Hybrid Ensembles Using Hopfield Neural Networks and Haar-like Features for Face Detection

Nils Meins, Stefan Wermter, and Cornelius Weber

University of Hamburg, Department of Informatics, Knowledge Technology
Vogt-Kölln-Strasse 30, D - 22527 Hamburg, Germany
{meins,wermter,weber}@informatik.uni-hamburg.de
http://www.informatik.uni-hamburg.de/WTM

Abstract. The success of an ensemble of classifiers depends on the diversity of the underlying features. If a classifier can address more different aspects of the analyzed objects, this allows to improve an ensemble. In this paper we propose an ensemble using as classifier members a Hopfield Neural Network (HNN) that uses Haar-like features as an input template. We analyse the HNN as the only classifier type and also combine it with threshold classifiers to a hybrid neural ensemble, so that the resulting ensemble contains –as members– threshold and neural classifiers. This ensemble architecture is evaluated for the domain of face detection. We show that a HNN that uses summed pixel intensities as input for the classification has the ability to improve the performance of an ensemble.

Keywords: Ensemble; Haar-like feature; Hopfield Neural Network; Neural Ensemble; AdaBoost

1 Introduction

Our particular focus in this paper is on real-time face detection for autonomous robots and specifically on ensemble methods, because they often achieve better results than other complex classifiers. In particular the method introduced by Viola and Jones [10] reaches real-time ability. To create an ensemble of classifiers that reaches high accuracy, it is important to have a highly diverse set of different classifiers. During the ensemble creation procedure, the accuracy of the added weak classifiers mostly decreases. Mita et al. [8] addressed this and show that classifiers added in later iterations reduce the training-error, but their benefit for generalization is low. As a remedy we focus here on increasing the diversity. This could be achieved by using diverse features or different classification methods, e.g. neural network ensembles where the network architectures, the learning algorithms and so on are varied [11]. Hence, we expect to improve the generalisation ability by more accurate later chosen classifiers that display more diversity. We introduce an ensemble containing HNNs as weak classifier. The rectangle pixel sum (realized as Haar-like features) is the input vector for the HNN. To

our best knowledge, this has never been done before. Usually HNN are used for image reconstruction or a single classification task (e.g. Malanik et al. [5], which reaches good results by using a single HNN for face identification). We use Haar-like features as usual but also an extended set including HNN-based classifiers. Previously, authors have introduced new variations of Haar-like features e.g. the "center-surround feature" or diagonal features proposed by Lienhart et al. [9], [6]. Here, we profit from the ability of the HNN to handle more differentiated and sophisticated feature templates (see figure 1 B.1 to D.1). This usage of the Haar-like feature can be seen as a template-based method. In summary, our aim is to preserve the generalization ability by getting more independent (and accurate) features through using HNN for the final ensemble. Specifically, the HNNs avoid overfitting because of using rectangle sums instead of single pixels and their ability to handle noisy input.

1.1 Detection Framework

In 1989, Schapire published the first polynomial-time boosting algorithm [7]. Later Schapire and Freund published an improved version [2] and finally the AdaBoost-algorithm [3]. Motivated by the work of Papageorgiou et al. [1], Viola and Jones [10] use Haar-like features instead of pure pixels. With a normalization step, Haar-like features represent the difference of the average of pixel intensities of the considered rectangles (see figure 1).

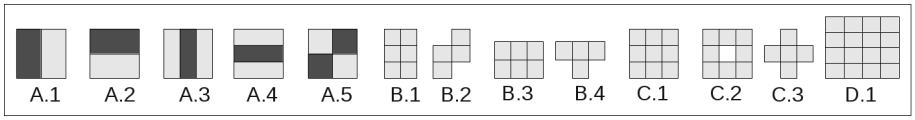

Fig. 1. To calculate the Haar-like feature value, the pixel sum of the light gray rectangle is subtracted from the pixel sum of the dark gray rectangle. Type A.1 to A.5 are the original ones from Viola and Jones [10]. Examples for the different types B to D are just used for the HNN, otherwise the feature set would exponentially increase and so would training time. The pixel sums of every small rectangle is part of the input vector.

