Universität Ulm
Fakultät für Ingenieurwissenschaften
und Informatik
Institut für Neuroinformatik
Direktor: Prof. Dr. Günther Palm

Feature Selection and Information Fusion in Hierarchical Neural Networks for Iterative 3D-Object Recognition

Dissertation zur Erlangung des Doktorgrades
Doktor der Naturwissenschaften (Dr. rer. nat.)
der Fakultät für Ingenieurwissenschaften und Informatik
der Universität Ulm

vorgelegt von
Rebecca Fay
aus Lich

Ulm 2007

Amtierender Dekan der Fakultät für Ingenieurwissenschaften und Informatik:
Prof. Dr. Helmut Partsch

Gutachter: Prof. Dr. Günther Palm
Gutachter: Prof. Dr. Heiko Neumann
Gutachter: Prof. Dr. Friedrich W. von Henke

Tag der Promotion: 20. Juli 2007

to my family

Zusammenfassung

Die zuverlässige Erkennung von dreidimensionalen Objekten anhand von zweidimensionalen Kamerabildern stellt immer noch ein großes Problem der Bildverarbeitung sowie der künstlichen Intelligenz dar. Hierfür existiert eine Vielzahl von Ansätzen, die jedoch im Allgemeinen weder die grundlegende hierarchische Natur von Klassifikationsproblemen ausnutzen noch das Hinzulernen neuer Objekte während der Erkundungsphase ermöglichen.

Bei Objekterkennungsaufgaben handelt es sich im Allgemeinen um Multi-Klassenprobleme, bei der die Anzahl der Klassen deutlich größer als zwei ist. Diese Arbeit bietet einen neuen Ansatz für die Lösung derartiger Probleme, indem die komplexe Aufgabe in mehrere hierarchisch angeordnete Teilprobleme zerlegt wird, die einfacher zu lösen sind.

Die vorliegende Arbeit behandelt mehrere Aspekte der Objekterkennung mit hierarchischen neuronalen Netzen:

1. Generierung geeigneter Klassifikatorhierarchien
2. Auswertung dieser Hierarchien (Informationsfusion)
3. Auswahl geeigneter Merkmale für die Klassifikation
4. Erweiterbarkeit / Adaptivität der Hierarchien
5. Generierung ähnlichkeitserhaltender spärlicher Binarcodes

Die Auswahl geeigneter Merkmalstypen für die Klassifikation von Objekten unterschiedlicher Domänen ist ein wichtiger Aspekt bei der 3D-Objekterkennung. Im Rahmen dieser Arbeit wurden unterschiedlichste Merkmalstypen verwendet und bewertet.

Für die Auswertung der Klassifikatorhierarchien wurden verschiedene Ansätze entwickelt und evaluiert. Als die vielversprechendsten Ansätze erwiesen sich

die Evaluation analog zu Entscheidungsbäumen, die Evaluation mit Hilfe der Dempster-Shafer Evidenztheorie sowie die Evaluation mit Hilfe von ähnlichkeitserhaltenden Codes bzw. Inter-State Decision Templates.

Die Generierung ähnlichkeitserhaltender spärlicher Binärcodes stellt einen zusätzlichen Aspekt bei der Objekterkennung mit Klassifikatorhierarchien dar. Im Rahmen dieser Arbeit wurden unterschiedliche Strategien für die Generierung solcher Codes basierend auf den Aktivierungen der neuronalen Klassifikatoren innerhalb der Hierarchie entwickelt und ausgewertet.

Einen weiteren Schwerpunkt bildet das inkrementelle Hinzulernen von bisher unbekannten Objekten. Dies ist in komplexeren Realwelt-Umgebungen oft wünschenswert, da die Wahrscheinlichkeit, mit bislang unbekannten Objekten konfrontiert zu werden, relativ hoch ist. Im Rahmen dieser Arbeit wurde ein Verfahren entwickelt, welches es ermöglicht, nachträglich neue Objekte effizient zu lernen, ohne jedoch die Erkennungsleistung zuvor gelernter Objekte negativ zu beeinflussen.

Die Funktionalität des Ansatzes konnte in statistischen Experimenten auf unterschiedlichsten Datensätzen sowie durch die Anwendung auf einem autonomen mobilen Roboter bestätigt werden.

Abstract

The reliable recognition of three-dimensional objects from two-dimensional camera images is still a major problem in computer vision and artificial intelligence. For this problem exist numerous approaches which in general neither incorporate the inherent hierarchical nature of classification problems nor offer the possibility to learn new objects during the exploration phase.

Object recognitions tasks are in general multi-class problems where the number of classes is considerably higher than two. This work offers a novel approach for solving such problems by dividing the complex task into several hierarchically structured sub-problems that are easier to solve.

The work at hand covers several aspects of object recognition with hierarchical neural networks:

1. Generation of appropriate classifier hierarchies
2. Evaluation of these hierarchies (information fusion)
3. Selection of suitable features for the classification
4. Extensibility / adaptivity of the hierarchies
5. Generation of similarity preserving sparse binary codes

The selection of suitable feature types for the classification of objects from various domains is an important aspect for the recognition of three-dimensional objects. In the context of this work diverse feature types were deployed and evaluated.

Different strategies for the retrieval of the combined classification result from the classifier hierarchies were developed and evaluated. The most promising approaches were the retrieval strategy similar to decision trees, the retrieval strategy utilising the Dempster-Shafer evidence theory as well as the retrieval strategy utilising similarity preserving codes and inter-state decision templates.

