Embedded Realtime Feature Fusion based on ANN, SVM and NBC

Andreas Starzacher and Bernhard Rinner
Institute of Networked and Embedded Systems
Klagenfurt University
Lakeside Park B02b, Austria.
[andreas.starzacher, bernhard.rinner]@uni-klu.ac.at

Abstract – *Artificial neural networks (ANNs), support vector machines (SVMs) and naive Bayes classifiers (NBCs) are common tools for multisensor data fusion applications. In this paper ANN, SVM and NBC are applied to embedded realtime feature fusion and compared to different algorithms concerning classification execution time as well as classification rate. These algorithms are implemented on our three-layered multisensor data fusion architecture and applied to traffic monitoring where we are focusing on fusing data originating from distributed acoustic, image and laser sensors for vehicle classification and tracking.*

The evaluation of the algorithms is performed on our embedded platform and has shown promising results concerning realtime classification execution time and classification rate.

Keywords: Realtime feature fusion, neural network, support vector machine, naive Bayes, embedded system

1 Introduction

Multisensor Data Fusion (MSDF) comprises well-known methods to combine homogeneous and heterogeneous sensor information. One of the main objectives of MSDF is to achieve significant advantages over single source sensor data in order to increase the robustness and confidence of sensor applications [1, 2, 3]. Furthermore, MSDF is used to extend spatial and temporal coverage and to reduce ambiguities and uncertainties of the sensor measurements. Especially in time-critical applications, such as traffic monitoring applications, it is crucial to be able to trust in robust estimates of the phenomenon within a certain region of interest. In literature there is a large number of various fusion algorithms ranging from estimation methods, e.g., (Kalman) filters, to different kinds of classification and inference methods, e.g. support vector machines [4], artificial neural networks [5, 6], Bayesian statistics [7, 8], Dempster-Shafer theory of evidence [9, 10].

Sensor fusion is typically performed at three levels of abstraction [11], i.e., raw-data fusion, feature fusion and high-level (decision) fusion. *Raw-data fusion* is used to combine data from different sensors which measure the same physical parameters (e.g., pixel-fusion in image processing). *Fusion at feature level* combines extracted features from the raw data of the individual sensors and finally, *high-level fusion* integrates preliminary fusion results derived from lower-level fusion. High-level fusion provides a final estimate of the observed scene, e.g., identity of an observed entity. For the sake of completeness, fusion may be also categorized into complementary, competitive and cooperative fusion. A detailed explanation of these categories can be found in [9].

There are two major architectural methodologies to MSDF. First, the fusion system consists only of a central fusion unit (centralized approach). The second approach allows sensor data to be combined locally on each sensor node. The partially fused data is then sent to a distinguished node in the sensor network to perform the final fusion (decentralized fusion approach).

In this paper, however, we focus on *embedded, realtime* MSDF within a decentralized three-layered fusion architecture [12]. The major challenges that arise here are limited resources on the embedded processing node and strict timing requirements on the fusion methods in order to keep pace with the permanent incoming data stream. Therefore, we show the feasibility of applying SVM, ANN and NBC for embedded realtime feature fusion and thus evaluate the classification performance concerning classification execution time for single feature vectors (samples) and classification rate (in terms of *true positives*) on different datasets on our embedded platform. Furthermore, we analyze the degree of scalability with increasing number of features and the maximal number of samples that can be classified per second. In [12] we have already shown that linear and quadratic discriminant analysis (LDA and QDA, respectively) are suitable for embedded feature fusion under realtime constraints. Standard k-nearest neighbor was rejected due to its unacceptable time needed for classification, i.e., not satisfying our realtime constraints especially if large numbers of features are being processed. Therefore, this paper also extends the evaluation of our previous work by comparing the classification time as well as classification rate of discriminant

