2013 IEEE Intelligent Vehicles Symposium (IV)
June 23-26, 2013, Gold Coast, Australia

Convolutional Neural Networks for Night-Time Animal Orientation Estimation

Raimar Wagner[1], Markus Thom[2], Michael Gabb[2],
Matthias Limmer[3], Roland Schweiger[3], and Albrecht Rothermel[1]

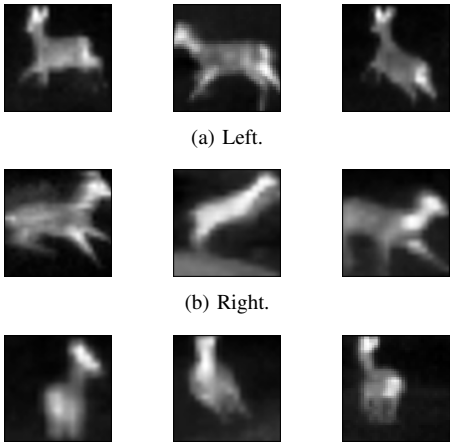
Abstract— **In rural areas, wildlife animal road crossings are a threat to both the driver and the wildlife population. Since most accidents take place at night, recent night vision driver assistance systems are supporting the driver by automatically detecting animals on infrared camera imagery. After detecting an animal on the roadside, the orientation towards the road can give a first cue for an upcoming trajectory prediction. This paper describes an novel classification-based scheme for night-time animal orientation estimation from single infrared images. Our system classifies already detected animals, in particular deer, as being either oriented left, right or back/front. We propose an approach based on Convolutional Neural Networks which learns multiple stages of invariant features in a supervised end-to-end fashion. Experiments show that our method outperforms baseline methods like HOG/SVM or boosted Haar-stumps on this task.**

I. Introduction

With increasing traffic in both urban and rural areas, the number of deer-vehicle collisions is on the rise. Recent studies have estimated that 726,000 of such collisions occur in the United States each year [1]. These result in over 200 human fatalities and over 1 billion dollars worth of damage per year [2]. The majority (80% - 95%, see [3]) of these accidents occur during night-time, when poor lighting conditions lower the visibility.

Car manufacturers are currently tackling these threats for both the driver and the wildlife population with on-board infrared night-vision systems which allow the driver to see three to five times farther than with standard low-beam lighting [4]. These systems not only let the driver see further, they are also able to automatically detect objects in the vehicle's environment. Early generations were restricted to pedestrians only. Current systems are also capable of automatically detecting wildlife animals such as deer [5].

After detecting animals and thereby roughly estimating their position with respect to the road, a behavioral prediction is necessary for advanced situation awareness. With a trajectory prediction that combines multiple cues into one prediction, possible animal road crossings and collisions are avoidable. A first cue for such a trajectory prediction system is the orientation of the animal. When an animal is already oriented towards the road, a trajectory that crosses the road is likely. Hence an orientation estimation on a frame-by-frame basis provides useful information for trajectory prediction and risk assessment.



(a) Left.

(b) Right.

(c) Back/Front.

Fig. 1: Night-vision deer images at various orientations.

Since our system estimates the orientation on the basis of a given detection, it relies on a separate detector component which is not part of this work. A short overview about techniques for automotive object detection is given in Sect. II. For every frame captured by an on-board infrared camera, such a detector component detects all deer in the image and returns a list of bounding boxes. Each bounding box is then processed by the proposed orientation estimation classifier which returns the estimated orientation for the shown deer. Sample images of all three orientation classes are depicted in Fig. 1. Our two-staged detection and orientation estimation approach is depicted in Fig. 2.

In this paper we propose a single-frame classification-based orientation estimation system for deer. Our *convolutional neural network* (CNN) approach outperforms state-of-the-art feature/classifier combinations on this task.

The remainder of this paper is organized as follows: An overview over the related work is given in Sect. II. Sect. III shows the network architectures and the training protocol used in this work. After describing the dataset in Sect. IV the experimental results are shown in Sect. V. The final Sect. VI contains a summary and a conclusion.