The generation of similarity preserving sparse binary codes is an additional aspect of the object recognition with hierarchical neural network classifiers. Within the scope of this work several strategies for the generation of such codes based on the activation of the neural classifiers within the hierarchy were developed and evaluated.

Another focus was the incremental learning of hitherto unknown objects. In complex real-world environments this is a often desirable capability as the confrontation with so far unfamiliar objects is very likely. Within the scope of this work a method for subsequently learning new objects in an efficient manner without negatively influencing the classification performance of previously learnt objects.

The functionality of the approach could be confirmed by means of statistical experiments on various data sets as well as by the implementation on an autonomous mobile robot.

Acknowledgments

I would like to express my profound thanks to the various people who, during the time that it took to complete this thesis, provided me with their useful and helpful assistance.

My deepest gratitude goes to Prof. Dr. Günther Palm, my PhD supervisor, for supporting this work with his inspiring ideas, suggestions, recommendations and constructive criticisms.

I am indebted to my mentor Dr. Friedhelm Schwenker, who, despite his own immense workload and tight schedule, always could spare me time and who contributed to this thesis with his knowledgable advice and guidance, his generous assistance and numerous motivating and valuable discussions.

I thank Prof. Dr. Heiko Neumann for his helpful recommendations and for kindly furnishing the second expert's opinion. My thanks go also to Prof. Dr. Friedrich von Henke for his commitment to act as third referee for this thesis.

My gratitude also to the German National Merit Foundation whose doctoral scholarship financed this thesis.

For their generous assistance and splendid company, I would like to acknowledge my colleagues at the Institute of Neural Information Processing of the University of Ulm.

Particular thanks are also due to Heiner Markert and Eduard Metzker for a critical reading of this thesis and for their useful comments and suggestions.

Thanks also to my parents for their continued support and encouragement and to my friends for encouraging and tolerating me through the long time of writing this thesis and for their patience and friendship.

Ulm
February, 2007.

Rebecca Fay

Contents

List of Figures

List of Tables

1 Introduction

This chapter gives an overview of the background and motivation of the thesis at hand. Research goals are formulated and the solution approach is sketched. On the basis of working hypotheses the main research questions are stressed. At the end of this chapter the structure of this work is outlined.

1.1 Background and Motivation

Object recognition has been subject to extensive research since numerous years resulting in various approaches that differ substantially including non-hierarchical classifiers, multiple classifier systems and hierarchical classifiers.

Pattern recognition in general and object recognition in particular are of importance in various domains and disciplines. In the automotive domain the reliable recognition of traffic signs, pedestrians and vehicles play an important role. For any robotic application the identification of miscellaneous objects is an essential capability. The recognition of written or typed characters is profitably applied e.g. in the postal domain. Other domains are speech recognition, personal identification, classification of text documents or the analysis of DNA sequences.

The thesis at hand focuses on the recognition of three-dimensional objects from two-dimensional camera images utilising hierarchical neural network classifiers.

In realistic applications object recognition problems are generally complex multi-class problems where the number of classes to discriminate between is considerably larger than two. To solve these multi-class problems is a vastly more demanding task than solving binary classification problems.

Many classification problems show a hierarchical structure, i.e. the classes to discriminate between can be grouped in a hierarchical manner. Figure 1.1 gives an example of such a hierarchically structured object domain. Findings in cognitive neuroscience [29] also indicate that categorisation, i.e. the grouping of objects into meaningful categories, and learning of categories plays an important role in

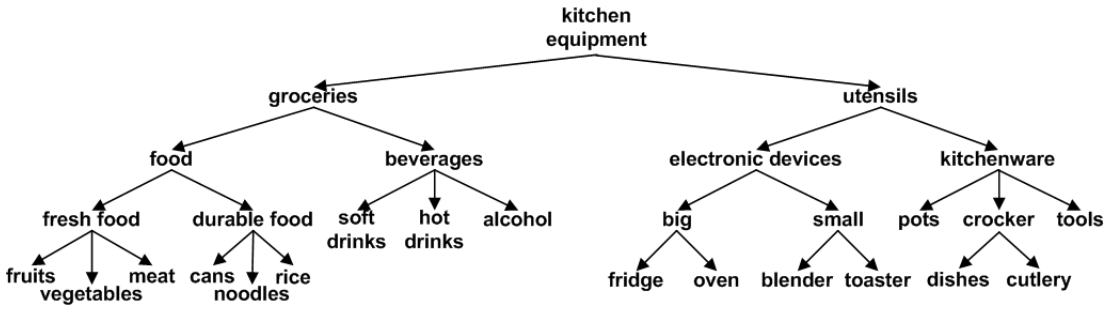

Figure 1.1: *Example of a hierarchically structured object recognition problem.*

advanced animals and humans.

These facts suggest that a divide-and-conquer-strategy might be suitable for solving such problems. Hierarchical neural network classifiers are an approach deploying such a divide-and-conquer-strategy.

Another problem also addressed within the scope of this research is the adaptive extension of the set of objects the recognition system can handle. In complex scenarios it is likely that the classifier has to identify so far unknown objects. Thus the capability of learning this new object is extremely useful.

One well-established domain of application for object recognition is the robotic field. For robots it is essential to be able to identify and categorise objects of various kind. Robots have e.g. to be able to identify obstacles in order to avoid them or to classify objects in order to manipulate them according to commands they are given. The approach proposed in this work has been deployed in this context.