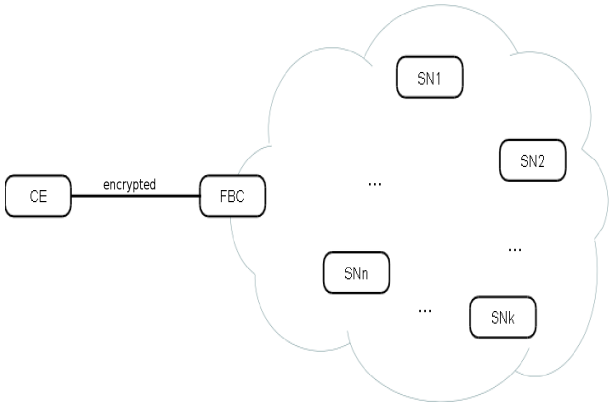

Figure 1: Overview of the sensor network structure

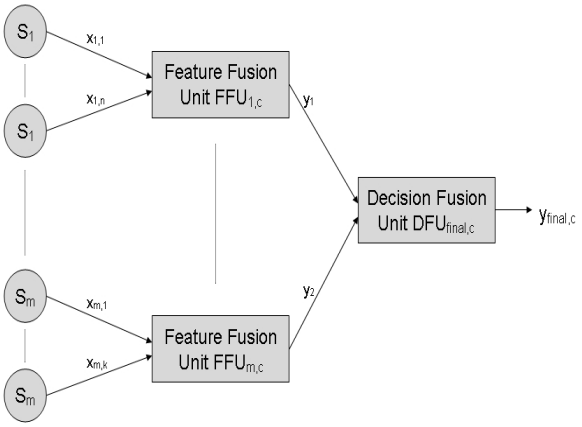

Figure 2: Feature and decision fusion with reference to fusion cluster c (PFC), where S_i represents sensor node i, $x_{i,j}$ the j^{th} feature value of sensor i and y_i the i^{th} local partial decision generated by FFU_i.

analysis with a support vector machine and an artificial neural network as well as a naive Bayes approach implementation on our embedded platform.

The remainder of the paper is as follows: Section 2 briefly describes our sensor network structure and three-layered multisensor data fusion architecture approach. Section 3 describes the selected algorithms applied to embedded feature fusion and the features used in our application scenario, i.e., traffic monitoring. Experimental results of ANN, SVM and NBC in comparison to LDA and QDA on different datasets are presented in section 4. The evaluation is done exclusively on our embedded platform. A summary and an outlook conclude this paper.

2 Systems Architecture

Our sensor network structure is shown in Fig. 1. It consists of the following components: several sensor nodes (SN_i), a single center node (CE) and a so-called "fusion backbone to center" node (FBC) which is a dedicated sensor node sending the final decision to the CE. Moreover, the FBC is responsible for the final fusion processing, i.e., deriving the final decision about a specific task which in our case is object classification and tracking information.

Each of the different sensor nodes has several attributes and methods that define its state and functionality. Attributes are for example an ID for unique identification, internal description and timestamps (logging functionality). SN methods are e.g., send, receive, fusion methods and methods interfacing external devices (e.g., stop lights in traffic monitoring environments).

In our sensor network the sensor nodes are organized in clusters (PFC_i, see beneath) performing different fusion tasks on specific regions of interest (ROIs) such as intersection areas, urban roads, etc. The assignment of sensor nodes to specific clusters is done a-priori considering several constraints (e.g., vicinity, adjacency, ROIs, etc.).

Our MSDF architecture consists of three layers [12]. We chose a layered architecture approach in order to easily abstract and encapsulate the various processing steps into independently working processing units. Each fusion layer performs a specific task in order to contribute to the whole fusion process. In brief, layer 1 is responsible for recording and if necessary normalizing raw sensor readings (data acquisition and alignment). Layer 2 performs intra-cluster fusion and finally layer 3 fuses the decisions from the second layer to generate the final decision (inter-cluster fusion). A more detailed description of our MSDF architecture and its different layers is given in our previous work [12]. In literature a lot of different fusion modeling approaches exist [13, 11, 14, 15]. The essential objective of our approach is to keep the overall structure light-weighted (embedded design) and to develop autonomous modules (layers) which can be developed separately and substituted if required (module-based approach).

The sensor readings acquired by layer 1 are processed (fused) in layer 2. After the data fusion in layer 2 is done, layer 3 creates the final decision. Layer 2 and 3 make use of two fusion methodologies to perform the fusion tasks (as depicted in Fig. 2 and Fig. 3) depending on the internal sensor network structure. Fig. 2 shows the combination of feature and decision fusion. We apply this approach to large sensor networks where the partial fusion clusters (*PFCs*) consist of lots of sensor nodes (SNs) and the overall bandwidth between (adjacent) sensors is low or costly (sending decisions costs much less than sending lots of features).

Let, for example, the number of sensor nodes within a PFC c in a sensor network be n (within layer 2). These n SNs locally extract features and perform feature fusion resulting in local partial decisions (y_i). These y_i are sent to a chosen SN h (SN head) which performs local partial decision fusion within PFC c resulting in $y_{final,c}$. Thus, layer 2 has to send n final partial decisions to layer 3 which is then performing decision fusion. If the network bandwidth between SNs in a PFC c is high, then another approach de-

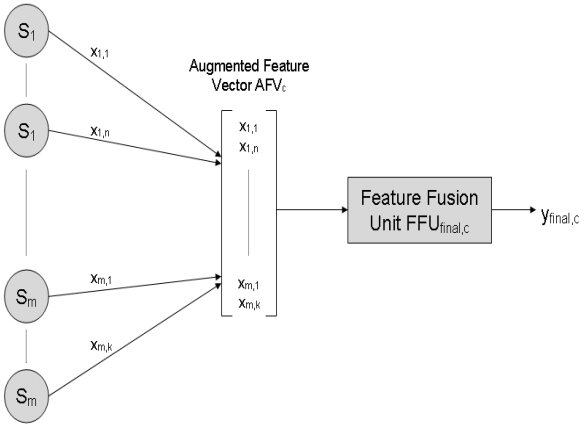

Figure 3: Feature fusion based on augmented feature vector with reference to fusion cluster c (PFC), where S_i and $x_{i,j}$ are the same as in Fig. 2 but with a single FFU_{final} combining all features of AFV.