Haar-like features can be computed very fast using the so-called integral image, which makes it possible to calculate the pixel sums in an arbitrary rectangle with just four array references. The integral image itself can be computed in just one loop. Based on such features, ensemble methods combine many classifiers to one strong classifier, the ensemble-classifier. The ensemble determines its classification result through a voting over all members (so-called "weak classifiers").

Viola and Jones employed [10] the ensemble algorithm AdaBoost and a simple threshold classifier that uses exactly one of the Haar-like features of the types A.1 to A.5, as can be seen in figure 1. The threshold classifier uses one Haar-like feature with a specific width and height and calculates its value for one position

over all training samples and determines the threshold that best seperates the negative and positive samples. In every iteration AdaBoost chooses the classifier that reaches the lowest error considering Haar-like features for all possible widths, heights and positions. With respect to the error of this classifier, AdaBoost reweights the samples of the training set in such a way that the algorithm focuses on those image samples that were mis-classified.

To increase computational speed, Viola and Jones wrapped several ensembles inside a degenerative decision tree, a so-called cascade, where one node contains one ensemble. The cascade is ordered from ensembles with low complexity (few members) to high complexity (many members) (as suggested in figure 2). If the current node classifies the considered sub-window as a face, it will be passed to the successive node. Only if the sub-window reaches the last node, it is finally classified as "face", otherwise it will be rejected as "background".

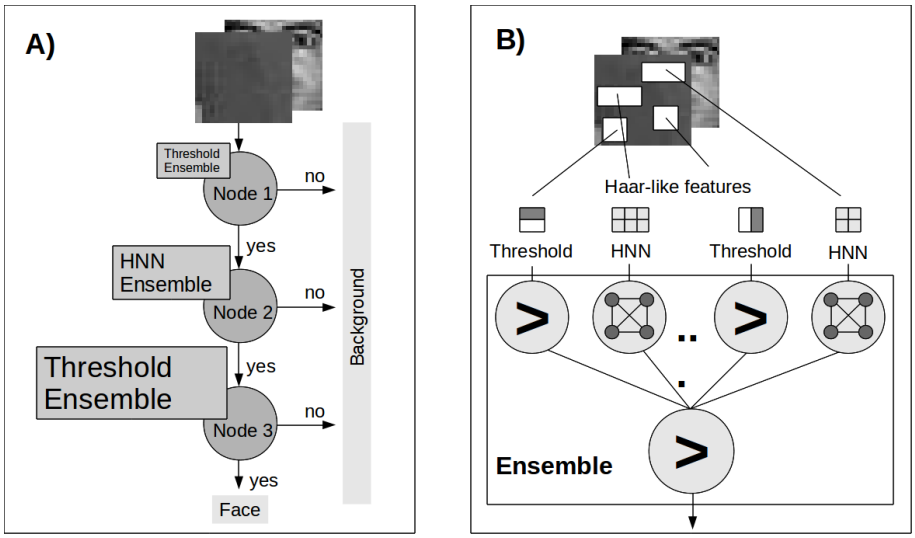

Fig. 2. Hybrid Neural Ensemble Architecture, combining threshold classifier and HNN classifier.

1.2 Hopfield Neural Network

John Hopfield [4] introduced a single-layer recurrent neural network model as an associative memory. This HNN has the ability to reconstruct a learned pattern from noisy input. If the weights of the HNN are symmetric, it can be shown that its activation pattern converges to a final stable state (see [4]). Equation 1 shows

the Hebbian learning rule for memorizing M patterns.

$$w_{ij} = \sum_{m=1}^{M} x_i^m \cdot x_j^m \text{ if } j \neq i,\ w_{ij} = 0 \text{ otherwise} \tag{1}$$

In the synchronous activation update, for every neuron the dot product of the weight vector w and the input vector x will be calculated. The result is the input for the activation function and becomes the new state of the considered neuron:

$$x'_j = \frac{1}{1 + e^{-s_j}}, \text{ where } s_j = \sum_{i=1}^{N} w_{ij}\ x_i, \tag{2}$$

where N is the number of neurons.

