times of the day). We used N=3 for three consecutive frames (framerate=25 FPS). We either chose a balanced training set in detection sizes or distance to the camera or a non-balanced random set. We trained with an RMSProp optimizer with a learning rate of 10^{-5} for 200 epochs. For the testing, we tested on a fixed test with 500*N frames with detections generated with VJ-tracker. These frames were clearly separated from train set (different time and/or different camera). The test set was always balanced in detection sizes.

4.4 Performance measure

To test the network accuracy we calculated the Mean average precision (MAP) score for the resulting detections and the VJ detections. We used 11 minimal recall values (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1) to calculate the MAP scores. We calculated the MAP scores for 5 different Intersection over Union (IOU) thresholds (0.5, 0.6, 0.7, 0.8 and 0.9). The IOU is a common way to determine the amount of overlap. We noted the MAP scores after 10 epochs and after 100 epochs for the 20k frames and 30 epochs for the 100k frames. We report results after 10 and 100 epochs, to give an assessment of the performance with limited training time.

4.5 Results

The MAP scores are shown in Table 1. The table summarizes the sampling approach (random or balanced), the type of network (dense or LSTM), the number of training samples used (each sample contains N=3 frames), and the amount of epochs. The MAP scores are reported for different IOU thresholds. The results show the following:

- The LSTM performs better than the dense network. For an IOU threshold of 0.5 and 10 epochs, we see an improvement from 54% (for dense) to 62% (for LSTM) and for 100 epochs we see an improvement from 63 to 73%.
- The balanced sampling does not improve the results in comparison to the random sampling. In 80% of the comparisons that can be made in the table, the difference is less than 1%.
- Furthermore, the table shows, as expected, that more training data leads to better results. For a balanced LSTM with 10 epochs, we see an improvement from 62% (for 20 000 samples) to 80% (for 100 000 samples).
- Another expected result is that it is beneficial to train more than 10 epochs for the IOU thresholds of 0.5 and higher. For the balanced LSTM with 20 000 samples, we see an improvement from 62% (for 10 epochs) to 73% (for 100 epochs).

The discretization effect in the results is caused by the 11 minimal recall values that we used to compute the MAP scores.

Example detectsions are shown in Figure 4. The images show that our method sometimes misses a GT detections (bottom left) but sometimes also finds a car that the VJ detector did not find (top right and bottom right).

Table 1: Mean average precision (%) of methods

Methods				**IOU threshold**				
Sampling	**Network**	**#samples**	**Epochs**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Random	Dense	20 000	10	54	36	18	09	9
Balanced	Dense	20 000	10	53	36	27	09	9
Random	LSTM	20 000	10	**62**	**53**	**36**	**18**	9
Balanced	LSTM	20 000	10	**62**	**53**	**36**	**18**	9
Random	Dense	20 000	100	63	45	36	18	9
Balanced	Dense	20 000	100	63	45	27	18	9
Random	LSTM	20 000	100	**72**	**64**	**54**	**36**	9
Balanced	LSTM	20 000	100	**73**	**64**	45	27	9
Balanced	LSTM	100 000	10	**80**	63	54	36	9
Balanced	LSTM	100 000	30	**81**	**72**	**63**	**45**	18