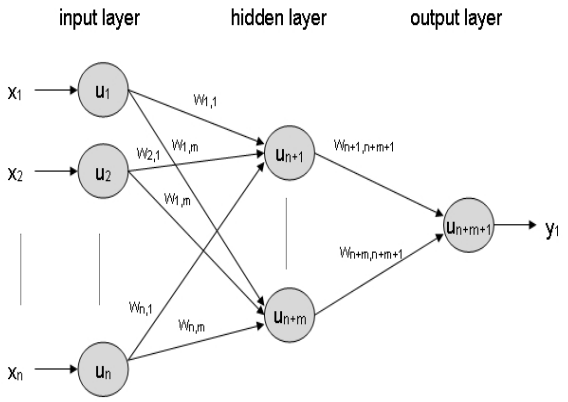

Figure 4: A three-layered feed-forward ANN representing a FFU_i with respect to cluster c

picted in Fig. 3 is used. After receiving data from layer 1, one of the sensor nodes is selected as head h (within layer 2) and each SN within the same PFC sends its features (homogeneous and heterogeneous) to h. Finally, the SN head h performs feature fusion based on the augmented feature vector (AFV_c) without using a decision fusion unit $\text{DFU}_{final,c}$. In both cases the output of the two approaches is a final partial decision ($y_{final,c}$). The n final partial decisions of each PFC are sent to layer 3 where high-level fusion results into a final decision.

3 Feature Fusion Algorithms

To perform embedded feature fusion under realtime constraints we decided to implement a layered artificial neural network, a support vector machine and a naive Bayes approach. Even though some parameters of artificial neural networks are still based on heuristics, e.g., choosing the number of hidden units and neurons, ANNs are tools often applied to MSDF problems [5, 6, 16, 17]. The general structure of an ANN is shown in Fig. 4.

Furthermore, support vector machines are a widely used classification technique and applied to sensor fusion applications, too [15, 18, 19, 20]. A basic geometrical representation of the SVM problem formulation is given in Fig. 5. The basic idea (challenge) behind SVMs is to obtain an optimal margin by minimizing $||w||$ in order to maximize $2\,||w||^{-1}$, i.e., the margin width.

Naive Bayes classifiers, the simplest kind of Bayesian networks [21], are also a powerful tool for feature-based decision modeling. Under the fundamental assumption of conditional independence of the features, it is possible to perform the fusion task efficiently with respect to processing power and memory consumption.

The following three sections give a description of the algorithms that are used for embedded realtime feature fusion and the evaluation (see section 4). Section 3.4 gives an overview of the laser-based, acoustic and visual features that we are using in our application scenario in order to classify vehicles.

3.1 Three-layered Neural Network

Instead of using a simple perceptron, we chose to implement a feed-forward artificial neural network model (multilayer perceptron) for embedded feature fusion, because in general our data is not linearly separable. Due to the property of universal approximation [22], we implemented a three-layered perceptron consisting of an input, a hidden and an output layer (three layers are sufficient for approximating any function in an arbitrary accuracy). The number of input neurons results from the number of features originating either from a single sensor node SN_i or the augmented feature vector AFV_c. There is a single output neuron defining the (partial) decision. The number of hidden neurons is specified by using the following heuristics: Let N be the number of input and M the number of output neurons. The number of hidden layers H is defined as the sum of number of input and output neurons multiplied by a constant d with $d \in (0, 1]$ (eq. 1). We evaluated the neural network with several $d \in \{2/3, 1/2, 1\}$ (see section 4).

$$H = \lfloor (N + M) \cdot d \rfloor \tag{1}$$

In Fig. 4 the three-layered feed-forward neural network represents one of the feature fusion units FFU_i with respect to cluster c (within layer 2 of our fusion architecture). The output of the FFU_i is either y_i or $y_{final,c}$ (as depicted in Fig. 2 and 3). After each cluster has computed its own partial decision $y_{final,c}$, the high-level fusion process takes place in layer 3 of our MSDF architecture.

In general, training an artificial neural network is very time-consuming depending on several parameters like number of layers, neurons, epoches, type of activation functions and the training algorithm (backpropagation and its en-

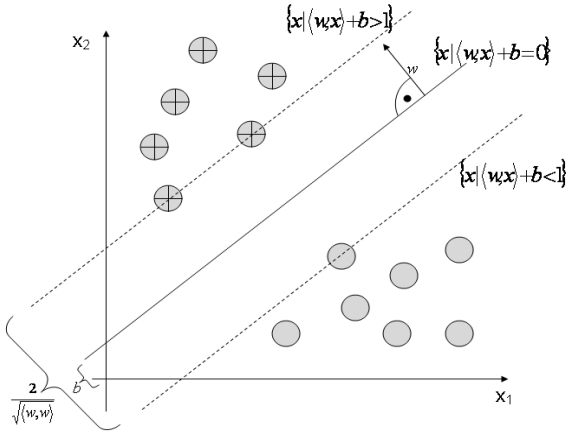

Figure 5: SVM problem formulation: maximizing margin is equivalent to minimizing $||w||$ ($\max$ margin $\Leftrightarrow \min ||w||$)

hancements [23, 24]). In order to meet realtime constraints and assure realtime fusion the training is done on an external computer and only trained ANN models are used on the embedded platform to perform the fusion task.

