Online Vehicle Detection using Deep Neural Networks and Lidar based Preselected Image Patches

Stefan Lange[1], Fritz Ulbrich[1] and Daniel Goehring[1]

***Abstract*— In this paper we present a vehicle detection system using convolutional neural networks on 2d image data. Since realtime capabilities are crucial for object detection systems running in real-traffic situations, we will show how the calculation time of our algorithm can be significantly reduced by taking advantage of depth information from lidar sensors. One part of this work focusses on useful network topologies and network parameters to increase the classification precision. We will test the presented algorithm on an autonomous car in different real-traffic scenarios with regards to detection accuracy and calculation time and show experimental results.**

I. Introduction

The last decade saw a tremendous increase in driver assistance systems for road vehicles as cars, busses and trucks. The final development could result in a fully autonomous vehicle. To develop such a vehicle, it is important to construct systems which allow the vehicle to perceive and to analyze its environment. Therefore, different types of sensors can be useful. One of the most important class of sensors are camera systems which allow the detection and classification of objects in the environment.

Fig. 1: Our test vehicles MadeInGermany and e-Instein

In recent years, many image based classification methods have been developed and applied, some of them will be discussed later. In this work we applied convolutional neural network (CNN) classifiers on 2d images in combination with lidar and radar based data. The goal of this work is a quick classification of other vehicles in realworld traffic scenarios. For this work, we used the Caffe framework [10] as a software architecture to train and to run the CNN classifiers. Furthermore, this work discusses the structure and choice of parameters for the CNN. To examine useful sets of parameters for CNN classifiers on the given task, different CNNs were tested with different parameters. In addition, we created our own training data set. Finally, we tested the presented approach on our autonomous vehicle "MadeInGermany", see Fig. 1. We will take advantage of different sensors, as depicted in Fig. 2, especially camera images, lidar, and radar data. The configuration of lidar and radar sensors is presented in Fig. 3.

[1]Stefan Lange, Fritz Ulbrich and Daniel Goehring are with the Department of Computer Science, Freie Universität, Berlin, 14195, Germany, email: slange89@gmx.de, {fritz.ulbrich | daniel.goehring}@fu-berlin.de

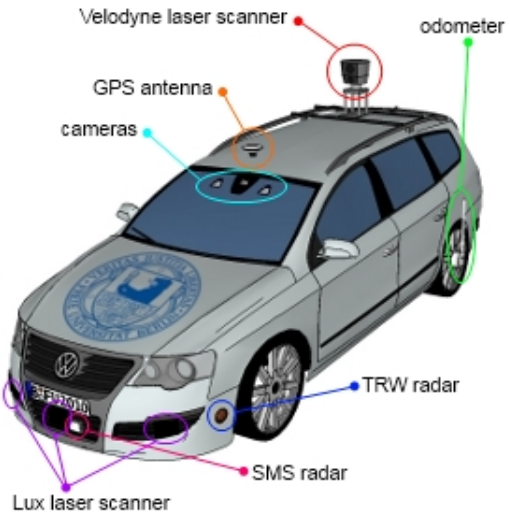

Fig. 2: Sensor configuration, camera, lidar-, and radar sensors on MadeInGermany. For our approach, the Lux laser scanners and TRW front radar (behind the VW sign) were used.

As a result, our system is able to process images with more than 5 frames per second. 80 percent of the objects within preselected image patches were classified correctly as vehicle or non-vehicle. The paper is structured as follows: Section II gives a brief overview about related work, Section III introduces basic concepts of CNNs. Experimental results are presented in Section IV, Section V gives a summary and outlook for future research.

II. Related Work

In the area of visual object detection and classification in 2d images, there have been a myriad of different approaches in recent years. One of the most popular approaches is the one of Viola and Jones [13] from year 2001. The main goal of their work was the real-time recognition of faces in grayscale images. In their algorithm, Haar features were applied in different scales on an integral image. An AdaBoost based algorithm [17] identifies the best classifying features

