In: Proceedings of the 2012 12th International Conference on Hybrid Intelligent Systems (HIS), pp. 531-536, Pune, IN, December 4-7, 2012.

Adaboost and Hopfield Neural Networks on Different Image Representations for Robust Face Detection

Nils Meins*, Doreen Jirak*, Cornelius Weber*, Stefan Wermter*
* *Department of Informatics, Knowledge Technology*
University of Hamburg, Hamburg, Germany
{*meins, jirak, weber, wermter*}*@informatik.uni-hamburg.de*

***Abstract*—Face detection is an active research area comprising the fields of computer vision, machine learning and intelligent robotics. However, this area is still challenging due to many problems arising from image processing and the further steps necessary for the detection process. In this work we focus on Hopfield Neural Network (HNN) and ensemble learning. It extends our recent work by two components: the simultaneous usage of different image representations and combinations as well as variations in the training procedure. Using the HNN within an ensemble achieves high detection rates but shows no increase in false detection rates, as is commonly the case. We present our experimental setup and investigate the robustness of our architecture. Our results indicate, that with the presented methods the face detection system is flexible regarding varying environmental conditions, leading to a higher robustness.**

***Keywords*-Adaboost, Face Detection, Computer Vision, Hopfield Neural Network, Ensemble Learning**

I. Introduction

In many detection tasks involving visual input processing one main question is how robust the developed detection technique is regarding e.g. different light conditions, non-stationary background, and natural image noise. In particular, face detection is challenging as some important face-representing features like eyebrows or nose could be hard to detect. In addition to robustness, the real-time constraints met in video streaming or in human-robot-interaction scenarios further push research towards fast algorithms. A major step in face detection is the feature extraction. Here, the computation of Haar Features [1] on grey-valued images has shown great success in detection and recognition applications. Our approach presented in this work relies on an architecture recently presented in [7]. We perform experiments with grey-valued images for input into the Adaboost algorithm (a review is presented in section II-A).

The underlying motivations for this work are the following: First, we aim at investigating the robustness of our existing approach concerning different preprocessing steps on the images employed in the detection process, by e.g. a thresholding procedure as explained later. Another approach to account for different image representations aiming for invariances establishes the definition of so-called Gradientfaces [4], which enables detection on images captured under varying illumination conditions. This method show good results on data-sets containing faces captured even outdoors. Magalhaes et al. [5] extends the mentioned approach by applying Adaboost on Gradientfaces. The authors motivate this combination as Gradientfaces are illumination-invariant and Adaboost is one of the fastet computational method in the field of face detection. Their work has been evaluated on face images in an artificial illumination setting but also on public data-sets like the YALE database, showing that the approach is superior to other methods.

Second, we introduce a new training scheme for classification. One crucial step in our framework is modification of the Adaboost algorithm and the use of Hopfield Neural Networks (HNN). Although Adaboost is a state-of-the-art algorithm it often has been modified in the last decade to improve accuracy. Recently, Wang et al. [9] have extended AsymBoost and cost-sensitive LogitBoost by introducing an asymmetric loss function for reaching a high detection rate and a low false-positive-rate. Here, our goal is to exploit the characteristics of HNN in terms of good detection rates while decreasing the false positive rate. This is an important aspect, since in general high detection rates come along coincidentally with high false positives. One potential solution is to change the training procedure during ensemble training. Generally, the Adaboost algorithm performs training and classifier selection on one training set. We alter this by selecting the classifier on combinations of differently preprocessed images.

This paper is structured as follows. In section II, we present our methods and the overall architecture. Section III describes the experimental set-up and shows the analysis on the obtained results. The paper closes with a discussion of the results and provides an outlook for future work on this topic in section IV.

II. Methodology and Overall Architecture

A. HNN-Ensemble

As introduced in our work [7], we use an ensemble consisting of HNN as members. Each HNN uses a vector of rectangle pixel sums as input. Our system is based on the architecture introduced by Viola and Jones [6] but differs in usage of the weak classifiers and the features. We use the integral image for fast calculation of this input vector.