3.2 Support Vector Machines

In the experiments we use a standard support vector machine implemented on our embedded platform. Fig. 5 shows the standard SVM problem geometrically. Generally, a SVM is a so-called *maximum margin* classifier. The objective of the SVM optimization problem is to obtain certain parameters in order to define a separating hyperplane that has an optimal class separability (optimal in terms of maximum margin that is defined by the support vectors). In real-world scenarios it often happens that features are close to the hyperplane or cannot be separated properly. Therefore so-called *slack* variables are introduced (*soft* side constraints). These allow for a certain amount of misclassified features. Especially in our case of fusing acoustic features the introduction of slack variables turned out to be absolutely essential due to the high sensitivity to noise in the acoustic sensor readings.

If the data is not linearly separable, the so-called *kernel-trick* comes into play. There are different kinds of kernels such as polynomial kernel, (Gaussian) radial basis and sigmoid functions. Kernels transform the original data into a higher dimensional feature space. Even if the original data are nonlinear, the transformed data is separable by a hyperplane in feature space.

3.3 Naive Bayes Approach

In addition to a neural network and support vector machine, we implemented a naive Bayes approach which has a lot of advantages. One of these are its simplicity and transparency. Assuming conditional independence of the feature values x_i within a feature vector $\mathbf{x}$ the probability of $\mathbf{x}$ given a certain group g_i is the result of multiplying the probabilities of each x_i in $\mathbf{x}$ (eq. 2).

$$p(\mathbf{x}|g_i) = \prod_k p(x_k|g_i) \tag{2}$$

Thus, the posterior probability is proportional to the product of the prior of group g_i and the independent likelihoods as shown in eq. 3.

$$p(g_i|\mathbf{x}) \propto p(g_i) \prod_k p(x_k|g_i) \tag{3}$$

Under our assumption, each of the likelihood functions for each group g_i follows a Gaussian distribution with $\mu_i = \hat{\mu}_i$ and $\sigma_i^2 = \hat{\sigma_i}^2$ ($\hat{\mu}_i$ and $\hat{\sigma_i}^2$ are the standard estimates of the mean and variance of the group g_i).

By computing the posterior for each group g_i, we can apply the *argumentum maximi* rule to infer a final decision $\hat{D}(\mathbf{x})$, i.e., a *maximum-a-posteriori*-estimate (*MAP*, eq. 4).

$$\hat{D}(\mathbf{x}) = \arg\max_i p(g_i|\mathbf{x}) \tag{4}$$

If lots of (small) probabilites are multiplied, the final (posterior) probability is enormously small and often converges to zero. In order to get around this problem, instead of taking probabilities, we take the logarithm of probabilities (*log-prob* modeling). This is possible due to the monotonic characteristic of the log function. In addition to a *MAP* decision rule, we implemented a *Maximum Likelihood* (ML) rule to evaluate the importance and influence of prior probabilities used in *MAP*.

In many applications the conditional independence assumption does not hold, because the features originating from different sensor nodes often correlate, e.g., if the same ROI is observed. But it is worth mentioning that even though in general the independence assumption does not hold, naive Bayes feature fusion gives reasonable fusion results [25]. Therefore, it can indeed compete with more sofisticated algorithms such as ANN and SVM if the correlation among features is not too strong.

Furthermore, especially in case of embedded fusion applications it is computationally too expensive to infer posteriors without assuming conditional independence, i.e., computing the joint probability models with all successive conditional probabilities (these conditional probabilities are often difficult to obtain).

3.4 Heterogeneous Features

Currently, we have implemented eight acoustic feature extraction algorithms in time as well as frequency domain. These algorithms extract 14 acoustic features. The acoustic features in time domain are computationally affordable. Time domain features are short time energy and zero-crossing rate. Spectral features are spectral centroid, bandwidth, flux, roll-off point and band energy ratio. Furthermore, cepstral coefficients extracted by cepstral analysis are used as feature values. As we are working on embedded realtime MSDF algorithms, we cannot implement arbitrarily

complex feature extraction algorithms that might extract features that have a higher degree of class separability. Therefore, we base upon light-weighted acoustic feature fusion algorithms that can be applied to realtime fusion applications. In order to enhance the classification results (presented in section 4.2) we perform feature fusion based on acoustic features (1) and decision fusion (2) based on results from (1) and decisions originating from visual feature classification. Our preliminary work concerning the evaluation of classification performance based on visual features on our embedded platform is found in [26]. Features originating from laser sensors might be height values (for altitude profile generation) and velocity. At the moment the laser sensors primarily function as triggers within our traffic monitoring scenario.