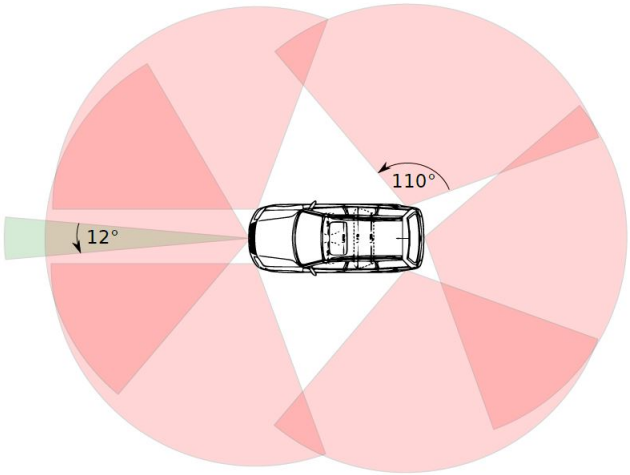

Fig. 3: Lidar sensor configuration and fields of view in pink(six sensors Ibeo sensors); the front facing 77 GHz radar in gray.

and sorts out images which do not show any faces. The approach is not very fast for training but allows real-time recognition. Another popular approach to recognize and classifiy objects on visual data are deformable parts models. The main idea is to detect objects consisting of multiple parts and to consider their relative position towards each other. The deformable parts model of Felzenszwalb et al. [12] applies HOG features which are used to train a Support Vector Machine classifier. For classification, a model consisting of three components is used. The first component is a root filter which defines the main appearance of the object core. The second part describes different parts of the object. The third part describes spatial relations of all the different parts towards the root filter.

One of the most recent and very promising field of research is Deep Learning using convolutional neural networks (CNNs). The approach has shown its superiority towards other methods in various classification competitions, e.g., in the Imagenet Large Scale Visual Recognition Challenge (ILSVRC), where Deep Learning based approaches scored best over the last years. The main work principles of CNNs try to mimic work principles of the human brain and its visual cortex. To do so, CNNs have multiple layers to recognize objects. One of their strengths is the ability to classify objects within thousands of classes. Krizhevsky et al. [2] created a CNN at the LSVRC-2010 which was capable to work with 1000 classes. Other advantages of CNNs are their relative robustness towards image noise, robustness towards rotation and change of position of objects within an image. A big disadvantage, though, is the long training time and the need for a large training data set.

Krizhevsky et al. [2] showed how training time of a CNN can be significantly reduced. Therefore, the up to then commonly used sigmoid or tanh functions were replaced by ReLU-Layers, which allowed an acceleration for the Backpropagation algorithm [4]. Furthermore, using Graphics cards for training and exploiting highly parallelized computing helped further reducing training and classification times. Lawrence et al. [14] developed a hybrid network from image sampling, self-organizing map (SOM) and a CNN to classifiy images from faces. They showed the superiority of a CNN towards a multi-layered perceptron (MLP). Lee et al. [8] developed a convolutional deep belief network with probabilistic max-pooling to classify an unlabeled data set. They showed how the network is able to generate visual high-level features efficiently. Recently CNNs have continued their success in applications to traffic perception scenarios, where pedestrians or vehicles need to be detected. Nguyen et al. [7] presented a CNN to recognize pedestrians and showed how their networks can be trained within a fraction of time compared to common training methods. In [6] an application of CNNs to detect vehicle colors was presented.

III. Application of CNNs for Realtime-Vehicle Recognition

A. Convolutional Neural Networks (CNNs)

Artificial neural networks (NN) consist of several layers of fully connected neurons. Each neuron gets an input vector $\vec{x} = x_1, \ldots, x_n$. The output o of each neuron is created by calculating the scalar product from a weight vector $\vec{\omega} = \omega_1, \ldots, \omega_n$ with the input vector $\vec{x}$ and applying an activation function f. Usually, f is a logistic function, e.g., a sigmoid function.

$$net = \sum_{i=1}^{n+1}(x_i \cdot \omega_i) \tag{1}$$

$$o = f(net) \tag{2}$$

For training of the network's weights, the backpropagation algorithm is used [4]. An error function E is defined over the squared differences between expected class index s_i (labeled in the training data) and calculated network outputs o_i.

$$E = \sum_{i=1}^{N}(o_i - s_i)^2 \tag{3}$$

The partial derivatives of this error function w.r.t. the weights define the gradient $\nabla E = (\frac{\partial E}{\partial \omega_1}, \ldots, \frac{\partial E}{\partial \omega_n})$, as described in [4]. The weight update $\Delta\omega_{ij}$ is generated by multiplying $\frac{\partial E}{\partial \omega_{ij}}$ with a learning rate μ.

$$\Delta\omega_{ij} = -\mu\frac{\partial E}{\partial \omega_{ij}} = -\mu \cdot \delta_j \cdot x_i \tag{4}$$

$$\delta_j = \begin{cases} e_j f'(net_j) & \text{, if } j \text{ is on last layer} \\ (\sum_{k=1}^{K} \delta_k \omega_{jk}) f'(net_j) & \text{, otherwise} \end{cases} \tag{5}$$

Here, e_j is the first derivative of the error function E at output neuron j on the last layer and δ_k is the backpropagated error from the kth neuron of the succeeding layer.