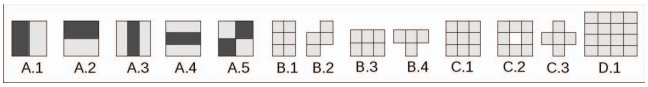

Figure 1. To calculate the Haar-like feature value, the pixel sum of the light grey rectangle is substracted from the pixel sum of the dark grey rectangle. Type A.1 to A.5 are the original Haar-like features from Viola and Jones [6]. The examples for the different types B to D are used for the HNN. The pixel sums of every small rectangle is part of the input vector.

Therefore the structure is similar to Haar-like features. For creating the ensemble we use the Adaboost algorithm.

The HNN was trained by Hebb-learning using only the positive samples, i.e. face images. For every HNN one possible patch with several rectangle pixel sums was created and used as input vector, as depicted in figure 2. The decision whether the converged HNN state indicates a learned patch was achieved by calculating the Euclidean distance from the reached stable state and the average of the positive training patches that were used for learning. For learning the best distance, we also considered negative samples, i.e. background and several non-face images.

As input for the HNN we used in addition to the Haar-like features introduced by Viola and Jones (see A.1 to A.5 in figure 1) also more complex rectangular structures (see B to D in figure 1).

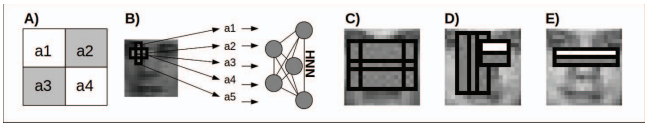

Figure 2. Visualisation of Haar-like feature usage. A) HNN uses the input vector $(a1, a2, a3, a4)$ instead of single value $(a1 + a4) - (a2 + a3)$ used by threshold classifier. B) Feature mapping to an HNN. C), D) and E) show samples of the Haar-like features. C) shows the first chosen Haar-like feature of the HNN ensemble using C.1. D) shows the first two chosen Haar-like features of an HNN+threshold ensemble using A.1 to A.5. E) shows the first chosen Haar-like feature of a threshold ensemble using A.1 to A.5.

B. Image Preprocessing

We want to emphasize that the computer vision methods for altering the input images can be applied in one iteration with low computational costs. The idea is that all of our used vision processings can be computed during the creation of the integral image representation. Integral images were first introduced in 1984 in the computer vision community [8] and measure the average intensity of grey-valued images by pixel summation. The concept has gained more attention with the publication of Viola and Jones in 2001.

Thresholding: For our experiments, we use fix and adaptive thresholding. Thresholding assigns every pixel that is greater than θ a value a and, respectively, a value b when the value is less than θ. For a and b we use numerical values in the range of 0 and 255. For the adaptive threshold we change our threshold θ in every step. $\theta = (\beta * \theta + p_i)/(\beta + 1)$, where i is the index and p_i is the value of the current pixel. In our experiments, we attained good results for $\beta = 30$. So, the current threshold is an averaging of the current pixel value and the last threshold. Additionally, the factor β ensures that the weighted average is focussed on the last threshold. This procedure creates some artifacts in the image, that would not exist by fixing the threshold. We accept this drawback as our underlying assumption is that this will not affect the classification results due to computation of rectangle pixelsums.

Image Subtraction: Our image substraction is based on the principle of a Sobel filter, i.e. we subtract two different pixels from the same image. If we calculate the difference of a pixel with another pixel two positions left of the image, it is similar to a $[-1, 0, 1]$ Sobel filter. The resulting values of the subtraction were applied to a thresholding so that the pixel of our resulting image is 0 or 255.

As stated above the idea to use such simple computer vision techniques is to obtain the ability of calculating all performed image modifications in one iteration with just a few more computational steps.

C. Combinations of Image Representations

There are different ways of combining image representations as further input for the ensemble. Basically, there are two important possibilities. On the one hand, we combine different image representations through computer vision methods and on the other hand, we take advantage of special classifier characteristics.

The following sections exploit combinations of image processing techniques. Their benefit is dependent on the classifier finally employed in a specific classification task. However, the method itself is independent of the used classifier. Also we introduce methods that depend on the classifier used and cannot be "as is" transformed to another classifier.

1) Averaging Image Representations: Applying one computer vision method to an image gives us as result a new image representation. We want to benefit from different image preprocessing methods. To combine these different image representations as one input image, we calculate the average image as illustrated in the architecture figure 3.B. The average image is the average for every dedicated pixel value within the same position.