4 Evaluation

With the evaluation presented in this section we discuss several performance tests and results. The focus of our experiments is to show that ANN, SVM and NBC are efficient algorithms for embedded realtime feature fusion. The evaluation of those algorithms has shown very promising results with respect to realtime classification execution time and classification rate (true positives), i.e., each algorithm can be used in our practical system. In order to validate our results, we compared them with linear and quadratic discriminant analysis algorithms that we have already implemented and evaluated in our previous work [12].

In the following section we describe our embedded test platform on which we performed the tests with simulated data (evaluation of scalability concerning number of features), real-world test datasets that are freely available online (performance comparison of the algorithms) and with a dataset containing acoustic features (evaluating acoustic features that are extracted from several audio samples by using our acoustic feature extraction algorithms).

4.1 Embedded Test Platform

Our test and evaluation platform is a MICROSPACE EBX (*MSEBX945*) embedded computer board from *DigitalLogic AG* that serves as a MSDF platform (see [12]). Its compact EBX single-board construction (146mm $\times$ 203mm) has several interfaces such as RS-232 (for laser sensors), LAN 100MB (for networking), FireWire IEEE 1394 over MiniPCI (for interfacing cameras) and USB (serving as a soundcard interface). The CPU module, a *SMX945-L7400*, makes use of Intel Core 2 Duo technology with 2 $\times$ 1500 MHz and a 667 MHz FSB.

Some of the main characteristics of the board are 2048MB DRAM, USB 2.0, RS232C, COM-Interface, 10/100BASE-T, 1GB-LAN PCIe, MiniPCI slot and PS/2 interface. The *MSEBX945* board has a total power consumption of about 12 to 15 W. The operating system is a Linux derivate called *Linux from Scratch* (LFS) provided by our industrial partner.

We successfully interfaced the following sensors: *Noptel CM3-30* (single-beam laser for distance measurements and altitude profile generation), acoustic sensors (for mono/stereo audio recordings), *Baumer FWX14-K08* cameras (for single shots and continuous recordings) and *ELV ST-2232* (an environmental sensor for recoring light intensity, temperature and sound level) for additional environmental information about specific regions of interest (currently not used within the fusion task).

The communication with the embedded platform is primarily done via remote interface.

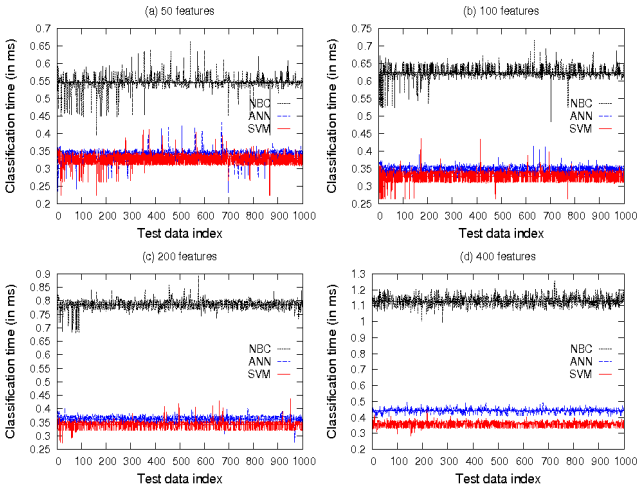

Figure 6: Classification time for increasing number of features with datasets of 1000 samples

4.2 Performance Evaluation

As already mentioned, the main objective of the evaluation is to present results concerning classification execution time (in milliseconds) as well as classification rates (in %) of our embedded feature fusion algorithms. The evaluation of the classification execution time is performed on simulated datasets and the analyses of the classification rates are done on four different real-world datasets.

First, we measured the time needed for executing the algorithms to classify a single feature vector $\mathbf{x}$ (i.e., to fuse the features) consisting of different numbers of simulated features (scalability). The number of measurements is 1000 (constant). The results are shown in Fig. 6. Even in case of a feature vector containing 400 features, the mean classification time spent is quite short, hence there are plausible reasons to apply SVM, ANN and NBC to embedded realtime MSDF problems. As can be seen in Fig. 6, SVM outperforms ANN and NBC. Furthermore, an interesting observation particularly with regard to scalability is that the mean classification time of SVM lies in a quite constant classification time interval concerning the different numbers of features of a feature vector $\mathbf{x}$. It lies in the open interval $(0.3, 0.4)$ in units of ms. This is not the case with the neural network and naive Bayes approach and neither with LDA nor QDA. The mean classification time difference of SVM and ANN is rather small whereas the difference between SVM (ANN) and NBC is significantly larger. For example,

Table 1: Classification Time (in ms)

Algorithm	t_μ	t_σ	t_{total}
SVM	0.35893	0.2344	366.23
ANN	0.442294	0.0844162	444.894
NBC	1.12886	0.355382	1153.96
LDA	37.8508	2.22199	37850.8
QDA	28.8196	1.25372	28819.6

if the number of features of **x** is set to 400, the difference is about $0.769ms$ ($0.686ms$). This difference is obviously reflected in the number of classification tasks per second (see Fig. 7).