2 HNN Using Haar-like Features

The rectangle pixelsums of a Haar-like feature are given as an input vector to the HNN. Using the integral image and the Haar-like features instead of the pixels themselves provides the advantage of fast computation of many different scalings, which makes it possible to be scale-invariant for most purposes. Another motivation is, that this coarse grained representation could prevent our method from overfitting.

As depicted in Figure 5.A we are using the vector $(a1, a2, a3, a4)$ as input for the HNN instead of the difference of the areas $a1 + a4$ and $a2 + a3$.

2.1 Classification

We use the logistic transfer function which slightly deviates from Equation 2:

$$x'_j = \frac{2\beta}{1 + e^{-s_j}} - \beta \tag{3}$$

where s_j is calculated as in Equation 2 and $\beta = max(|p_i|)$ where p is the learned pattern. After the HNN has reached a stable state x', we compare this state with the learned pattern p. If the Euclidean distance d from the stable state to the learned pattern is less than a given distance threshold θ, it will be classified as positive:

$$1 \text{ if } d(x', p) < \theta, \quad 0 \text{ otherwise} \tag{4}$$

So, the parameters to learn are the pattern and the distance (and an adjusting offset, see Section 2.3).

2.2 Training and Testset

As training images we use the MIT cbcl database containing 2500 positive and 4500 negative samples.[1] Tests are done with a subset of the "Frontal Face Images" of the CMU Face Database.[2]

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

2.3 Learning the Pattern

Unlike many other neural networks we can train the HNN very fast, which allows to learn an ensemble of HNN classifiers in an acceptable time. First, we train the weights of the HNN using the positive samples and then - considering also the negative samples - we choose the distance threshold θ to the calculated pattern that most reduces the error. In detail we will calculate the pattern for the HNN as follows: during the calculation of the weights for the HNN according to Equation 1 we calculate the average pattern.

We adjust the input x for every positive sample with a calculated offset ϕ, so that $p_i = x_i - \phi$ for every element of x and $max(p_i) = -min(p_i)$ is fulfilled to use our HNN in a simple binary manner. The offset ϕ will be:

$$\phi = \frac{max(x_i) + min(x_i)}{2} \tag{5}$$

Our final pattern for comparison will be calculated as $p^a = average(p^m)$ where m is an index that ranges from 0 to the number of positive samples. Accordingly the final offset will be the average of the current offsets $\phi^a = average(\phi^m)$.

During Hebb-learning we take into account the weights of the training samples that are iteratively changed by AdaBoost. As mentioned in Section 1.1, AdaBoost adapts the weights of the trianing samples to swap the training set from already learned images to the mis-classified ones. Our experiments have shown that we reach best results by weightening the training patterns in Equation 1 with the AdaBoost training set weights.

3 Results

Analysing whether the HNN can increase the diversity regarding threshold classifiers, we notice that HNN and threshold classifiers produce different results by using the same exact Haar-like features. For comparison, we use the same Haar-like feature from A.1 to A.5 (figure 1) and also we analyse the performance by using more complex Haar-like features as suggested in Figure 1, types B, C and D.

All classifiers are trained with the same conditions. Our experiments show that HNN and threshold classifiers have a low correlation. There are some Haar-like features, where the HNN yields better results in comparison to threshold classifiers but also vice versa. We interpret this as a good hint of a benefit of using HNN in addition to threshold classifiers.