[1] Raimar Wagner and Albrecht Rothermel are with the drive-u / Institute of Microelectronics, University of Ulm, Ulm, Germany. {firstname.lastname} at uni-ulm.de

[2] Markus Thom and Michael Gabb are with the drive-u / Institute of Measurement, Control and Microtechnology, University of Ulm, Ulm, Germany. {firstname.lastname} at uni-ulm.de

[3] Roland Schweiger and Matthias Limmer are with Daimler AG R&D, Ulm, Germany. {firstname.lastname} at daimler.com

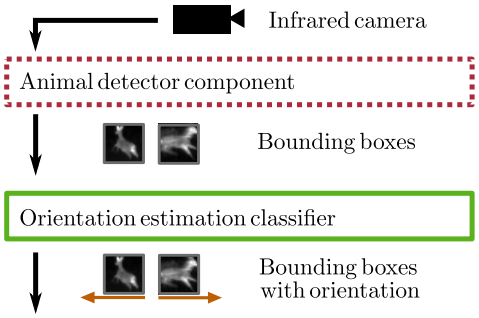


Fig. 2: Wildlife animals like deer are detected in infrared images by a detector component (dotted) and the proposed orientation estimation classifier returns an orientation estimation for each bounding box.

II. Related Work

Object detection in the automotive domain is mostly carried out in a sliding window fashion, that is a classifier is shifted over the input image at various positions and scales. The most frequently used features are edge-based (HOG [6], EOH [7], Haar-like Filters [8]) or learned from the pixel level (CNN [9], [10]). These features are then processed by boosted cascade classifiers [11], support vector machines [12] or multi-layer perceptrons [13] to compute a classification decision.

In the research community, little effort has been made on deer orientation estimation so far. Most applications for orientation estimation are related to vehicles or to the human pose. In [14] a vehicle orientation was inferred by combining a part and an outline detector on monocular image sequences. Others used stereo vision [15], [16] or optical flow [17] cues for vehicle orientation estimation. Estimating the human orientation from 2D images was either interpreted as an integrated detection and classification task [18] or – like in this work – as a separate post-processing step after the detection phase [19], [20], [21], [22].

While the previously mentioned methods mostly rely on hand-engineered features with a classifier on top, our approach is based on CNNs [23]. This neural network architecture learns multiple layers of invariant features in a supervised end-to-end fashion. Instead of manually tweaking features, all used features are learned directly from the data. Such CNNs are promising candidates for this classification task, since they have shown to be very competitive on automotive tasks like traffic sign recognition [24], [25] and pedestrian detection [9] as well as on non-automotive classification tasks [26], [27], [28]. Despite their computational complexity, [29] has shown that CNNs are well portable to embedded architectures which is a mandatory prerequisite in the automotive domain.

III. Architecture

To classify the orientation of a deer into one of three orientation categories (left, right, back/front) we employ CNNs on the pixel level. These consist of a hierarchy of layers, forming a hybrid feature extractor and classifier combination. Filter bank layers extract features from the input pattern, pooling layers reduce the spatial resolution by combining nearby feature responses, and fully connected layers represent neural networks without particular structure.

A. Layers

Three layer types are distinguished in this work while neglecting the input layer which just holds the input image:

a) Filter Bank Layer: The filter bank layer extracts local features by a convolution with a filter bank. The input of one filter bank layer are the n feature maps x_i from the previous layer. Each filter $k_j \in \mathbb{R}^{n\times m\times m}$ consists of n filter kernels $k_{j,i}$ with filter size $m \times m$. Every filter is connected to one output feature map y_j:

$$y_j = \tanh\left(b_j + \sum\nolimits_i k_{j,i} * x_i\right).$$

Here $*$ denotes the 2D-convolution-operator and the hyperbolic tangent is used as a squashing function. The trainable parameters are the filter coefficients in k_j and a bias b_j for each filter.