1.2 Research Goals and Solution Approach

The research presented in this work was basically motivated by the aim of reliable recognition of three-dimensional objects from two-dimensional camera images. The recognition of three-dimensional objects shows several intricatenesses. Three-dimensional objects often have strongly varying appearances depending on the point of view. Figure 1.2 depicts this. It gives three considerably differing views of one object, that cannot straightforwardly be recognised as belonging to the same object. Other problems when classifying three-dimensional objects are inter alia the varying illumination conditions, occlusion, unfamiliar objects and cluttered scenes.

Taking these suppositions into account a straightforward implication is the usage of feature types appropriate for the recognition of three-dimensional objects. Thus

Figure 1.2: *Different views of a cup. The appearance of a three-dimensional object such as a cup vary strongly depending on the point of view.*

one focus of this thesis is the identification of feature types which enable the reliable recognition of three-dimensional objects.

As approach for the object classification hierarchical neural network classifiers were chosen due to the advantages they provide such as the availability of intermediate results, easy extensibility and incorporation of the hierarchical nature of classification problems, and their suitability for solving large-scale multi-class problems.

1.3 Working Hypotheses

In this thesis several research questions were elaborated. This section summarises and formulates these questions as six working hypotheses. The working hypotheses are evaluated by means of several statistical experiments.

H1: Hierarchical neural network classifiers are suitable for solving complex classification problems.

Large scale multi-class problems are intricate and difficult to solve whereas classification with a small number of classes such as binary classification problems are considerably easier to solve. Thus a divide-and-conquer strategy which reduces the complex multi-class problem to several simple classification problems seems a promising approach. Moreover it seems favourably to exploit the hierarchical structure often inherent to realistic classification tasks. Therefore it is necessary to find an appropriate way of decomposing the original classification task in smaller tasks that are easier to cope with.

H2: Complex strategies considering the complete hierarchy for retrieving the combined classification result are superior to simple evaluation strategies only considering part of the classifiers.

When only considering part of the classifiers in the decision process information provided by the classifiers not involved is disregarded. The individual classifiers only provide isolated information. In case one of the classifier provides flawed information this can more easily be compensated for in a global context. If for example in a decision tree like manner only the classifiers on the path with the highest evidence are taken into account wrong decisions at higher levels of the hierarchy cannot be corrected or when only considering the results of the end nodes and disregarding the information coming from the upper classifiers erroneous classifiers may have a strong impact which cannot be rectified.

H3: The Dempster-Shafer theory of evidence is an appropriate framework for combining the information provided by the individual classifiers within the hierarchy.

Hierarchical neural network classifiers provide hierarchically structured information. The information provided does not only apply to single classes but also to sets of classes. Moreover the classifier outputs express both uncertainty and ignorance. During the retrieval phase a non negligible number of classifiers within the hierarchy are presented samples of classes they have not been trained with. Then this should be expressed as ignorance. In case a classifier is in doubt regarding the classification result this should be expressed as uncertainty. Moreover the classifiers within the hierarchy only provide incomplete knowledge. The classifiers do not give evidence for each individual class but only for a subset of classes and in many cases this information cannot be further subdivided between the elements of a group of classes. Eventually the evidences provided by the individual classifiers need to be combined to a collective result. The Dempster-Shafer theory of evidence addresses all of the above listed issues and thus is likely to be a suitable framework for fusing the different classifier outputs to a combined output. An additional benefit is the availability of combined evidences for all classes as well as possible sets of classes.

H4: Classifier hierarchies are eligible for adding new classes during run-time.

Hierarchical classifiers can rather easily be extended as only local adaption is required and the major part of the hierarchy remains unchanged. If non-hierarchical classifiers have to be adapted global changes are required and the complete classifier is affected.

A requirement for the incremental adaption of classifiers is that new classes should be learnt with sufficient quality and in adequate time, while not negatively effecting the classification quality of the previously learnt classes. The fact that only local changes are required when incrementally extending the hierarchy gives reason for the assumption that classifier hierarchies facilitate this.

H5: Visual similarity preserving codes can be generated rather facile and straightforward from classifier hierarchies.

As the hierarchy is generated finding natural groups within the data in the different feature spaces it represents similarity with respect to the feature space. This similarity is a visual similarity as the features are extracted from camera images of the objects to be classified. Hence codes generated from the activation of the neurons of the neural classifiers within the hierarchy are likely to represent visual similarity.

H6: Complex histogram based features such as orientation histograms and colour histograms are appropriate for the classification of three-dimensional objects.

Histogram based features show several beneficial features. They are invariant to translation. Furthermore colour-based histograms are also invariant to rotation. Simple features presumably do not have enough discrimination power. Moreover during the hierarchy generation process supposably the powerful feature types are preferred to less powerful feature types.

1.4 Structure of This Work

This thesis is composed of three main parts. The first part describes the theoretical foundations of this work. As neural networks are a central building block of the developed approach, their basics and how they relate to the goals of this work are particularised. Furthermore theories for representing uncertainty are introduced. The section describes how these theories can be applied to the problem of information fusion in general. The most suitable of these theories, the belief theory, has been utilised for information fusion within the hierarchical neural network classifiers. Moreover this part shortly describes methods for preprocessing the data as well as methods for statistically evaluating the proposed approaches. In part II the solution approach developed within the scope of this thesis is outlined. This approach comprises several aspects of hierarchical neural network classifiers such as the generation of reasonable hierarchies, the selection of appropriate features, strategies for fusing the information within the hierarchy, extending the

hierarchy as well as the generation of sparse codes from these hierarchies. The third part shows different applications of classifier hierarchies and the results of the evaluation of the proposed approach. In part IV a summary of this work is given and the proposed approach is discussed. The methods developed within the scope of this thesis is also compared to related approaches and the main contributions of this work are emphasised.