SVM, ANN and NBC have a much shorter mean classification time t_μ, smaller standard deviation t_σ as well as total classification time t_{total} than LDA and QDA, respectively. This can be seen in Table 1 where the feature vector **x** consists of 400 features and the number of samples processed is set constant to 1000.

If the total CPU performance is used exclusively for the classification task, a huge amount of feature vectors can be classified per second. In Fig. 7 the clustered histogram shows the average number of feature vectors (samples) that can be classified within one second of time (n_{max}). Additionally, the results show that our algorithm implementations are *light-weighted* and hence definitively suitable for and applicable to embedded MSDF). Again, SVM, ANN and NBC extremely outperform the discriminant analysis approach concerning n_{max}.

Second, we evaluated the classification rates of the individual algorithms for four different datasets. The characteristics of the datasets are found in Table 2.

Table 2: Datasets

Number	Dataset	#(train)	#(test)	#(features)
1	australian	681	9	14
2	svmguide1	3089	31	4
3	heart	258	12	13
4	own dataset	444	17	14

The tests are structured as follows: for each dataset one parameter setting (session) for both LDA and QDA, two sessions for SVM and NBC and three sessions for ANN are applied to obtain the classification rates under specific parameter settings. We have unchanged (default) as well as altering parameters for each algorithm. The default parameter settings for SVM, ANN and NBC are summarized in Table 3, whereas LDA and QDA are executed with default parameters specified in [12].

For SVM we evaluated the algorithm with two different kernel types. We chose to take a sigmoidal kernel and a kernel based on a radial basis function (rbf). In our case we applied a Gaussian rbf. As the datasets, especially the one containing acoustic features, are not linearly separable, we neglected to apply a linear kernel.

The three-layered ANN is evaluated with different numbers of hidden neurons (altering d of eq. 1). It is obvious

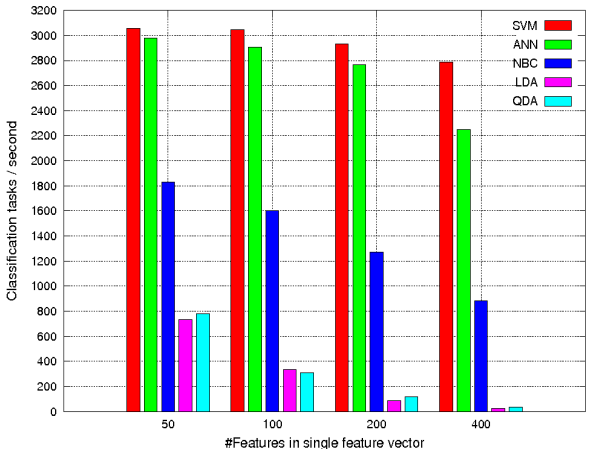

Figure 7: Maximal number of samples that can be classified per second

that the number of input neurons results from the number of features in a feature vector **x**. As we model the fusion task based on features as a classification model, where the desired output is a single class label of high confidence, the number of output neurons is one (the estimated class label). As described in section 3.1 we apply several heuristics to obtain a robust number of hidden neurons within the hidden layer of the neural network. The heuristics we used depend on the single parameter d (see eq. 1). Some tests showed that it is also reasonable to choose a slightly, but decisive modified version of eq. 1, i.e., $H = \lfloor (N + M) \cdot d \rfloor - 1$. We applied this rule to dataset 4 in an additional session and gained an increased true positives rate of 76.47%.

Table 3: Parameter Settings (unchanged)

Algorithm	Unchanged Parameters
SVM	Degree in kernel function $deg = 3$ Cost parameter $C = 1$ Tolerance of termination criterion $\gamma = 0.001$
ANN	#(Input Neurons) = #(Features) #(OutputNeurons) $= 1$ #(Neural Network Layers) $= 3$ #(Training Epoches) $= 10^5$ *Gaussian* Activation function
NBC	*Priors* estimated from training data *Means* estimated from training data *Variances* estimated from training data *Gaussian* distribution assumed

Concerning NBC we substituted the classification rule with log Maximum Likelihood (*LogML*) and log Maximum a posteriori (*LogMAP*) likewise. Using prior probabilities does not necessarily imply an increase of the classification rate (as it is with datasets 1 and 3). With dataset 2 (by 0.8%) and 4 it does incease, especially with dataset 4 (by 23.5294%). The parameters *priors*, *means* and *variances* are estimated from the training datasets. The underlying distribution for each class in the datasets are assumed to be

Gaussian defined by the estimated class parameters.

A summary of the altered parameters within the sessions is depicted in Table 4. The minus sign (-) indicates that there is no parameter setting in the appropriate session.