Convolutional neural networks (CNNs) are a special form of neural networks. A CNN consists of three different classes of layers [3]:

- convolution layers

- pooling or subsampling layers
- a set of fully-connected layers

The convolution layer was introduced by LeCun et al. [18]. In case of an input image Y, the image pixels with coordinates (i, j) are convoluted with a filter W with weights ω_{uv} resulting in an output image S with pixels s_{ij}.

$$s_{ij} = \sum_{u=0;v=0}^{M-1} y_{i+u,j+v}\omega_{uv} \tag{6}$$

An activation function $f(x)$ with bias b is applied to each s_{ij} of S.

$$o_{ij} = f(s_{ij} + b) \tag{7}$$

A convolution layer consists of not just one but several of these filters. A convolution layer is usually followed by a subsampling (or pooling) layer. In the presented approach max pooling was used [5], [9]. Here, the maximum value of a set of input values is propagated towards the next layer.

$$o_j = \max_{N\times N}(x_{n\times n} f(n, n)) \tag{8}$$

During backpropagation on a max pooling layer, the δ is only propagated to the neuron which created the maximum value during the feed-forward step [5].

Multiple alternating convolution and pooling layers are followed by a conventional fully-connected neural network (NN).

B. Further Improvements

For the calculation of the CNN the *Caffe* framework of the *Berkeley Vision and Learning Center* (BVLC) [10] was used. It supports performance speed up by GPU using *CUDA* [1].

Furthermore, stochastic gradient descent (SGD) as descibed in [4] was used for the calculation of the weight updates in each iteration. The update in the current iteration $\Delta\omega_i$ is calculated as convex combination of the gradient $L(\omega)$ and the previous weight update $\Delta\omega_{i-1}$.

$$\Delta\omega_i = \mu\Delta\omega_{i-1} - \alpha L(\omega_{i-1}) \tag{9}$$

$$\omega_i = \omega_{i-1} + \Delta\omega_i \tag{10}$$

The optimal learning rates α und μ must be experimentally determined [11]. Furthermore the parameter α is decreased during each iteration, which results in smaller weight updates with an increasing number of iterations.

The ReLU function $f(x) = \max(0, x)$ has the desired property that it can be calculated very efficiently. It's derivative is either 0 , for $x < 0$ or 1. It is usually used instead of the sigmoid function, especially while training CNNs.

$$f(x) = \max(0, x) \tag{11}$$

The ReLU layer is followed by a local response normalization, as introduced by Krizhevsky et al. [2].

C. Data Set

For the training of the CNN a data set of images extracted from the front camera of the autonomous vehicle "MadeInGermany" was used. The 64×64 images were extracted manually from the logged data from various trips through Berlin.

A training set (approximately 1400 vehicle and 1700 background images), a training-validation set (750 images of both classes) and a validation set (800 images of both classes) were created. During the training process, the data of the training set is sent through the network for a specified number of iterations and the corresponding weight updates are validated with the training-validation set to prevent overfitting. After finishing the training process, the resulting network is validated using the validation set.

D. Detection Module

For vehicle detection the Caffe CNN was integrated into the existing framework for autonomous driving as an integrated module. Input for the detection module is a 16 Bit grayscale image from the on-board camera (which is then converted to 8 bit grayscale) and bounding boxes of the obstacles surrounding the vehicle generated from lidar and radar fusion [15]. The three-dimensional bounding boxes are projected into the two-dimesional camara space. Bounding boxes which are not in the field of view of the camera are ignored. The different processing steps to extract image patches using lidar and radar sensory data are shown in Fig. 4 To guarantee the performance of the detector module, only the ten closest objects are selected for classification. These are the most important objects for the behavior of the car. In the next stage of the detection module, for each object a bounding rectangle is fitted around each projected bounding box. The rectangle is scaled up by 10 percent to ensure the whole car is inside and then enlarged to be square-shaped. The resulting square area is rescaled to 56×56 and passed to the CNN. The result is visualized as colored overlay in the image (green for car, red for background) and provided to other modules in the framework.