Part I: Basic Methods

In the following four chapters the theoretical foundations of this work are described.

2 Neural Networks

This chapter focuses on artificial neural networks which are a core element of this work. At first neural networks are introduced in general and their application to classification problems in particular is exemplified. Then the neural networks used within the scope of this thesis are described. This chapter is concluded by comparing the different types of neural networks, outlining their advantages and drawbacks as well as their application within the developed approach.

Neural networks have been subject to comprehensive research for many years. An artificial neural network or simply neural network is a massively parallel computing paradigm that roughly follows cortical structures of the brain. Its interconnected processing units are called neurons which produce a collaborative output. The neurons utilise mathematical models for information processing on a connectionistic basis. Neural networks are very robust and fault tolerant as their overall functionality is not severely compromised by a few malfunctioning neurons owing to the collective performance of the function. Neural networks are trainable, i.e. they learn to solve complex tasks from a set of representative examples and generalise the thereby attained knowledge thus being able to accomplish so far unseen tasks. Neural networks can model complex non-linear dependencies between the network inputs and outputs. Neural networks are used for various tasks such as data mining, function approximation or classification.

Neural networks play an essential role in this thesis. In the following the neural networks utilised within the scope of this thesis are described compendiously. In the context of this work these neural networks are used for two tasks: classification and clustering. Classification incorporates the construction of a classifier on the basis of labelled data and the categorisation of data utilising this trained classifier. The aim of clustering is the partition of not necessarily labelled data into groups consisting of similar data points. The former task is conducted by k-nearest neighbour classifiers, fuzzy k-nearest neighbour classifiers, learning vector quantisation networks, radial basis function networks and associative memories. The k-means algorithm and the learning vector quantisation networks can be used for the latter task.

The objective of a given classification problem is the assignment of one of l predefined classes to a presented input sample. If neural networks are used for object recognition an object is represented by a number of features, which form a d dimensional feature vector x within the feature space $\mathcal{X} \subseteq \mathbb{R}^d$. A classifier therefore realises a mapping from feature space $\mathcal{X}$ to a finite set of classes $\Omega = \{1, 2, ..., l\}$ which estimates the class ω of a pattern x. A neural network is trained to approximate this unknown mapping and thus perform a classification task using supervised learning algorithms. A set of training examples $\tau := \{(x^\mu, t^\mu), \mu = 1, 2, ..., M\}$ is presented to the network. The training set consists of M feature vectors $x^\mu \in \mathcal{X}$ each labelled with a class membership $t^\mu \in \Omega$. The set of feature vectors is denoted by $X = \{x^{\mu,\mu=1,2,...,M}\}$ and the set of training vectors is denoted by $T = \{t^{\mu,\mu=1,2,...,M}\}$. During the training phase the network parameters are adapted to approximate this mapping as accurately as possible. This is accomplished by successively presenting the network all training examples x^μ of the training data set τ. Depending on the network output the weights of the network are adapted in order to adjust the network output to the desired output t^μ. The presentation of a sample x^μ to the network is interpreted as one learning step t. The samples of the training data set τ are presented repeatedly to the network until the network output is sufficiently accurate. There are different strategies for neural network learning: incremental or online learning and batch-mode learning. When using online learning the weights of the neural network are adapted immediately after the presentation of one training sample x^μ without considering the next training sample. In batch-mode learning the weights are updated after the presentation of all training samples in the training data set τ and the learning rule averages over all presented training samples. Batch-mode learning is a deterministic process whereas online learning is a stochastic process as it depends on the order of the presented samples. With regards to storage capacity online learning is less demanding than batch-mode learning. Within the scope of this thesis online learning has been deployed. In the classification phase unlabelled data $x^\mu \in \mathcal{X}$ are presented to the trained network. The network output $K(x^\mu) = \omega \in \Omega$ can be interpreted as an estimation of the class corresponding to the input vector x given a sufficient generalisation capability of the classifier.

The number of classes l is used to categorise l-class classification problems. A two-class problem is the simple case $l = 2$. Such a problem is also called dichotomous classification problem. If $l > 2$ the problem at hand is a multi-class problem or a so called polychotomous classification problem.

The following five sections introduce the most important neural architectures used in this thesis.

2.1 k-Means Clustering

The k-means cluster analysis [50] is an unsupervised algorithm utilising competitive learning for determining groups, so called clusters, of data points that belong together. The distance in the feature space is used as a similarity measure. Data points having a small distance in the feature space will be grouped together in one cluster, which is defined by a prototype vector c_j. It divides the input space $\mathcal{X}$ into k disjoint regions $\mathscr{R}_j \subset \mathbb{R}^d$ that are represented by corresponding k prototype vectors c_j on the basis of the Euclidean distance. These regions $\mathscr{R}_j$ are defined by

$$\mathscr{R}_j = \{x \in \mathcal{X} : \|x - c_j\| = \min_{i=1,2,...,k} \|x - c_i\|\} \tag{2.1}$$

and form a Voronoi tessellation. These regions divide the data set τ into k disjoint subsets called clusters:

$$\mathscr{C}_j = \mathscr{R}_j \cap X \tag{2.2}$$

where X is the set of feature vectors in the data set τ.

The clustering algorithm minimises the sum-of-square criterion

$$E = \sum_{j=1}^{k} \sum_{x^\mu \in \mathscr{C}_j} \|x^\mu - c_j\|^2 \tag{2.3}$$

searching for the optimal values for $c_j, j = 1, 2, ..., k$.