Table 4: Parameter Settings (sessions)

Algorithm	Session 1	Session 2	Session 3
SVM	sigmoid kernel	rbf kernel	-
ANN	$d = 2/3$	$d = 1/2$	$d = 1$
NBC	*LogML* rule	*LogMAP* rule	-

Fig. 8 shows a clustered histogram with the appropriate classification rates of each algorithm according to the sessions (for all datasets). A zero classification rate (no bar for a certain session) in the histogram indicates that there is no parameter setting in this session (noted as a - in Table 4). In Fig. 8(a) SVM can be significantly improved by applying a suitable kernel for dataset 1. By applying a rbf kernel, the classification rate increases from 22.22% to a 77.78% percentage. With ANN there is also an increase observed by changing parameters. A 100% classification rate is achieved by using $d = 1$. This shows the influence of d in choosing a reasonable number of hidden neurons. With dataset 1 priors have no influence on the classification rates (same rates with *LogML* and *LogMAP*). With dataset 1 SVM is outperformed by the all other algorithms. In Fig. 8(b) SVM and ANN give the highest classification rates. SVM again implements a rbf kernel and ANN uses $d = 1$ (also $d = 2/3$ and $d = 1/2$ give rates above 95%). LDA and QDA are outperformed with rates under 90%. In subfigure (c) SVM reaches a 91.67% rate with a rbf kernel. The other algorithms give rates under 90% for each session. Dataset 4 is our created dataset containing 14 acoustic features per sample (Fig. 8(d)). These acoustic features are extracted by means of our acoustic feature extraction algorithms. Tests give classification rates under 80% for each algorithm (highest rate 76.47%). As described in section 3.4 feature and decision fusion with heterogeneous features and decisions are to be applied in order to increase the classification rates.

5 Conclusion

In this paper we evaluated the applicability of a support vector machine, three-layered artificial neural network and a naive Bayes approach to embedded realtime feature fusion. The algorithms are implemented on our three-layered multisensor fusion architecture and evaluated concerning classification execution time and classification rate (true positives rate). The evaluation is exclusively performed on our embedded test platform. It includes determining the classification execution time for a single feature vector, classification rate based on different datasets, degree of scalability with increasing number of features and the maximal number of feature vectors that can be classified per second. Moreover, SVM, ANN and NBC are compared with linear and quadratic discriminant analysis approaches concerning those evaluation items addressed in this paper.

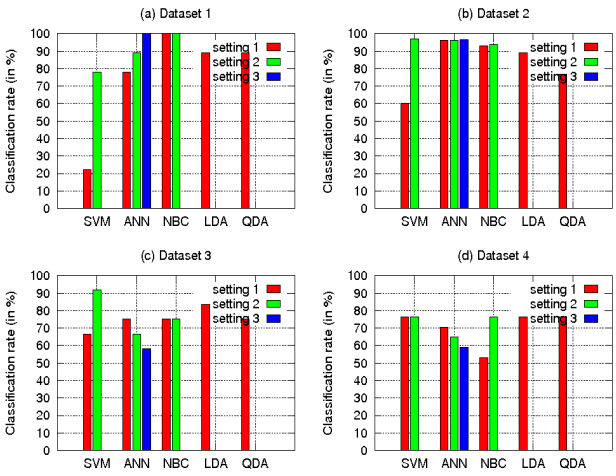

Figure 8: Classification rates of each fusion algorithm

In conclusion, the algorithms presented are applied to embedded realtime feature fusion in our practical system. Based on these algorithms, our next steps are towards extending our feature fusion with high-level fusion within the third layer of our fusion architecture. Using additional high-level fusion we want to exploit the partial feature fusion results (i.e., the partial decisions) of the individual algorithms by fusing those partial decisions in order to improve the classification results in terms of less false positives.

Acknowledgment

A grant from the Austrian Research Promotion Agency supported this work (under the FIT-IT[visual computing] grant 813399).

References

[1] H. Ruser and F. Puente León. Informationsfusion - Eine Übersicht. *tm - Technische Messen, Oldenburg*, 74(3):093–102, 2007.

[2] H. Hu and J. Q. Gan. Sensors and Data Fusion Algorithms in Mobile Robotics. *Technical Report: CSM-422, Department of Computer Science, University of Essex, Colchester CO4 3SQ, United Kingdom*, 2005.

[3] D. Loebis, R. Sutton, and J. Chudley. Review of multisensor data fusion techniques and their application to autonomous underwater vehicle navigation. *Journal of Marine Engineering and Technology*, 1:3–14, 2002.

[4] Andreas Klausner, Christian Leistner, Allan. Tengg, and B. Rinner. An audio-visual sensor fusion approach for feature based vehicle identification. In *Proceedings of the 2007 IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS 2007)*, page 6, London, UK, 2007.