2) Concatenating Image Representations: One method for combination is the so-called *voting*. For every image representation the classifier calculates a result and the final result will be the voting of the single results.

In order to take advantage of the HNN dynamics, we concatenate the HNN to a bigger HNN architecture. It is mentioned above, that we apply an HNN to different image representations, but uncoupled from each other. Every image representation was classified separately. The idea is to put simultaneously different image representations -i.e. images,

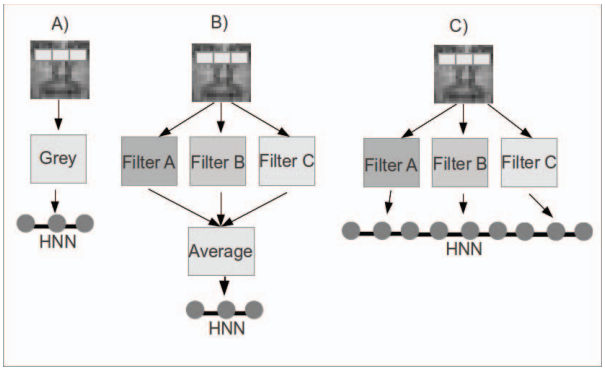

Figure 3. A) depicts the standard approach. One grey-valued image is used as input for the ensemble and therefore for the entire HNNs. B) illustrates the average solution. The captured image is prepared by three different filters. One image is calculated by averaging every pixel. C) shows the concatenation solution. The HNN is duplicated to handle simultaneously the results from Filter A, B, and C.

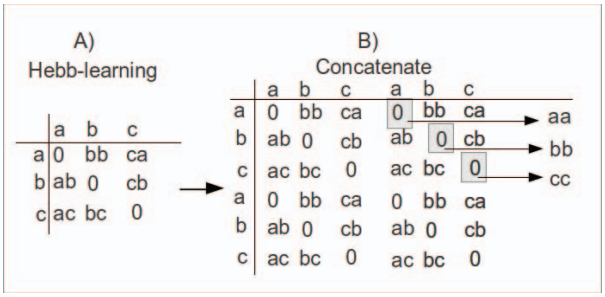

Figure 4. Concatenation of one HNN to one 3times increased HNN.

processed by different vision methods or filters- into the same HNN. This implies that we obtain a concatenated input version as a new input. As described in figure 3.C) we process the original image representation with different vision methods (Filter A to C in figure 3). Every image representation gets one input vector as outcome. We put these vectors one after another to get one input vector that contains all the different input vectors. Now the concatenated vector has the length of the accumulated lengths of the single vectors. The HNN is created by concatenating the single weight matrices.

3) Zero Diagonal of HNN: Because of the HNN architecture each matrix contains a zero diagonal (see [3]). An HNN matrix is forced to have zero value for every element where the index i equals index j. This would not happen if the HNN was constructed over a "normal" pattern (except the value itself was zero). Now, if we just duplicate a copy of the HNN matrix, we will have a zero value on different places (not only for $i \neq j$). We have tested three solutions. The first is to leave the value as it is, so leave it as zero. The second is to fix the value to 1. The last solution is to calculate $f(cp_i * cp_j)$, where f is a normalization step, cp is the concatenated pattern and i, j are the index of the elements as depicted in figure 4. The two approaches reach mostly similar results while the last one reaches slightly better results. Our choice is the last of these solutions, because it fits to the idea of Hebb-learning.

4) Altering Adaboost: By changing the Adaboost process in a small step, we try to train a classifier that is robust against changes of the environmental conditions. During the Adaboost trainingiteration, the weak classifier was trained and afterwards the error of this classifier was calculated. We train our HNN in the same way, but for calculating the error we execute our classifier to different image representations, but the same training set. For example, we train an HNN with the grey-valued training images, but then we calculate the error of this classifier by using the concatenated simultaneous input as described above (see II-C2).

III. Experiments and Analysis

The underlying question throughout the experiments is, how it is possible to combine different image preprocessing methods to improve classification accuracy.