A k-means network consists of an input layer and a hidden layer. Figure 2.1 depicts the k-means network architecture.

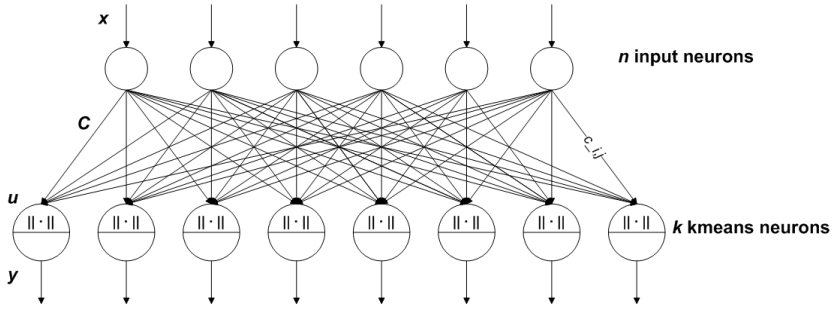

Figure 2.1: *k-means network. A k-means network is a two-layer network with n neurons in the input layer and k neurons in the hidden layer.*

First of all the number of clusters $k \geq 1$ is defined and the cluster centres are initialised. In the training phase the training samples x^μ are consecutively presented

to the network. For each prototype c_j the Euclidean distance $d_j = \|x^\mu - c_j\|$ to the presented sample x^μ is calculated and the index of the prototype with the smallest distance is determined by

$$j^* = \text{argmin}_j \|d_j\|. \tag{2.4}$$

The winning prototype c_{j^*} is then adapted using the following learning rule

$$c_{j^*} = c_{j^*} + \eta_t (x^\mu - c_{j^*}) \tag{2.5}$$

which moves the prototype c_{j^*} in direction of the sample x^μ. The learning rate η_t is defined by

$$\eta_t = \frac{1}{|\mathscr{C}_{j^*}| + 1} \tag{2.6}$$

where $|\mathscr{C}_{j^*}|$ is the number of data points assigned to cluster $\mathscr{C}_{j^*}$.

2.1.1 Initialisation Methods

As the k-means clustering algorithm is sensitive to the initialisation of the prototypes such that the results differ considerably depending on the initialisation and inappropriate initialisation is likely to lead to poor performance, appropriate initialisation strategies are beneficial. The different initialisation strategies strongly differ with regard to their computational complexity. The different initialisation methods can be categorised as data independent, i.e. completely neglecting the positions of the data points in the feature space, simple data dependent, i.e. the locations of the data points are utilised in the initialisation process, and sophisticated, i.e. complex algorithms are used to initialise the cluster centres.

A simple data independent approach is the initialisation of the prototypes with random values, i.e. the prototypes are placed randomly across the feature space. As this method does not take the distribution of the data across the feature space into consideration the selection of the initial prototypes could possibly be disadvantageous in such that the prototypes could be located far off the data points. Random sampling, an initialisation with randomly chosen data points, makes allowance for this. Nevertheless this simple data dependent method cannot ensure that the prototypes are properly spread out over the data. Moreover it is very sensitive to how well the chosen data points represent the overall population of the data.

A more sophisticated and less arbitrary data dependent approach is the maximin algorithm [3], which uses data points x^μ to initialise the prototypes c_j, but takes

the distance between the data points into account. Prototypes lying close together are avoided by maximising the distances between them. It also allows for controlling the number of prototypes by a parameter $\gamma \in [0, 1]$.

The algorithm starts with randomly selecting one data point $x^{\mu_1} \in \tau$ out of the training data set. This data point forms the first prototype $c_1 = x^{\mu_1}$. The second prototype is set to the data point $x^{\mu_2} \in \tau \setminus \{x^{\mu_1}\}$ whose distance to the first prototype c_1 is maximal:

$$\mu_2 = \text{argmax}_{\mu=1,2,...,M;\mu\neq\mu_1} \|x^\mu - c_1\|^2. \tag{2.7}$$

The additional centres are determined by identifying the data points lying farthest aside of the prototypes. They are chosen as additional prototypes if the distance to the prototypes exceeds a certain amount that is a fraction of the mean prototype distance (see equation 2.10). The process of selecting data points as additional prototypes is repeatedly carried out until the newly selected data point lies to close to the already chosen prototypes.

Let $\tilde{k}$ be the number of prototypes initialised so far. For each data point $x^\mu \in \tau \setminus \{x^{\mu_j}\}_{j=1,2,...,\tilde{k}}$ in the training remaining data the distances to the $\tilde{k}$ prototypes $c_j, j = 1, 2, ..., \tilde{k}$ is calculated and the closest prototype $c_{j^*_\mu}$ is identified:

$$j^*_\mu = \text{argmin}_{j=1,2,...,\tilde{k}} \|x^\mu - c_j\|. \tag{2.8}$$

Then the maximin distance $d^{maximin}$ is given by

$$d^{maximin} = \max_{\mu\in\{1,2,...,M\}\setminus\{\mu_1,\mu_2,...,\mu_{\tilde{k}}\}} \min_{j=1,2,...,\tilde{k}} \|x^\mu - c_j\|. \tag{2.9}$$

Thus the corresponding data point $x^{\mu_{\tilde{k}+1}}$ which has the largest distance to its nearest prototype $c_{j^*_{\mu_{\tilde{k}}}}$ is determined.