We consider a heterogeneous ensemble containing threshold classifiers and our HNN classifiers (see Figure 2 B)). The Figures (3) shows the performance of the different classifier methods. The classifiers were all trained using AdaBoost to create one ensemble containing a fixed number of 150 members. The figures show the performance over the number of used weak classifiers starting with one until the maximum of 150 members is reached. For all trainings we perform a random choice of 2500 classifiers for each AdaBoost training iteration. No

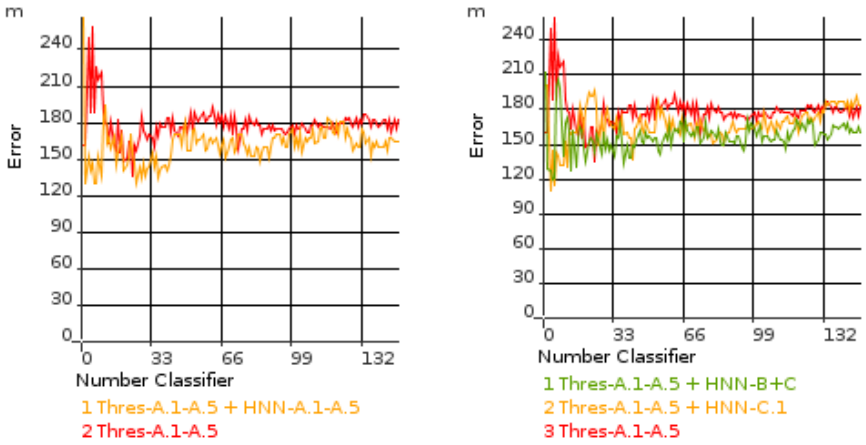

Fig. 3. Both charts show the error versus the number weak classifiers. The hybrid ensembles reaches the lower error.

variance normalization is used for the ensemble creation. The resulting ensembles are tested against a subset of the CMU test set containing 208 positive and 35.000 negative samples.

Figure 3 (left), shows the performance of one threshold ensemble and one hybrid ensemble. Both are trained with Haar-like features A.1 to A.5 as depicted in Figure 1. Instead of aggregating both classifier sets we have forced AdaBoost to alternate the two sets. After AdaBoost has chosen a threshold classifier and re-weighted the training images, it has to choose a HNN classifier and so on. We can see in the Figure 3 (left) that the hybrid ensemble reaches the lower error. The same can be seen right in Figure 3 but the HNNs there were trained with the more complex Haar-like features type B and C (Figure 1).

In Figure 4 (left) we see three ensembles, one threshold ensemble which is trained with threshold classifiers and features type A.1 to A.5 and two HNN ensembles out of which one is trained with feature type C.1 and the other is trained with feature type B and C (see Figure 1). The HNN ensembles yields a higher detection rate (but also a higher false positive rate).

Combining different weak classifiers by alternating homogenous ensembles within a cascade trained with threshold classifers on the one hand and HNN on the other hand (see figure 2 A) shows clearly an improvement concerning the false positives. As seen in figure 2 A) the combination reduces the false positives faster (with usage of less weak classifiers) than the homogenous cascades.

4 Summary and Conclusion

In this paper an ensemble that uses a HNN with Haar-like features as input template acting as a weak classifier is described. We use Hebb-learning and

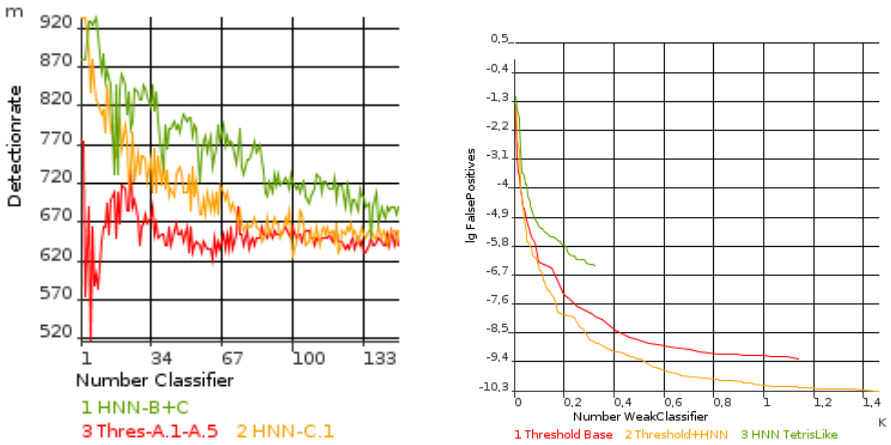

Fig. 4. Left is shown the detectionrate versus the number of ensemble members. The right chart shows the false positive versus the number of weak classifier within a cascade as depicted in figure 2 A)