5) Training- and Test Set: We are using as training set the MIT CBCL face database[1] and parts of the CMU frontal face database[2] as test set, as in our previous work [7]. The training of the ensembles was performed on a set of grey-valued images. The same applies to the test set for classification evaluation.

A. Initial Observations

For testing, we applied the ensemble on combinations of images, varying the way they were processed in a previous step. The classifier is assumed to achieve high detection rates for distinct input, relaxing experimental constraints like stationary background or equal light conditions. Clearly, this leads to more robust and flexible systems, while maintaining comparable training time.

Table I shows the classification results of one ensemble containing threshold classifiers and another ensemble containing HNN as a weak classifier. Table I shows results on a test set with grey-valued images and also the results on the same test set, but preprocessed with a simple thresholding technique (see II-B).

As depicted in table I, varying parameters derived from thresholding show similar results for the classification behaviour of the HNN in the range of the threshold parameter from 80 to 120 compared to the results of the grey-valued images in the first line of table I. This result underlines the relative stability of the classification behaviour of the HNN, especially compared to the threshold classifier as seen in the lower part of table I. Although this is a quite simple method with a fix threshold, the HNN reaches proper results compared to the performance of the grey-valued images.

[1] http://cbcl.mit.edu/software-datasets/FaceData2.html
[2] http://debut.cis.nctu.edu.tw/čhing/Face/FaceDataBaseSurvey/FaceDataBase.htm

combining	DR(%)	FP	Preparation	Classifier
-	76	90	grey	HNN C.1
-	50	146	thres-40	HNN C.1
-	76	129	thres-80	HNN C.1
-	78	115	thres-100	HNN C.1
-	67	113	thres-120	HNN C.1
-	29	106	thres-160	HNN C.1
-	3	98	thres-200	HNN C.1
-	86	250	thres-Adaptive	HNN C.1
-	74	79	grey	Threshold A.1-A.5
-	21	89	thres-40	Threshold A.1-A.5
-	62	91	thres-80	Threshold A.1-A.5
-	55	92	thres-100	Threshold A.1-A.5
-	42	80	thres-120	Threshold A.1-A.5
-	15	81	thres-160	Threshold A.1-A.5
-	2	79	thres-200	Threshold A.1-A.5
-	75	113	thres-Adaptive	Threshold A.1-A.5

Table I
HNN C.1 AND THRESHOLD A.1-A.5 WITH DIFFERENT IMAGE PREPROCESSING (THRESHOLDING). THE 80 OF "THRES-80" IS THE THRESHOLD SET FOR THE VISION METHOD.

In the following section we analyze the training procedures applied on modified input images and the effect on the classification resulting in moderate false positive rates.

1) Averaging Image Representations: In this section we perform different image processing methods we would like to apply for the actual classification. Then we calculate the average image of the different representations as described in the architecture figure 3.B).

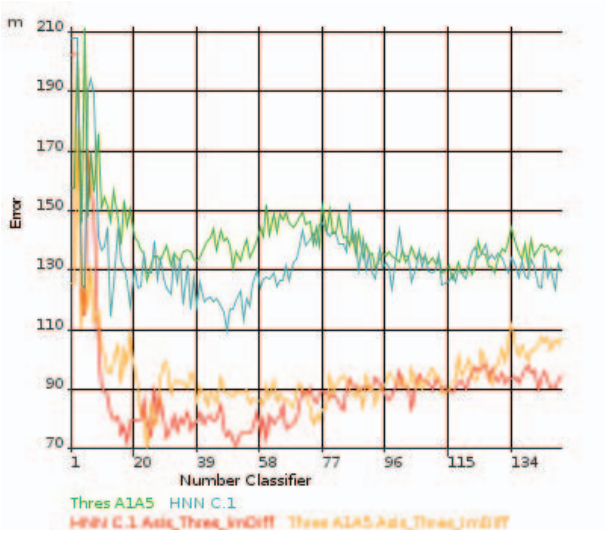

Figure 5. HNN C.1 and Threshold A.1-A.5 with different image preprocessing averages.