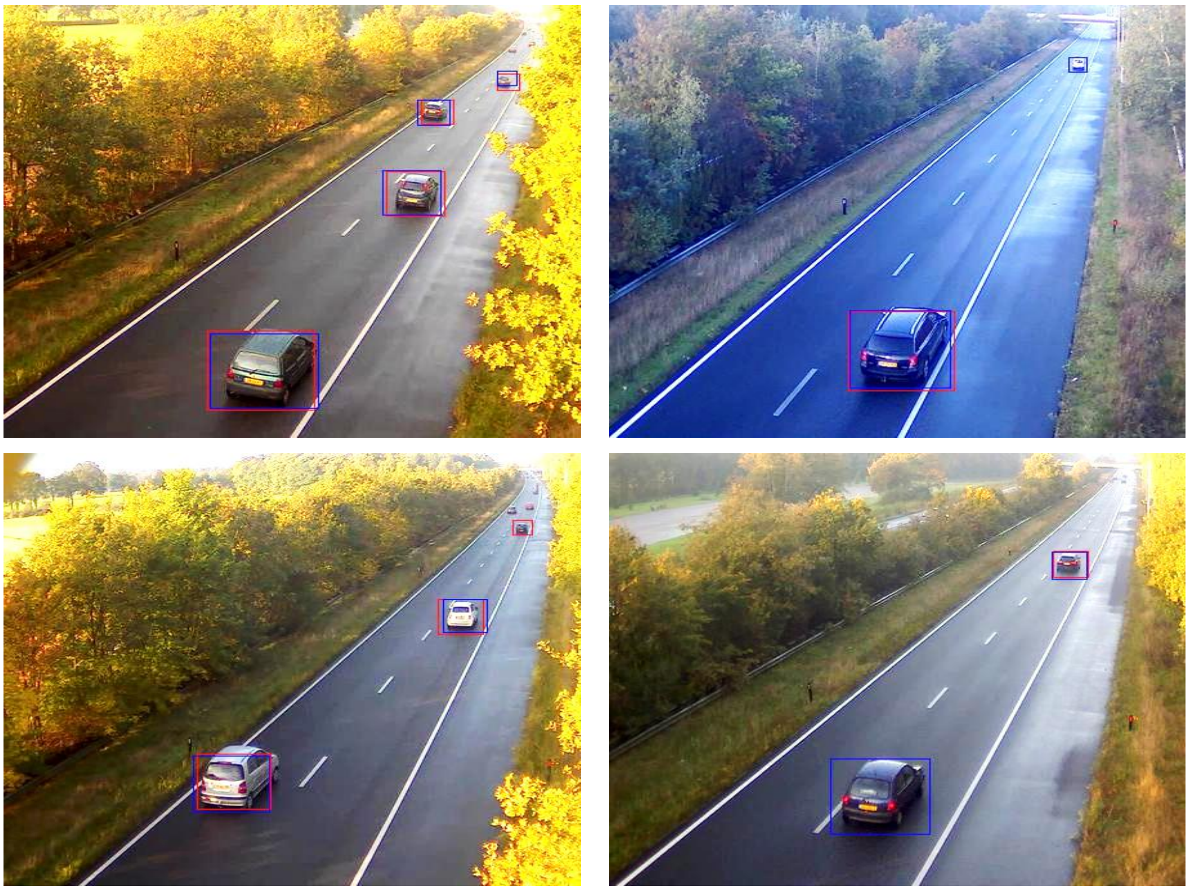
Figure 4: Example detections in several images. The GT detections (red) and our detections (blue) are shown.

5. CONCLUSIONS

We combined an end-to-end feedforward neural network for static detection with a recurrent Long Short-Term Memory (LSTM) network for multi-frame analysis. We presented a practical guide with special attention to the selection of the optimizer and batch size. The end-to-end network is able to localize and recognize the vehicles in video from traffic cameras. We show an efficient way to collect relevant in-domain data for training with minimal manual labor. Our results show that the combination with LSTM improves performance for the detection of moving vehicles.

ACKNOWLEDGEMENT

The work for this paper was conducted in the research program “Making Sense of Big Data” (MSoBD).

REFERENCES

[1] Alexe, B., Deselaers, T., Ferrari, V., “Measuring the objectness of image windows,” IEEE Trans. PAMI 34(11), 2189-2202 (2012).

[2] Azizpour, H., Laptev, I., “Object detection using strongly-supervised deformable part models,” ECCV, 836–849 (2012).

[3] Bell, S., Zitnick, C., Bala, K., Girshick, R., “Inside-outside net: detecting objects in context with skip pooling and recurrent neural networks,” IEEE CVPR, 2874 – 2883 (2016).

[4] Bengio, Y., “Deep learning of representations for unsupervised and transfer learning,” JMLR Proc. Unsupervised and Transfer Learning 27, 17-36 (2012).

[5] Bengio, Y., Bastien, F., Bergeron, A., et al., “Deep learners benefit more from out-of-distribution examples,” JMLR Proc. AISTATS, 164 – 172 (2011).