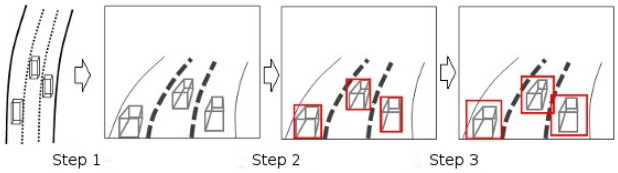

Fig. 4: Work steps of the detection module: Projection of three-dimensional bounding boxes in the two-dimensional camera space (step 1), calculating rectangular bounding boxes in camera space (step 2), enlarging two-dimensional bounding boxes to be square-shaped (step 3).

IV. Experimental Results

For the vehicle detection module, the lidar point cloud was clustered, representing different object within our vehicle's surrounding. Each point cluster is a vehicle object candidate. Its location was projected back into the image plane, the

corresponding rectangle within the image was classified by a CNN. This approach has the advantage that not all areas within the image have to be classified but only those where an object was detected - resulting in a faster execution time and higher frame rate of the whole approach.

Besides applying the CNNs, we also tried to figure out, how to set the parameters of the network to achieve the best classification performance.

A. Network topology

To optimize the network topology we created different networks with different topologies and compared them to each other. Here we compared the average precision while classifying 800 test images as a qualitative measure for the recognition and runtime. To accomodate for the high number of randomly chosen parameters, every network topology was trained three times. Therefrom the average of the maximum precision during training over time was created and used for comparison. Here, average precision means that we calculate the precision recall curve by incrementing a threshold parameter. The average precision is the integral (in our discrete case the sum) under the precision recall curve. Topologies of networks are shown in Table I.

B. Deep networks

To conduct research on the depth of a network, we created Net03 and Net04. Net04 contains three convolutional layers, whereas Net03 contains an additional convolutional layer. The filter size and their number were adjusted in such a way that the fully connected layer has to process the same amount of data, c.f. Table I. This means that the output of each layer ($width \times heigth \times$ *number of neurons*) before the fully connected layer stays approximately the same. The results in Fig. 5 show, that the maxima of Net03 are higher than those of Net04. Also, the averages in Fig. 5 show, that Net03 achieves a higher precision score with its additional convolution layer.

Taking a look at the execution times in Fig. 5, we see that Net04 with its fewer layers needs four times as much calculation time as Net03. This observation suggests that deeper networks can be calculated more efficiently than broad networks with larger filter sizes. In a next step, to continue with our analysis on network depth, we extended Net03 and Net04 by an additional fully-connected layer - resulting in Net05 and Net06. The fully-connected layer was placed within the already existing fully-connected layers. The added fully-connected layer is followed by an additional ReLU-layer, as shown in Table I.

Results of the comparison between network Net05 and Net06, as shown in Fig. 5 do not indicate a significant classification accuracy improvement compared to the networks without a fully-connected layer Net03 and Net04, as the variance between the three test runs is comparably large. It is still reasonable to assume that a further extension of a network is useful, since for both networks (Net05 and Net06), there are higher levels of peak precision, c.f., Fig. 5 than for Net03 and Net04, respectively.

Comparison Net05 with Net06 shows, that a further convolution layer leads to better results than a broader network, since Net05 has a better peak performance than Net06. Fig. 5 shows also that the runtime is barely affected by the fully-connected layer, whereas the further convolution layer in combination with a narrower network does decrease the runtime significantly.