[5] N. Yadaiah, Lakshman Singh, Raju S. Bapi, V. Seshagiri Rao, B. L. Deekshatulu, and Atul Negi. Multisensor Data Fusion Using Neural Networks. In *International Joint Conference on Neural Networks*, pages 875–881, Vancouver, BC, Canada, 2006.

[6] Quan Liu and Xiaomei Zhang. An adaptive multisensor data fusion system based on wavelet denoising and neural network. *Sensors for Harsh Environments II*, 5998, 2005.

[7] S.K. Jayaweera. Optimal Bayesian data fusion and low-complexity approximations for distributed DS-CDMA wireless sensor networks in Rayleigh fading. In *International Conference on Intelligent Sensing and Information Processing*, pages 19–24, 2005.

[8] G. Wang, D. Zhang, and H. Zhao. An improved Bayes fusion algorithm with the Parzen window method. In *Proceedings of the Fifth International Conference on Information Fusion*, pages 651–657, Annapolis, MD, USA, 2002.

[9] K. Faceli, A.C.P.L.F. de Cavalho, and S.O. Rezende. Combining intelligent techniques for sensor fusion. *Applied Intelligence*, 20(3):199–213, 2004.

[10] D. Fasbender, V. Obsomer, J. Radoux, P. Bogaert, and P. Defourny. Bayesian data fusion: spatial and temporal applications. In *MultiTemp2007*, pages 001–006, Provinciehuis Leuven, Belgium, 2007.

[11] J. Llinas and D.L. Hall. An introduction to multisensor data fusion. In *ISCAS-2008*, pages 537–540, Monterey, CA, 1998.

[12] Andreas Starzacher and Bernhard Rinner. Evaluating KNN, LDA and QDA Classification for embedded online Feature Fusion. In *Proceedings of the Fourth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP-2008)*, pages 85–90, Sydney, NSW, 2008.

[13] J. Esteban, A. Starr, R. Willetts, P. Hannah, and P. Bryanston-Cross. A review of data fusion models and architectures: towards engineering guidelines. *Neural Computing and Applications*, 14(4):273–281, 2005.

[14] M. Bedworth and J. O'Brien. The omnibus model: a new model of data fusion? *IEEE Aerospace and Electronic Systems Magazine*, 15:030–036, 2000.

[15] Andreas Klausner, Allan Tengg, and Bernhard Rinner. Distributed multilevel data fusion for networked embedded systems. *IEEE Journal on Selected Topics in Signal Processing*, 2(4):538–555, 2008.

[16] A. Kostrzewski, Dai Hyun Kim, Jeongdal Kim, T. Jannson, and G. Savant. Fuzzified neural network for similar/dissimilar sensor fusion. In *IEEE International Conference on Neural Networks*, pages 938–942, Washington, DC, USA, 1996.

[17] Joris W. M. Van Dam, Ben J. A. Kr, and Franciscus C. A. Groen. Neural network applications in sensor fusion for an autonomous mobile robot. In *Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics*, pages 263–278, 1995.

[18] Xiaowu Liu, Huiqiang Wang, Jibo Lai, Ying Liang, and Chunmei Yang. Multiclass Support Vector Machines Theory and Its Data Fusion Application in Network Security Situation Awareness. In *International Conference on Wireless Communications, Networking and Mobile Computing (WiCom2007)*, pages 6349–6352, Shanghai, P.R. China, 2007.

[19] Jerome J. Braun. Sensor data fusion with Support Vector machine techniques. In *Proceedings of Sensor Fusion: Architectures, Algorithms, and Applications VI*, pages 98–109, Orlando, FL, USA, 2002.

[20] S. Challa, M. Palaniswarni, and A. Shilton. Distributed data fusion using support vector machines. In *Proceedings of the Fifth International Conference on Information Fusion*, pages 881–885, Annapolis, USA, 2002.

[21] Dan Geiger, Moises Goldszmidt, G. Provan, P. Langley, and P. Smyth. Bayesian Network Classifiers. In *Machine Learning*, pages 131–163, 1997.

[22] Domonkos Tikk, László T. Kóczy, and Tamás D. Gedeon. A survey on universal approximation and its limits in soft computing techniques. *International Journal of Approximate Reasoning 33 (2003)*, 2002.

[23] Yann LeCun, Leon Buttou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. *Neural networks: Tricks of the trade*, 1998.

[24] Hui Hui, Dayou Liu, and Yafei Wang. Sequential Back-Propagation. *Journal of Computer Science and Technology)*, 9, 1994.

[25] Jan-Nikolas Sulzmann, Johannes Fürnkranz, and Eyke Hüllermeier. On Pairwise Naive Bayes Classifiers. In *Proceedings of the 18th European Conference on Machine Learning*, pages 371–381, Warsaw, Poland, 2007.

[26] Christian Leistner, Peter M. Roth, Helmut Grabner, Horst Bischof, Andreas Starzacher, and Bernhard Rinner. Visual On-line Learning in Distributed Camera Networks. In *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC-08)*, pages 1–10, Stanford, USA, 2008.