[6] Boer, M. de, Schutte, K. and Kraaij, W., “Knowledge based query expansion in complex multimedia event detection,” Multimedia Tools and Applications, 1-19 (2015).

[7] Boer, M. de, Brandt, P., Sappelli, M., Daniele, L.M., Schutte, K., Kraaij, W., “Query Interpretation–an application of semiotics in image retrieval,” Int. J. Adv. Software 3/4(8), 435-449 (2015).

[8] Bouma, H., Eendebak, P., Schutte, K., Azzopardi, G., Burghouts, G., “Incremental concept learning with few training examples and hierarchical classification,” Proc. SPIE 9652, (2015).

[9] Bouma, H., Azzopardi, G., Spitters, M., et al., “TNO at TRECVID 2013: multimedia event detection and instance search,” Proc. TRECVID, (2013).

[10] Carreira, J., Sminchisescu, C., “Constrained parametric min-cuts for automatic object segmentation,” in CVPR, (2010).

[11] Carreira, J., Caseiro, R., Batista, J., Sminchisescu, C., “Semantic segmentation with second-order pooling,” ECCV, (2012).

[12] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A.L., “DeepLab: Semantic image segmentation with deep convolutional nets atrous convolution and fully connected CRFs,” IEEE Trans PAMI, (2017).

[13] Dai, J., He, K. and Sun, J., “Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation,” IEEE ICCV, 1635-1643 (2015).

[14] Dalal, N., Triggs, B., “Histograms of oriented gradients for human detection,” IEEE CVPR, 886–893 (2005).

[15] Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L., “Imagenet: a large-scale hierarchical image database,” IEEE CVPR, 248–255 (2009).

[16] Donahue, J., Jia, Y., Vinyals, O., et al., “Decaf: A deep convolutional activation feature for generic visual recognition,” ICML, (2014).

[17] Endres, I., Hoiem, D., “Category independent object proposals,” ECCV, (2010).

[18] Erhan, D., Szegedy, C., Toshev, A., Anguelov, D., “Scalable object detection using deep neural networks,” IEEE CVPR, 2155–2162 (2014).

[19] Everingham, M., van Gool, L., Williams, C., Winn, J., and Zisserman, A., “The PASCAL visual object classes (VOC) challenge,” IJCV 88, 303–338 (2010).

[20] Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan. D., “Object detection with discriminatively trained part-based models,” IEEE Trans. PAMI 32(9), 1627–1645 (2010).

[21] Geiger, A., Lenz, P., Urtasun, R., “Are we ready for autonomous driving; The KITTI vision benchmark suite,” IEEE CVPR, 3354-3361 (2012).

[22] Girshick, R., Donahue, J., Darrell, T., and Malik, J., “Rich feature hierarchies for accurate object detection and semantic segmentation,” IEEE CVPR, 580–587 (2014).

[23] Girshick, “Fast R-CNN,” IEEE ICCV, 1440 – 1448 (2015).

[24] He, K., Zhang, X., Ren, S., Sun, J., “Spatial pyramid pooling in deep convolutional networks for visual recognition,” ECCV, (2014).

[25] Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S. and Malik, J., “Semantic contours from inverse detectors,” IEEE ICCV, 991-998 (2011).

[26] Hosang, J., Beneson, R., Schiele, B., “A convnet for non-maximum suppression,” German Conf. Pattern Recognition, 192 – 204 (2016).

[27] Hosang, J., Beneson, R., Dollar, P., Schiele, B., “What makes for effective detection proposals,” IEEE Trans. PAMI 28(4), 814 – 830 (2016).