b) Pooling Layer: This layer type reduces the spatial resolution of the input feature map x_i and pools nearby feature responses into the output feature map y_i. This leads to robustness against distortions and small translations. Responses in non-overlapping rectangular regions of size $r \times r$ pixels are pooled with either an average-pooling operation or a max-pooling operation. The average-pooling returns the mean feature response of the corresponding input region:

$$y_i = \underset{r\times r}{\text{mean}}(x_i).$$

The max-pooling operation, which has shown superior performance in some scenarios [30], [31], [32], returns the maximum feature response of the corresponding input region:

$$y_i = \max_{r\times r}(x_i).$$

c) Fully Connected Layer: The fully connected layer models a standard one-layer neural network, which takes the input x, multiplies it with a trainable weight matrix W, adds a bias vector b and applies a squashing function f:

$$y = f(b + W \cdot x).$$

The squashing function f is either a $\tanh$ in the hidden layers, or a $\mathrm{softmax}$ function [13] in the output layer.

B. Hierarchy Construction

The aforementioned layers are then combined into a hierarchy of feature extractors (Filter Bank and Pooling Layers) which is followed by a fully connected neural network classifier with a $\mathrm{softmax}$ transfer function. This allows the outputs of the last layer to be interpreted as the probability to belong to one specific class. This work studies both one- and two-layered architectures, each either utilizing max- or average-pooling: The input layer of all networks has 40×40 pixels, which is the input dimension of the used data. The output layer has three neurons, one for each class (left,

Network with One-Layered Feature Extraction			
	Kernel	#Kernels	Feature Maps
Input			40×40 pixels
Filter Bank	7×7 pixels	16	34×34 pixels
Pooling	3×3 pixels	16	12×12 pixels
Fully Connected	200 neurons		
Fully Connected	3 neurons		

Network with Two-Layered Feature Extraction			
	Kernel	#Kernels	Feature Maps
Input			40×40 pixels
Filter Bank	7×7 pixels	16	34×34 pixels
Pooling	3×3 pixels	16	12×12 pixels
Filter Bank	5×5 pixels	32	8×8 pixels
Pooling	2×2 pixels	32	4×4 pixels
Fully Connected	200 neurons		
Fully Connected	3 neurons		

TABLE I: Network parameters of the used CNNs.

	Left	Right	Back/Front	Total
Training Dataset	3700	4244	2087	10031
Test Dataset	464	924	263	1651

TABLE II: Training and test dataset statistics.

right, back/front). The detailed topology of the two CNNs is depicted in Table I.

The network with one layer of feature extraction has a filter bank layer with 16 filter kernels of size 7×7 pixels which is followed by a 3×3 pixels pooling layer. The output of the pooling layer is then fed through a fully connected layer with 200 hidden neurons to three softmax output neurons.

The two layer feature extraction network utilizes the same first feature extraction layer, but has another filter bank layer with 32 kernels of size 5×5 pixels with a 2×2 pooling on top. An exemplary architecture with two layers of feature extraction is sketched in Fig. 3.

C. Training

All networks were trained with stochastic gradient descent from random starting points by minimizing the error E of the probabilistic outputs with respect to the labels of the training dataset. As we use $\mathrm{softmax}$ output neurons, we chose the cross-entropy loss function as E. Cross-entropy loss has been shown to have a better classification accuracy [33] and can lead to an improved test-error performance [34]. To allow the training procedure to be parallelizable on a GPU, we average the gradient over mini batches of samples prior to the weight update. This enables us to compute the gradient of all mini-batch samples in parallel and therefore achieve a speedup. For every weight κ in a kernel k and a learning rate η, the weight update after each mini batch is computed as follows:

$$\kappa_{t+1} = \kappa_t - \eta \frac{\partial E_t}{\partial \kappa}$$

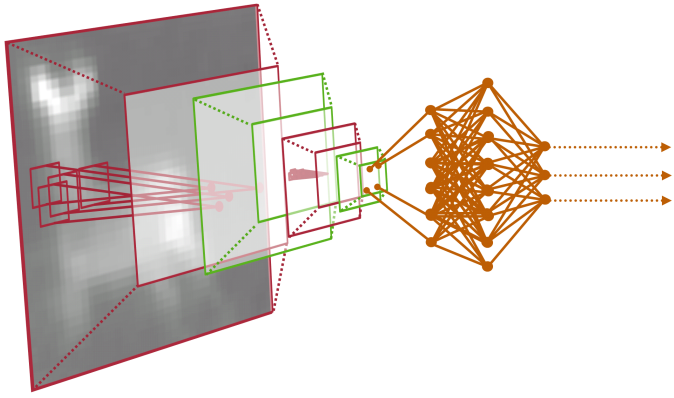


Fig. 3: Visualization of a two layer CNN with one filter kernel in each feature extraction layer. Filter bank layers (red) which densely extract features are alternating pooling layers (green) which combine nearby feature responses. The output of the second pooling layer is then fed into a fully connected neural network (orange). Figure is best viewed in color.

where $\frac{\partial E_t}{\partial \kappa}$ is the averaged gradient over the mini batch t. A training epoch is completed when every training sample was presented once as part of a mini batch. The training is stopped after a fixed number of epochs which was chosen large enough to ensure convergence.

IV. Dataset and Methodology

A dataset with 44 image sequences was captured by a vehicle mounted automotive night-vision infrared camera, totaling in 6491 frames. In all sequences the deer were manually labeled with a bounding box and an orientation class. The 44 sequences were then split up into 35 sequences for training and 9 sequences for testing. Jittering and mirroring the raw samples resulted in 10,031 trainings and 1651 test samples, see Table II. These samples were resized using bilinear interpolation to match the network input size of 40×40 pixels. Besides subtracting the mean and dividing by the standard deviation of the image, no further preprocessing was applied.

V. Experiments

In addition to the experiments with the proposed CNNs, we compared our approach with two baseline methods which are common in the automotive domain: histograms of oriented gradients (HOG) combined with a linear support vector machine (SVM) [6] and boosted Haar-like filters [11]. The results are summarized in Fig. 4.

A. HOG and Linear SVM

For this experiment, we considered rectangular HOG features [6] combined with a linear SVM [35]. Rectangular HOGs build up local histograms of different edge orientations. These histograms are then contrast normalized within larger regions. The concatenation of the normalized histograms form the feature vector.

In this work, the HOG feature was parameterized with 9 orientation bins, cell sizes from 4×4 to 10×10 pixels