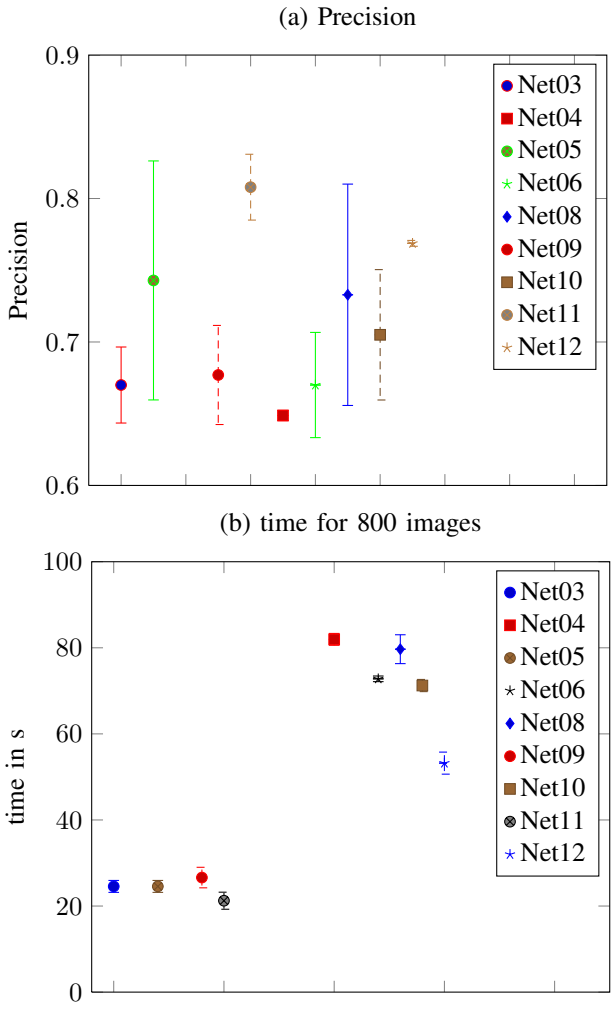

Fig. 5: Measurements on classification precision and running time; (a) shows the averages of the maxima of the average precision. (b) shows the averages of the running time for classification of 800 images.

C. Additional Dropout-Layer

Furthermore, Net06 was extended by a dropout-layer, to prevent overfitting as described in the work of Srivastava et al. [16], resulting in Net08. This layer was placed in between the fully-connected layers, and switches off half of all connections, c.f., I. Network Net08, as presented in Fig. 5, shows no significant average increase on classification performance compared to Net06, but reaches higher performance peaks. Regarding the calculation time, c.f., Fig. 5, dropout- and fully-connected layer do not seem to affect the running time during execution.

Fig. 6: Examples with a good classification precision. All vehicles within close and medium proximity could be correctly classified. Far away or occluded vehicles were not classified as a vehicle.

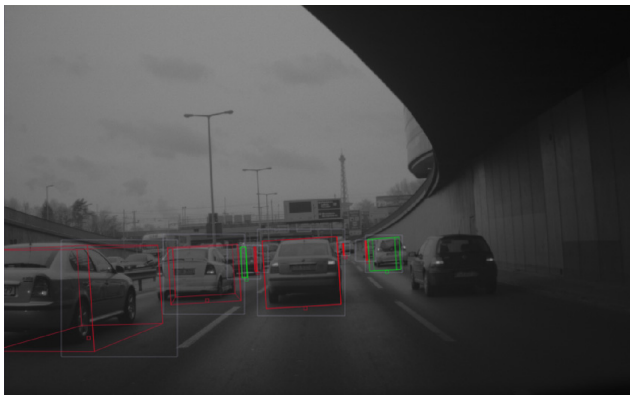

Fig. 7: Examples for multiple wrongly classified objects. Changing lighting conditions can affect the classifier

D. Gropping

In addition, the amount of training data was artificially increased by gropping, its effect on classification performance was analyzed. This means, we cut out a few patches of size 56×56 for each original 64×64 image patch and used those patches for training. Because of the smaller dimension of input images, the network needed to be adapted by reducing the convolution- and pooling parameters. Therefore, two networks were created, expecting a 64×64 image as an input while not performing gropping (Net09, Net10) and in addition, two more networks were tested (Net11 and Net12) which are identical to Net09 Net10, but received inputs of 56×56 images by gropping (Fig. I). Comparing the networks Net03 and Net09, see Fig. 5, we did not find significant differences in average precision, but Net10 shows significantly better results in terms of precision compared to Net04. This can be explained by the reduction of the filter size. It is not surprising to find that gropping improves the detection rate significantly, see Fig. 5, as demonstrated by Net11 or Net 12, while reaching higher precision rates than Net09 Net10. This demonstrates that gropping is a useful way to achieve good classification models using small data sets, or to increase precision while working on big data sets as well.