If the maximin distance $d^{maximin}$ is greater than the mean cluster distance multiplied by the factor γ the data point $c_{j^*_{\mu_{\tilde{k}}}}$ is chosen as a new prototype otherwise the algorithm terminates:

$$d^{maximin} > \gamma \frac{1}{\frac{\tilde{k}(\tilde{k}-1)}{2}} \sum_{i=1}^{\tilde{k}} \sum_{j=i}^{\tilde{k}} \|c_i - c_j\|. \tag{2.10}$$

If there is a considerable decrease of the maximin distance $d^{maximin}$ from one selection step to the next this is an indication for the fact that with the last selection the number of prototypes exceeded the number of clusters that can be found in the data set. Thus this last chosen prototype should not be added. The

amount of decrease still allowed is defined by the parameter γ. The choice of the parameter γ has a direct influence on the number of prototypes selected. For $\gamma = 0$ all data points in τ will be selected as prototypes.

One drawback of this initialisation algorithm besides its computational complexity is its sensitivity to outliers, which tend to be chosen as prototypes by the algorithm.

2.2 Nearest Neighbour Classifier

The nearest neighbour algorithm is a classification method based on the closest training samples in the feature space with respect to some distance d. It does neither require extensive training nor the adjustment of numerous parameters. The classifier is trained by making all M training samples $x^\mu \in \tau$ to prototypes c_j resulting in $m = M$ prototypes. To each prototype the class $\omega(c_j) = t^\mu$ of the corresponding training sample x^μ is assigned where $\omega : \mathcal{X} \to \Omega$ is a function specifying the class of a training sample or a prototype. No further adjustment is required. The only parameter to be determined is the number of prototypes k included in the decision process.

2.2.1 k-Nearest Neighbour Classifier

The k-nearest neighbour classifier assigns a class ω to a presented sample x by identifying the $k \leq m$ nearest neighbours c_{υ_i} out of the set of m prototypes regardless of their class. Let $d_j = \|x - c_j\|$ be the Euclidean distance of the presented data point x and $(\upsilon_i)_{i=1}^m$ with $\upsilon_i \in 1, 2, ..., m$ a permutation such that $d_{\upsilon_1} \leq d_{\upsilon_2} \leq ... \leq d_{\upsilon_m}$. Then the set of the k nearest neighbours is given by

$$\mathcal{N}_k(x) = \{c_{\upsilon_1}, ..., c_{\upsilon_k}\}. \tag{2.11}$$

The subset containing the nearest neighbours which belong to class j is defined as

$$\mathcal{N}_k^j = \{c \in \mathcal{N}_k(x) | \omega(c) = j\}. \tag{2.12}$$

The class $\omega \in \{1, 2, ..., l\}$ assigned to the sample x is determined by

$$\omega = \text{argmax}_{j=1,2,...,l} |\mathcal{N}_k^j(x)| \tag{2.13}$$

i.e. the class which is most frequently represented in the set of the k nearest neighbours.

A special case is the 1-nearest neighbour classifier. This is the simplest form of this classifier type, which only considers the nearest prototype to the presented sample. The sample is then assigned the class of this prototype.

2.2.2 Fuzzy k-Nearest Neighbour Classifier

The fuzzy k-nearest neighbour classifier provides an estimation for the degree of membership of an input vector x to each class $j \in \{1, 2, ..., l\}$. The distances between the data vector x and the k nearest prototype vectors are incorporated in the calculation of these membership values. Therefore the k nearest neighbours $\mathcal{N}_k(x)$ and the distribution of the class affiliations $\mathcal{N}_k^j(x)$ of these nearest neighbours are determined where $\mathcal{N}_k(x)$ is the set of the k nearest prototypes to the input vector x (see equation 2.11) and $\mathcal{N}_k^j(x)$ is the set of the prototypes of the k nearest prototypes that belong to class j (see equation 2.12).

The class membership Ξ is represented by a mapping $\Xi : \mathcal{X} \rightarrow [0, 1]^l$ specifying for a given input vector x its membership to each of the l classes such that $\Xi(x) = (\Xi_1(x), ..., \Xi_l(x))$ is a normalised vector with

$$\Xi_j(x) = \frac{\xi_j(x)}{\sum_{i=1}^{l} \xi_i(x)} \tag{2.14}$$

and $\xi_j(x)$ is defined as

$$\xi_j(x) = \frac{1}{\sum_{c_i \in \mathcal{N}_k^j} \|x - c_i\| + \alpha} \tag{2.15}$$

where the parameter α is used to amplify small distances.

The so calculated class memberships $\Xi_j(x)$ obviously satisfy the following conditions: $\Xi_j(x) \in [0, 1]$ and $\sum_{j=1}^{l} \Xi_j(x) = 1$.

The class ω assigned to the presented sample x is then the class with the highest class membership

$$\omega = \text{argmax}_{j=1,2,...,l} \Xi_j(x). \tag{2.16}$$

2.3 Learning Vector Quantisation

Learning **V**ector **Q**uantisation (LVQ) [43] is a competitive supervised prototype-based learning algorithm for solving classification problems. It is closely related

to the k-nearest neighbour classifier[1].

The underlying principle of the classification of data points is the comparison of the data points x^μ to a set of k prototypes c_j. The prototypes are located in the feature space and can be interpreted as an exemplary representation of typical data. To each prototype c_j a class $\omega(c_j)$ is assigned. The localisation of the prototypes within the feature space is determined during the learning phase in a supervised manner by means of labelled training data. The similarity between the presented data points and the prototypes is measured in terms of Euclidean distances in the feature space. The decision boundaries are defined by the nearest-neighbour rule utilising the Euclidean distance. Hence the feature space $\mathbb{R}^d$ is divided into a Voronoi tessellation. The class assigned to a data point x is the corresponding class $\omega(c_{j^*})$ of the nearest prototype c_{j^*}. The index j^* of the nearest prototype is determined by

$$j^* = \text{argmin}_j \|x - c_j\|. \tag{2.17}$$

This decision process is identical to the decision process utilised by the 1-nearest neighbour classifier.