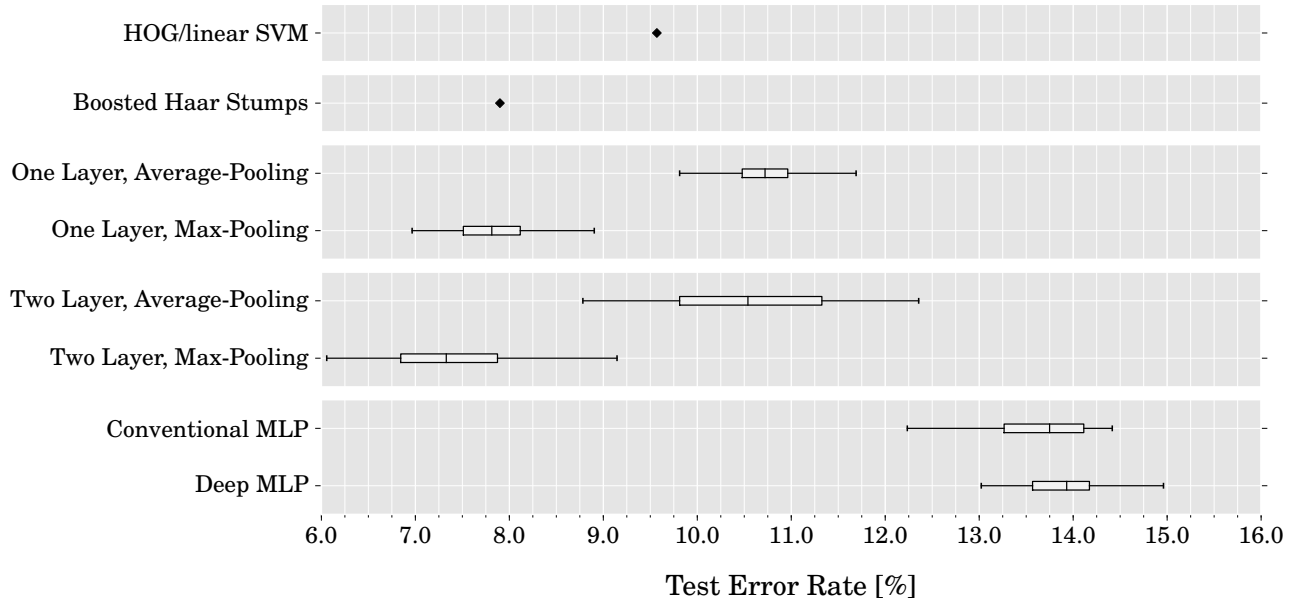


Fig. 4: Error rates [%] on the test dataset for HOG/linear SVM, boosted Haar-Stumps, one- and two-layered feature extraction CNN and MLP architectures.

Predicted	Left	Right	Back/Front
Label			
Left	88.36%	0.21%	11.42%
Right	1.51%	97.94%	0.54%
Back/Front	9.12%	1.14%	89.73%

TABLE III: Confusion matrix for the best trained two layer convolutional neural network.

and block sizes from 2×2 to 4×4 cells. The block stride was fixed to half of the block size [6]. To select the SVM regularization parameter C for every combination of cell and block size, a validation set was split up from the training set where the best performing C from 2^{-10} to 2^{-1} was chosen according to the best validation set performance. Afterwards the whole training set was trained with the chosen regularization parameter C and the performance on the test dataset was computed. The best performing HOG/SVM combination used a cell size of 8×8 and a block size 5×5. It achieved a test error rate of 9.57%.

B. Boosted Haar-Filters

To compare the proposed concepts, we trained several different boosted multi-class classifiers [36] using Haar-like filters [11] as basic image primitives. These have been widely used in the past and have shown to be applicable to a wide range of problem settings (see [37] for an overview). Typically, feature values are combined by boosting Decision Trees [38] of a fixed depth. Besides the most basic case—using the feature response in conjunction with a single threshold (often termed Decision Stump)—we used trees of depth 2 and 4 in our experiments.

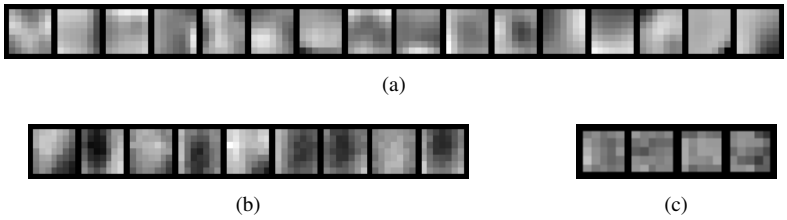
Recently, it was shown that boosting product terms instead of the traditional Decision Trees leads to an improved classification performance [39]. Thus, we incorporated this boosting technique as well. We investigated setups with 2, 3 and 4 product terms.

In our experimental setup we boosted 100 randomly chosen Haar-like filters per boosting round (LazyBoost) until convergence on the training dataset. As can be expected, the most complex boosted classifier performs best in our experiments, i.e. the deeper trees outperform the simple Decision Stumps and the product learner outperforms the tree learner. The best performing classifier is the boosted product with 3 product terms per weak learner, reaching a classification performance of 7.95%.