The red graph of figure 5 shows the performance of the average of the grey-valued image, the adaptive threshold image and a Sobel like difference image, while the blue graph shows the result for the grey-value image. Obviously the averaging of multiple input (red graph) leads to a clear improvement of the error compared to the grey-valued image (blue graph). This confirms our hypothesis. The same applies to the averaging (yellow graph) and grey-valued image (green graph) of the ensemble comprising threshold classifiers.

2) Concatenating Image Representations: However, a drawback of this solution is that we have to calculate a compensation for the Euclidean distance. The dimensionality of our HNN grows, but the "best" Euclidean distance to distinguish the positive and negative samples is calculated for the original dimensionality. For the results in table II we first choose a good compensation factor manually. Later, we have to find a way to handle this problem during the training process and one possible approach is described in section II-C2.

The first column of table II describes the combination method, voting "vote", average of the pixelvalues "average" (see II-C1) and parallel input of multiple images "concatenation" (see II-C2). The second column describes the used vision methods. ID means Image Difference, similar to a Sobel filter with different directions (see II-B), TA means adaptive thresholding and "Thres" means thresholding with a fix threshold value (see II-B). The last column names the used weak classifier and the used Haar-like feature template as depicted in figure 1.

The number in table II at "vote 0.5" respectively "vote 0.7" is a threshold that depicts how many classifiers have to have a positive result for the corresponding accumulative result. But as can be seen in table II voting is here the worst choice.

Further validation for this finding can be derived from other experiments as well, hence this method is no longer attractive for our work. That is why we concentrate on HNN as described below.

combining	DR(%)	FP	Preparation	Classifier
-	76	90	grey	HNN C.1
-	77	108	ID	HNN C.1
-	86	250	TA	HNN C.1
vote 0.5	76	106	3x Thres	HNN C.1
concate	80	102	3x Thres	HNN C.1
average	81	111	3x Thres	HNN C.1
average	82	67	Grey+ID+TA	HNN C.1
concat	81	73	Grey+ID+TA	HNN C.1
vote 0.5	93	647	4x ID	HNN C.1
vote 0.7	39	6	4x ID	HNN C.1
average	76	78	4x ID	HNN C.1
concat	85	77	4x ID	HNN C.1
average	80	53	5x ID + 3x Thres	HNN C.1
concat	84	45	5x ID + 3x Thres	HNN C.1
average	78	51	5x ID + 4x Thres	HNN C.1
concat	80	38	5x ID + 4x Thres	HNN C.1
average	82	49	5x ID + 5x Thres	HNN C.1
concat	84	56	5x ID + 5x Thres	HNN C.1
average	84	108	5x ID + 5x Thres	Thres A.1-A.5

Table II
THE TABLE SHOWS THE RESULT OF ALTERNATING THE COMBINATION METHOD AND THE VISION METHODS.

Combining	DR(%)	FP	Preparation	Classifier
-	78	115	Thres	HNN C.1
average	78	115	3x Thres	HNN C.1
concat	78	180	3x Thres	HNN C.1
concat	76	151	5x Thres	HNN C.1
concat	75	167	9x Thres	HNN C.1

Table III
THE TABLE DEPICTS THE RESULTS OF EQUALLY TRAINED HNN DUPLICATED ACCORDING TO THE AMOUNT OF CONCATENATED IMAGE FILTERS.

The Effect of HNN-Concatenation: Our HNNs are quite small. As our approach comprises parallel computation of multiple inputs, we obtain bigger HNNs as a result of concatenation. One could argue that it is quite natural to obtain better results with the increased networks due to a probable increase in detection capabilities. This assumption is not supported by our experiments. In fact, table III depicts the following: As expected, the results of averaging the image representations do not change. The result for the concatenated HNN alternate but will not be better than the average method. In the last two rows, where the HNNs were 5 times and 9 times bigger, the detection rate is lower and the false positive rate is higher compared to a single HNN. It is seen that just growing the HNN does not increase accuracy. Thus, we argue in favour of the simultaneous usage of several differently processed images as it becomes obvious that the better detection results are independent of the HNN concatenation. This result further supports our hypothesis that our parallel evaluation step in training within our system has a crucial impact on detection rates.