adopt it to use it for the binary class problem of face detection. This ensemble was analyzed according to its performance containing HNN as the only type of a classifier member and also as a hybrid combination with threshold classifiers. While our results show just a minor improvement of detection results, they show the general possibility to use a HNN as an ensemble member. This is a novelty and brings the prospect of using the advanced abilities of an HNN to an ensemble. We have further demonstrated that this is an appropriate method for face detection. HNNs used in this manner are fast to train and to execute (the HNN mostly needs only between one and three iterations to reach a stable state). While they produce a higher false positive rate, the performance can be improved by using HNNs within a hybrid ensemble containing threshold and HNN classifiers. The HNN is promising because it has the ability of handling more input values and therefore more complex Haar-like features. Similar results are also possible by combining or using other classifiers with these features and a template-like approach e.g. normalised cross correlation. However, for future work we see the important advantage in using the ability and further potential of an HNN –as for example the multiclass ability– contrary to a more simple classifier. For example, it would be interesting to investigate multiclass HNNs using random forests in a compact form.

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 ROBOT-DOC from FP7, Marie Curie Action ITN.

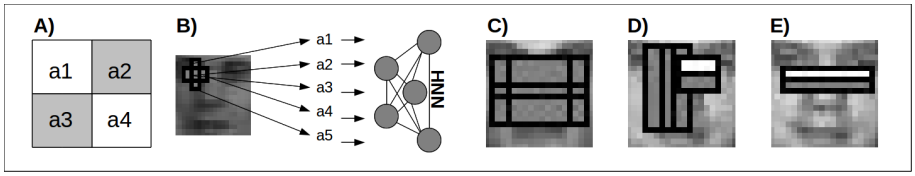

Fig. 5. Visualisation of Haar-like feature usage. A) HNN used the input vector $(a1, a2, a3, a4)$ instead of single value $(a1 + a4) - (a2 + a3)$ used by threshold classifier. B) Feature mapping to an HNN. C), D) and E) showing samples of the chosen Haar-like features. C) shows the first chosen Haar-like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

References

1. Papageorgiou, C., Oren, M., and Poggio, T.: A general framework for object detection. International Conference on Computer Vision (1998)
2. Freund, Y.: Boosting a weak learning algorithm by majority. Information and Computation, 121(2):256285 (1995)
3. Freund, Y. and Robert E. S.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119139 (1997)
4. 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)
5. Malanik, D. and Jasek, R.: Neural network face identification. Proceedings of the 11th WSEAS international conference on Applied informatics and communications (2011)
6. Lienhart, R. and Maydt, J.: An extended set of Haar-like features for rapid object detection. In Proc. of ICIP (2002)
7. Schapire, R. E.: The strength of weak learnability. Machine Learning, 5(2):197227 (1990)
8. Mita, T., Kaneko, T. and Hori, O.: Joint Haar-like features for face detection. In Proc of ICCV (2005)
9. Pisarevsky, V., Lienhart, R. and Kuranov, A.: Empirical Analysis of Detection Cascades of Boosted Classiers for Rapid Object Detection. MRL Technical Report (2002)
10. Viola, P. and Jones, M.: Robust Real-time Object Detection. International Journal of Computer Vision (2001)
11. Zhao, Y., Gao, J. and Yang, X.: A survey of neural network ensembles. International Conference on Neural Networks and Brain (2005)