[28] Huang, L., Yang, Y., Deng, Y., Yu, Y., "DenseBox: unifying landmark localization with end-to-end object detection," arXiv:1509.04874, (2015).
[29] Jia, Y., Shelhamer, E., Donahue, J., et al., "Caffe: convolutional architecture for fast feature embedding," Proc. ACM Multimedia, 675–678 (2014).
[30] Krizhevsky, A., Sutskever, I., and Hinton, G., "ImageNet classification with deep convolutional neural networks," Adv. NIPS, (2012).
[31] Kruithof, M., Bouma, H., Fischer, N., Schutte, K., "Object recognition using deep convolutional neural networks with complete transfer and partial frozen layers," Proc. SPIE 9995, (2016).
[32] Kuo, W., Hariharan, B., Malik, J., "DeepBox: Learning objectness with convolutional networks," IEEE ICCV, 2479 – 2487 (2015).
[33] Lenc, K., Vedaldi, A., "R-CNN minus R," CoRR 1506.06981, (2015).
[34] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., "Microsoft COCO: Common objects in context," ECCV, 740-755 (2014).
[35] Long, C., Wang, X., Hua, G., and others, "Accurate object detection with location relaxation and regionlets re-localization," ACCV, 260–275 (2014).
[36] Long, J., Shelhamer, E., Darrell, T., "Fully convolutional networks for semantic segmentation," CVPR, 3431–3440 (2015).
[37] LeCun, Y., Boser, B,. Denker, J., and others, "Backpropagation applied to handwritten zip code recognition," Neural computation 1(4), 541–551 (1989).
[38] Liang, X., Wei, Y., Shen, X., and others, "Proposal-free network for instance-level object segmentation," arXiv:1509.02636, (2015).
[39] Liu, W., Erhan, D., Szegedy, C., Reed, S., "SSD: Single shot multibox detector," ECCV, 21 – 37 (2016).
[40] Lowe, D., "Distinctive image features from scale-invariant keypoints," IJCV 60(2), 91–110 (2004).
[41] Najibi, M., Rastegari, M., Davis, L., "G-CNN: an iterative grid-based object detector," IEEE CVPR, 2369 – 2377 (2016).
[42] Papageorgiou, C., Poggio, T., "A trainable system for object detection," IJCV 38(1), 15-33 (2000).
[43] Park, E., Berg, A., "Learning to decompose for object detection and instance segmentation," ICLR Workshop, (2016).
[44] Pinheiro, P., Collobert, R., Dollar, P., "Learning to segment object candidates," Adv. NIPS, 1990 – 1998 (2015).
[45] Pont-Tuset, J., Arbelaez, P., Barron, J.T., Marques, F. and Malik, J., "Multiscale combinatorial grouping for image segmentation and object proposal generation," IEEE Trans. PAMI 39(1), 128 - 140 (2017).
[46] Redmon, J., Divvala, S., Girshick, R., Farhadi, A., "You only look once: unified real-time object detection," IEEE CVPR, 779 – 788 (2016).
[47] Redmon, J., Farhadi, A., "YOLO9000: Better, faster, stronger," IEEE CVPR, (2017).
[48] Ren, S., He, K., Girshick, R., Sun, J., "Faster R-CNN: Towards real-time object detection with region proposal networks," IEEE Trans. PAMI 39(6), 1137 – 1149 (2017).
[49] Schutte, K., Bouma, H., Schavemaker, J., et al., "Interactive detection of incrementally learned concepts in images with ranking and semantic query interpretation," IEEE Content-Based Multimedia Indexing CBMI, (2015).
[50] Sermanet, P., Eigen, D., Zhang, X., et al., "Overfeat: integrated recognition, localization and detection using convolutional networks," ICLR, (2014).
[51] Simonyan, K., Zisserman, A., "Very deep convolutional networks for large-scale image recognition," CoRR, abs/1409.1556, (2014).
[52] Szegedi, C., Reed, S., Erhan, D., Anguelov, D., Ioffe, S., "Scalable high-quality object detection," arXiv:1412.1441, (2014).
[53] Uijlings, J., Sande, K., Gevers, T., Smeulders, A., "Selective search for object recognition," IJCV, 104(2), (2013).
[54] Viola, P., Jones, M., "Robust real-time face detection," Int. J. Comp. Vision 57(2), 137–154 (2004).
[55] Yosinski, J., Clune, J., Bengio, Y., Lipson, H., "How transferable are features in deep neural networks," Adv. NIPS, (2014).
[56] Zeiler, M., Fergus, R., "Visualizing and understanding convolutional networks," ECCV LNCS 8689, 818-833 (2014).
[57] Zitnick, C., Dollar, P., "Edge boxes: Locating object proposals from edges," ECCV, (2014).