Useful Image Representation for Training: Training an HNN-ensemble with grey-valued images and assigning the ensemble to images that were prepared with e.g. the thresholding method reaches feasible results. But the converse procedure does not hold, i.e. using adaptive thresholding for preparing the training set and testing on grey-valued images. To prove this, we have trained an ensemble with a training set that was processed with the introduced simple adaptive threshold technique (see II-B) and executed by processing the test set with different vision methods. Table IV shows the classifier results assigned to images which are processed with different vision methods. While the HNN-ensemble reaches good results for image representations that are the same as the HNN was trained for (see table IV first row), it does not reach as flexible results considering other image representations compared to the results of the ensembles trained with grey-valued images (see I).

Our conclusion is, only changing the training set with different vision methods is not enough for increasing the accuracy.

3) Altering Adaboost: As described in section II-C4 we alter the Adaboost algorithm in the way that we select our classifier simultaneously on different image representations.

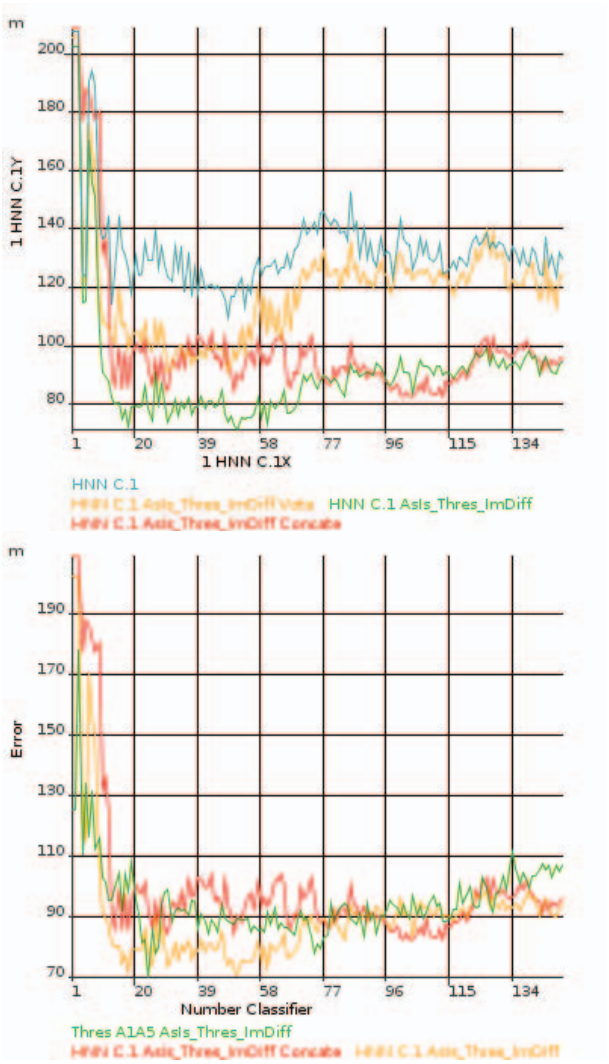

Figure 6. HNN C.1 and Threshold A.1-A.5 with different image preprocessings and combinations.

Training	DR(%)	FP	Preparation	Classifier
TA	79	64	TA	HNN C.1
TA	31	36	Grey	HNN C.1
TA	25	2	ID	HNN C.1
TA	36	42	Thres120	HNN C.1
TA	66	52	Thres80	HNN C.1
TA	58	53	Thres60	HNN C.1

Table IV
RESULT OF AN HNN. THE TRAINING SET WAS PROCESSED WITH ADAPTIVE THRESHOLDING.

The resulting HNNs differ from the HNNs that are choosen by assigning the HNN to the "normal" grey-valued training set. So, we also increase the diversity of our possible classifier set using the altered version of Adaboost. Table V depicts the starting points of different image representations for training and testing in combination with the detection

rates and the false positive rates. The modified algorithm scheme achieves detection rates within $69\% - 87\%$. An interesting aspect derived from the table is the comparison between various representations. The worst case in our experiments is testing on grey-valued images, comparing the detection rate results performed on different test sets. While the training and testing on grey-valued images in addition with image differencing and adaptive thresholding gains a rate of 87%, this combination also leads to a high false positive rate as depicted in the first row of the table. In comparison, the last row shows a detection rate of 84% but contains only approximately $1/4$ of the false positives. From our findings we assume that images processed in a more 'extreme' way than the simple grey-value images have more effect on detection accuracy and force a decrease in false positives. We want to emphasize at that point that the hypothesis needs further support and the statement is in the context of experiments with the Adaboost modification only. When we compare table II and table V for a more global view on our experiments, using grey-valued images can achieve good detection rates, for instance, 76% detection and only 90 false positives opposed to our worst results here that stem from the voting procedure.