C. Convolutional Neural Networks

All networks introduced in Sect. III were trained on a GPU implementation[1]. The learning rate was fixed for all experiments at 10^{-3} and all weights were initialized from a normal distribution with a standard deviation of 0.01. The training procedure was carried out as a stochastic gradient descent with a mini batch size of 32 samples for 700 epochs. Every network was trained 50 times from a different random initialization to suppress random effects from the initial weights and the random order of sample presentation. From the four network types introduced in Sect. III, the two layer max-pooling network performs best with a mean test error rate of 7.51%. A second layer of feature extraction has only led to a slight performance gain for both types of pooling. For average-pooling networks, an improvement on the mean test error rate of 0.14% is reported when compared to the

[1] `http://code.google.com/p/cuda-convnet/`



(a) (b) (c)

Fig. 5: Learned first layer filter kernels of the max-pooling one layer network (a) and the two layer network (b). Second layer filters of the two layer max-pooling network are shown in (c).

(a) (b) (c)

Fig. 6: Wrong classified testset samples: (a) label: left, predicted: back/front. (b) label: back/front, predicted: left. (c) label: right, predicted: left

one layer network. The max-pooling network gains slightly more with an improvement of 0.29%.

The most important factor to achieve a performance superior to competing methods (see Sect. V-B & Sect. V-A) is the use of max-pooling instead of conventional average-pooling. This was already shown in recent literature for other image recognition tasks [31]. With max-pooling the mean test error decreased for one-layer networks from 10.68% to 7.80% and for the two-layer networks from 10.54% to 7.51% The confusion matrix for the best performing two-layer network is shown in Table III; most errors occur in the confusion of the back/front and the left class. This is also the case for the competing approaches which have similar confusion matrices. We found that the test dataset contains a lot of samples for which it is hard even for humans to distinguish between left or back/front. Such examples are depicted in Fig. 6. The resulting filter kernels of the CNN are shown in Fig. 5. They look like unsharp blob detectors. This is assumed to come from unsharp object boundaries, especially when smaller samples were scaled up to the input size of 40×40 pixels.

Since [40] had success in training very deep MLP for vision tasks, we evaluated a deep MLP with $1000 - 500$ hidden units and a standard MLP with 100 hidden units on our dataset in the same fashion. The conventional MLP with 100 hidden neuron performed quite well for its architectural simplicity. With a mean error rate of 13.58% it slightly beats the deep MLP (mean test error: 13.95%). The use of an infinite number of deformed training samples made the training of a deep MLP by pure gradient descent possible in [40]. With our small data set the training of such a deep MLP did not lead to improved results.

VI. Conclusion

This paper presented a novel approach for orientation estimation of deer at night. Our convolutional neural network model estimates the orientation of deer from low-resolution night vision images with a mean test error rate of 7.51% on a disjoint test dataset and hereby beats competing baseline methods such as HOG/SVM or boosted Haar-like filters.

For future work on the classifier side, the impact of unsupervised pre-training should be investigated. Given such a small dataset, an unsupervised pre-training of the feature extraction stages could be beneficial [41]. On the orientation estimation side, incorporating cues such as optical flow could further increase the performance. This could be especially beneficial when not dealing only with simple orientation classes but with an angular estimation of the animal's orientation. Additionally an integration over time to filter outliers could result in a more accurate orientation prediction. It should also be investigated, if an extension to other wildlife animal species like boar is feasible.

References

[1] M. Conover, W. Pitt, K. Kessler, T. DuBow, and W. Sanborn, "Review of human injuries, illnesses, and economic losses caused by wildlife in the united states," *Wildlife Society Bulletin*, vol. 23, no. 3, pp. 407–414, 1995.

[2] M. Conover, "Monetary and intangible valuation of deer in the united states," *Wildlife Society Bulletin*, vol. 25, no. 2, pp. 298–305, 1997.

[3] L. Mastro, M. Conover, and S. Frey, "Deer-vehicle collision prevention techniques," *Human-Wildlife Interactions*, vol. 2, no. 1, p. 75, 2008.

[4] K. Schreiner, "Night vision: infrared takes to the road," *Computer Graphics and Applications, IEEE*, vol. 19, no. 5, pp. 6–10, 1999.

[5] Daimler AG, "Mercedes-benz 'intelligent drive' tecday: Networked with all senses," Press release, Nov. 2012.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. of CVPR*, 2005.