Net	**03**	**04**	**05**	**06**	**08**	**09/11**	**10/12**
Gropp	no	no	no	no	no	no/yes	no/yes
C+R	25/6	50/6	25*6	50/6	50/6	25/5	50/5
P+L	3/2	3/2	3/2	3/2	3/2	2/2	3/2
C+R	51/5	100/5	51/5	100/5	100/5	51/4	100/4
P+L	3/2	3/2	3/2	3/2	3/2	2/2	2/2
C+R	50/2		50/2			50/2	
C+R	40/2	256/2	40/2	256/2	256/2	40/2	256/2
P	3/1	3/1	3/1	3/1	3/1	2/1	2/1
F+R	500	500	500	500	500	500	500
D					50		
F+R	2	2	200	200	200	2	2
D					50		
F			2	2	2		

TABLE I: Topology of networks. C+R - convolution layer followed by a ReLU layer with "filter count / filter size". P[+L] - pooling layer [followed by local response normalisation] with "pooling raster size / step length". F[+R] - neuron layer [followed by a ReLu layer] with "neurons count". D - dropout layer with "dropped connection in percent". Networks 09 and 11 have identical topologies, but 11 use gropping, network 09 does not. Network 10 and 12 have identical topologies, but 12 use gropping, network 10 does not.

Fig. 5 shows that reducing the filter size of convolution layers, as done for all networks from Net04 to Net10, can have a positive effect on classification precision. However, in another example, this assumption was not true for a three layered network (Net03 to Net09). The precise effect of the filter size on precision and maturity still needs to be further investigated.

As a result, on the 800 test images we achieved a precision score of approx. 80 percent, i.e., 80 percent of images containing a vehicle were correctly classified as vehicle containing images. Fig. 6 and 7 show good and bad examples for vehicle classification.

V. Conclusion

The approach introduced in this paper, which is based on a convolutional neural network, was applied to a vehicle detector and classifier of an autonomous vehicle at the Freie Universität Berlin. A front camera in this vehicle provided the images on which the algorithm runs.

We could demonstrate that it is possible to train a CNN with less than 6000 images and a test data set of 800 images. The algorithm was able to classify more than 80 percent of the image patches correctly. The images used were taken from real traffic situations, in part in bad lighting conditions.

To optimize the network topology, several networks were trained on the data set. Comparing the resulting networks showed that deeper networks with more convolution layers, as well as more fully connected layers, can improve the detection rate significantly. Furthermore, the experiments showed that the choice of appropriate filter sizes leads to higher precisions. The optimal ratio of filter size to image size still needs to be determined. In addition, gropping helped increasing the robustness of the classificator w.r.t. varying positions of the object within the image.

The results presented in this paper show that a CNN based classifier is suitable for a real-time classification of vehicles.

Using the here described CNNs on preselected image patches, acquired by sensor fusion, allowed us to run the detection and classification algorithm with a frame rate of 5 Hz.

Still, especially a larger training data set would help to increase classification precision and robustness towards different lighting conditions, object poses within the image, different objects, and different camera parameters. It is likely that bigger training data sets decrease the classification precision variance over different training runs on the same network type.

Besides the aforementioned improvements, the ability to classify objects many classes, i.e., in our scenario pedestrians, traffic lights, traffic signs, different vehicly types, constitutes another very important area for future research.

ACKNOWLEDGMENT

This work was supported in part by the Deutsche Forschungsgemeinschaft (DFG), SPP 1835 - Kooperativ interagierende Automobile.

REFERENCES

[1] http://www.nvidia.de/object/cuda-parallel-computing-de.html.

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

[3] B. LeCun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer, 1990.

[4] D. E. Rumelhart, G. E. Hinton, Williams, J. Ronald. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[5] D. Scherer, A. Müller, S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks–ICANN 2010*, pages 92–101. Springer, 2010.

[6] R. Fuad Rachmadi and I. Ketut Eddy Purnama. Vehicle Color Recognition using Convolutional Neural Network. *ArXiv e-prints*, Oct. 2015.

[7] H. G. Nguyen, S. L. Phung, A. Bouzerdoum. Reduced training of convolutional neural networks for pedestrian detection. In *International Conference on Information Technology and Applications*, 2009.

[8] H. Lee, R. Grosse, R. Ranganath, Y. A. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[9] J. Nagi, F. Ducatelle, G. Di Caro, D. Ciresan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, L. M. Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pages 342–347. IEEE, 2011.

[10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[11] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[12] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[13] p. Viola, M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[14] S. Lawrence, C. L. Giles, A. C. Tsoi, A. D. Back. Face recognition: A convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, 8(1):98–113, 1997.

[15] M. Schnürmacher, D. Göhring, M. Wang, and T. Ganjineh. High level sensor data fusion of radar and lidar for car-following on highways. In *Recent Advances in Robotics and Automation*, pages 217–230. Springer, 2013.

[16] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research 15*, page pp: 19291958, 2014.

[17] Y. Freund, E. Schapire, E. Robert. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[18] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.