For all trained HNNs we use the concatenation method.

Training	DR(%)	FP	Preparation	Classifier
Grey+ID+TA	87	159	Grey+ID+TA	HNN C.1
Grey+ID+TA	69	127	Grey+Grey+Grey	HNN C.1
Grey+ID+TA	83	158	Grey+TA+Grey	HNN C.1
Grey+ID+TA	81	35	ID+TA+ID	HNN C.1
Grey+ID+TA	84	44	ID+TA+ID (2)	HNN C.1

Table V
RESULT OF AN HNN. THE TRAINING SET WAS PREPROCESSED WITH ADAPTIVE THRESHOLDING AND THE RESULTING ENSEMBLE WAS ASSIGNED TO THE TEST SET COMPRISING MULTIPLE IMAGE REPRESENTATIONS.

IV. Conclusion and Future Work

In this work we have shown, that we can increase the accuracy of a classifier by simultaneous use of combined inputs from different image preprocessing steps.

First, we have shown that an averaging of the resulting images of different vision methods increases the accuracy for HNN ensembles and also for the ensemble that used the original threshold classifiers. Second, by using our described HNN concatenation, we also increase the accuracy compared to the grey-valued results and are similiar to the averaging method. Finally, we have altered the Adaboost algorithm and have shown that with these changes we can also increase the accuracy of such created ensembles which reach slightly better results compared to the other methods.

An interesting point in the context of ensemble method is that we also increase the diversity of the overall possible set of classifiers but keep the training time.

For future work, it is important to expand our results to bigger and varied test sets and also to fix the condition for learning parameters. Therefore, we will investigate how the learning process could be adapted, so that already during learning the decision of which and how many vision methods can be combined appropriately can be made.

Further, we want to investigate how the classification method could be adapted, respectively extended to choose automatically the best method and parameter during execution time (instead of the training time mentioned above). The idea is that the different parameters for the vision methods depend on the image environment in analogy to the human ability to adapt to different lighting conditions resulting in robust vision.

Acknowledgment

This work has been supported by the KSERA project funded by the European Commission under FP7 for Research and Technological Development, grant agreement n 2010- 248085, and project RobotDoc under 235065 ROBOTDOC from FP7, Marie Curie Action ITN.

References

[1] Papageorgiou, C., Oren, M., and Poggio, T.: A general framework for object detection. International Conference on Computer Vision (1998)

[2] Pisarevsky, V., Lienhart, R. and Kuranov, A.: Empirical Analysis of Detection Cascades of Boosted Classiers for Rapid Object Detection. MRL Technical Report (2002)

[3] Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, Vol.79, pp.2554-2558 (1982)

[4] Zhang, T., Tang, Y.Y., Fang, B., Shang, Z., Liu, X.: Face Recognition Under Varying Illumination Using Gradientfaces. IEEE Transactions on Image Processing (2009)

[5] Magalhaes, J., Ren, T., Cavalcanti, G.: Face Detection under Illumination Variance Using Combined AdaBoost and Gradientfaces. Intelligent Data Engineering and Automated Learning (IDEAL), pages 435-442, (2012)

[6] Viola, P. and Jones, M.: Robust Real-time Object Detection. International Journal of Computer Vision (2001)

[7] Meins, N., Wermter, S. and Weber, C.: Hybrid Ensembles Using Hopfield Neural Networks and Haar-like Features for Face Detection. International Conference on Artificial Neural Networks (ICANN) (2012)

[8] F. Crow: Summed-area tables for texture mapping. In Proceedings of SIGGRAPH, volume 18(3), pages 207-212, (1984)

[9] Wang, P., Shen, C., Barnes, N., Zheng, H.: Fast and Robust Object Detection Using Asymmetric Totally Corrective Boosting. In IEEE Transactions on Neural Networks and Learning Systems, (2012)