[7] K. Levi and Y. Weiss, "Learning object detection from a small number of examples: the importance of good features," in *Proc. of CVPR*, 2004.

[8] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proc. of ICCV*, 1998.

[9] M. Szarvas, A. Yoshizawa, M. Yamamoto, and J. Ogata, "Pedestrian detection with convolutional neural networks," in *Proc. of IV*, 2005.

[10] C. Wöhler and J. Anlauf, "An adaptable time-delay neural-network algorithm for image sequence analysis," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1531–1536, 1999.

[11] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2001.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[13] C. Bishop, *Neural networks for pattern recognition*. Oxford Clarendon Press, 1995.

[14] M. Gabb, O. Löhlein, M. Oberländer, and G. Heidemann, "Efficient monocular vehicle orientation estimation using a tree-based classifier," in *Proc. of IV*, 2011.

[15] A. Barth and U. Franke, "Where will the oncoming vehicle be the next second?" in *Proc. of IV*, 2008.

[16] B. Barrois, S. Hristova, C. Wöhler, F. Kummert, and C. Hermes, "3D pose estimation of vehicles using a stereo camera," in *Proc. of IV*, 2009.

[17] T. Suzuki and T. Kanada, "Measurement of vehicle motion and orientation using optical flow," in *Proc. of ITSC*, 1999.

[18] M. Enzweiler and D. Gavrila, "Integrated pedestrian classification and orientation estimation," in *Proc. of CVPR*, 2010.

[19] T. Gandhi and M. Trivedi, "Image based estimation of pedestrian orientation for improving path prediction," in *Proc. of IV*, 2008.

[20] C. Nakajima, M. Pontil, B. Heisele, and T. Poggio, "Full-body person recognition system," *Pattern Recognition*, vol. 36, no. 9, pp. 1997–2006, 2003.

[21] H. Shimizu and T. Poggio, "Direction estimation of pedestrian from multiple still images," in *Proc. of IV*, 2004.

[22] A. Schulz, N. Damer, M. Fischer, and R. Stiefelhagen, "Combined head localization and head pose estimation for video-based advanced driver assistance systems," in *Pattern Recognition*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6835, pp. 51–60.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in *Proc. of IJCNN*, 2011.

[25] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *Proc. of IJCNN*, 2011.

[26] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[27] Y. LeCun, F. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. of CVPR*, 2004.

[28] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. of ICCV*, 2009.

[29] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *Proc. of ISCAS*. IEEE, 2010, pp. 257–260.

[30] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber, and G. Fricout, "Steel defect classification with max-pooling convolutional neural networks," in *Proc. of IJCNN*, 2012.

[31] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," *Artificial Neural Networks–ICANN 2010*, pp. 92–101, 2010.

[32] K. Labusch, E. Barth, and T. Martinetz, "Simple method for high-performance digit recognition based on sparse coding," *Neural Networks, IEEE Transactions on*, vol. 19, no. 11, pp. 1985–1989, 2008.

[33] R. Dunne and N. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," in *Proc. of Aust. Conf. on the Neural Networks*, vol. 185. Citeseer, 1997.

[34] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. of ICDAR*, 2003.

[35] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[36] D. Benbouzid, R. Busa-Fekete, N. Casagrande, F. Collin, B. Kégl *et al.*, "Multiboost: a multi-purpose boosting package," *Journal of Machine Learning Research*, vol. 13, pp. 549–553, 2012.

[37] M. Gabb, R. Wagner, M. Gressmann, O. Hartmann, O. Löhlein, R. Schweiger, and K. Dietmayer, "Feature selection for automotive object detection tasks-A study," in *Proc. of ICCE-Berlin*, 2012.

[38] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[39] B. Kégl and R. Busa-Fekete, "Boosting products of base classifiers," in *ICML*, 2009.

[40] D. Ciresan, U. Meier, L. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[41] D. Erhan, Y. Bengio, A. Courville, P. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.