In the training phase a set of M training samples $\tau = \{(x^\mu, t^\mu)|\mu = 1, 2, ..., M\}$ is repeatedly presented to the network. The prototypes are adapted according to their distance to the presented training data and the identicalness of their corresponding class and the class of the presented data. The learning rule rewards correct classifications and penalises incorrect classifications. There are several learning strategies which differ inter alia in the number of prototypes adapted in one learning step. Some of the established learning schemes are described below.

An LVQ network consists of an input layer and one hidden layer. The neurons in the hidden layer are called prototypes or code vectors. Figure 2.2 shows an example of an LVQ network.

In the following variants [44] of the LVQ algorithm, namely LVQ1, OLVQ1, LVQ2.1 and LVQ3, exploiting different learning algorithms are described. The initialisation and the retrieval of the classification result is identical for all variants.

2.3.1 LVQ1

This learning algorithm is the basic version of the LVQ algorithm. It is a so called winner-takes-all scheme, where only the prototype closest to the presented data

[1] A 1-nearest neighbour classifier can straightforwardly be realised as an LVQ network by constituting a prototype out of each data point in the training set. No learning is conducted and the classification is performed by identifying the prototype closest to the presented data point and assigning the corresponding label.

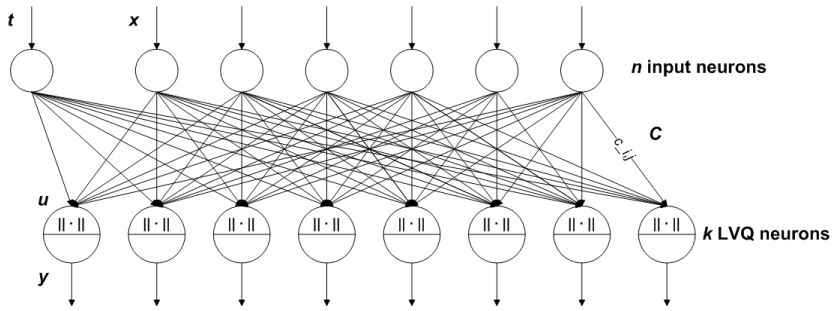

Figure 2.2: *Learning Vector Quantisation network. An LVQ network is a two-layer network with n neurons in the input layer and k neurons in the hidden layer.*

point, the winner neuron, is adapted. The winner c_{j^*} is identified as follows:

$$d_j = \|x^t - c_j^t\| \tag{2.18}$$

where d_j is the distance of the prototype c_j to the presented data point x at a given learn step t.

The index of the winner neuron is given by

$$j^* = \text{argmin}_j d_j. \tag{2.19}$$

The update rules are defined as:

$$c_j^{t+1} = \begin{cases} c_{j^*}^t + \eta(t)[x^t - c_{j^*}^t] & : \omega(x^t) = \omega(c_{j^*}^t), j = j^* \\ c_{j^*}^t - \eta(t)[x^t - c_{j^*}^t] & : \omega(x^t) \neq \omega(c_{j^*}^t), j = j^* \\ c_j^t & : j \neq j^* \end{cases} \tag{2.20}$$

Only the winner neuron is updated. If the class of the winner neuron is identical to the class of the presented data point the centre of the winner neuron is shifted towards the presented data point, otherwise it is pushed away.

The learning rate $\eta(t)$ with $0 \leq \eta(t) \leq 1$ can either have a constant value or decrease monotonically with time t.

2.3.2 Optimised Learning Rate LVQ1

The **O**ptimised Learning Rate **LVQ1** (OLVQ1) algorithm is a modification of the LVQ1 algorithm such that an individual learning rate $\eta_j(t)$ is assigned to each prototype c_j. This optimisation of the learning rate yields accelerated convergence.

The learning rule is defined as

$$c_j^{t+1} = \begin{cases} c_{j^*}^t + \eta_j(t)[x^t - c_{j^*}^t] & : \omega(x^t) = \omega(c_{j^*}^t), j = j^* \\ c_{j^*}^t - \eta_j(t)[x^t - c_{j^*}^t] & : \omega(x^t) \neq \omega(c_{j^*}^t), j = j^* \\ c_j^t & : j \neq j^* \end{cases} \tag{2.21}$$

The individual learning rates are determined by

$$\eta_j(t) = \min(\frac{\eta_j(t-1)}{1 + s(t)\eta_j(t-1)}, \eta_{max}) \tag{2.22}$$

with $s(t) = 1$ if c_j and x belong to the same class and $s(t) = -1$ otherwise.

As the learning rate η_j can also increase in time, it is necessary to restrict η_j^t by η_{max}.

2.3.3 LVQ2.1

The learning in this version of the LVQ algorithm is performed by concurrently adapting the two prototypes c_{j^*} and $c_{j^{**}}$ that are closest to the presented sample x^t, where c_{j^*} is the prototype closest to the sample x^t and $c_{j^{**}}$ is the second closest prototype. The update is only performed if two requirements are met:

1. One of the two prototypes must belong to the same class as the presented sample x^t and the other prototype must belong to a different class.
2. The data point x^t falls into a so called window of relative width w.

The window is a zone of values defined around the midplane of d_{j^*} and $d_{j^{**}}$. The data point x^t falls into this window if

$$\frac{d_{j^*}}{d_{j^{**}}} > \frac{1-w}{1+w} \tag{2.23}$$

where d_{j^*} and $d_{j^{**}}$ are the Euclidean distances of the data point x^t to the two closest prototypes c_{j^*} and $c_{j^{**}}$ respectively.

The corresponding learning rules for the case $\omega(c_{j^*}) = \omega(x^t)$ and $\omega(c_{j^{**}}) \neq \omega(x^t)$ given by

$$c_{j^*}^{t+1} = c_{j^*}^t + \eta(t)[x^t - c_{j^*}^t] \tag{2.24}$$

$$c_{j^{**}}^{t+1} = c_{j^{**}}^t - \eta(t)[x^t - c_{j^{**}}^t] \quad . \tag{2.25}$$

The case $\omega(c_{j^*}) \neq \omega(x^t)$ and $\omega(c_{j^{**}}) = \omega(x^t)$ is defined analogously.

2.3.4 LVQ3

This variant of the LVQ algorithm like the LVQ2.1 variant updates the two closest prototypes c_{j^*} and $c_{j^{**}}$, where c_{j^*} is the prototype closest to the sample x^t and $c_{j^{**}}$ is the second closest prototype. The prototypes are only adapted if at least one of the two closest prototypes belongs to the same class as the presented data point x^t. Moreover the data point x^t must fall into the window of relative width w.

The corresponding learning rules for the case $\omega(c_{j^*}) = \omega(x^t)$ and $\omega(c_{j^{**}}) \neq \omega(x^t)$ if x^t falls into the window w are given by

$$c_{j^*}^{t+1} = c_{j^*}^t + \eta(t)[x^t - c_{j^*}^t] \tag{2.26}$$
$$c_{j^{**}}^{t+1} = c_{j^*}^t - \eta(t)[x^t - c_{j^*}^t] \tag{2.27}$$

The case $\omega(c_{j^*}) \neq \omega(x^t)$ and $\omega(c_{j^{**}}) = \omega(x^t)$ if x^t falls into the window w is defined accordingly.

In the case $\omega(c_{j^*}) = \omega(c_{j^{**}}) = \omega(x^t)$ the learning rule is defined as

$$c_{j^*}^{t+1} = c_{j^*}^t + \epsilon\eta(t)[x^t - c_{j^*}^t] \tag{2.28}$$
$$c_{j^{**}}^{t+1} = c_{j^*}^t + \epsilon\eta(t)[x^t - c_{j^*}^t] \tag{2.29}$$

where ϵ is a scaling factor depending on the size of the window w.

2.3.5 Initialisation Methods

The prototypes can be initialised randomly or with k randomly chosen data points of the training data set τ. When using data points to initialise the prototypes there are some strategies for improving this simple initialisation procedure. It is beneficial to use only data points for initialising the prototypes that are no outliers but are surrounded by data points of the same class. Thus only data points whose k nearest neighbours belong to the same class are chosen as possible data points for initialisation. Also the distribution of the classes across the prototypes should be considered. Two straightforward strategies for this are on the one hand the equal distribution of the classes across the prototype and on the other hand the distribution of the classes across the prototypes proportional to the distribution of data points across the classes. The former strategy initialises the prototypes such that approximately the same number of prototypes belongs to each class. The latter one initialises the prototypes such that a contingently different number of prototypes belongs to the individual classes.

2.4 Radial Basis Function Networks

Radial **B**asis **F**unction (RBF) networks were first introduced to neural network literature in the late 1980s [9]. They originate from regularisation theory and multivariate function approximation theory. They are applied to regression and classification problems. Another field of application is time series prediction. Radial basis function networks are non-parametric models, i.e. no a priori knowledge about the function to be approximated is used.

2.4.1 Radial Basis Functions

Radial basis functions form a special class of functions which monotonically decrease or increase with increasing distance from the centre complying the general equation

$$h(x) = \Phi((x-c)^T R^{-1}(x-c)) \tag{2.30}$$

where $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}$ is the respective radial basis function, c is the centre of the radial basis function and R is called metric. The term $(x-c)^T R^{-1}(x-c)$ defines the distance between the input vector x and the centre c of the radial basis function in the metric given by R. In case of the Euclidean metric $R = r^2 I$ for a given scalar radius r with I being the identity matrix equation 2.30 reduces to

$$h(x) = \Phi(\frac{(x-c)^T(x-c)}{r^2}) = \Phi(\frac{\|x-c\|_2^2}{r^2}). \tag{2.31}$$

Common examples of radial basis functions used in RBF networks are the Gaussian $\Phi(u) = e^{-u}$, the Cauchy or inverse quadric $\Phi(u) = (1+u)^{-1}$, the multi-quadric $\Phi(u) = (1+u)^{\frac{1}{2}}$ and the inverse multi-quadric $\Phi(u) = (1+u)^{-\frac{1}{2}}$.

Radial basis functions show local response behaviour, i.e. they respond only to a small region of the input space close to their centres. This resembles the locally tuned responses observed in biologic neurons, e.g. in the auditory and visual system.

2.4.2 Network Architecture

An RBF network consists of an input layer with n neurons, one non-linear hidden layer consisting of k neurons and one linear output layer comprising l neurons. The neurons in the hidden layer use radial basis functions as transfer functions

(see below). They are determined by the centres c_j of the radial basis functions and are called prototypes. An example of an RBF network is shown in figure 2.3.

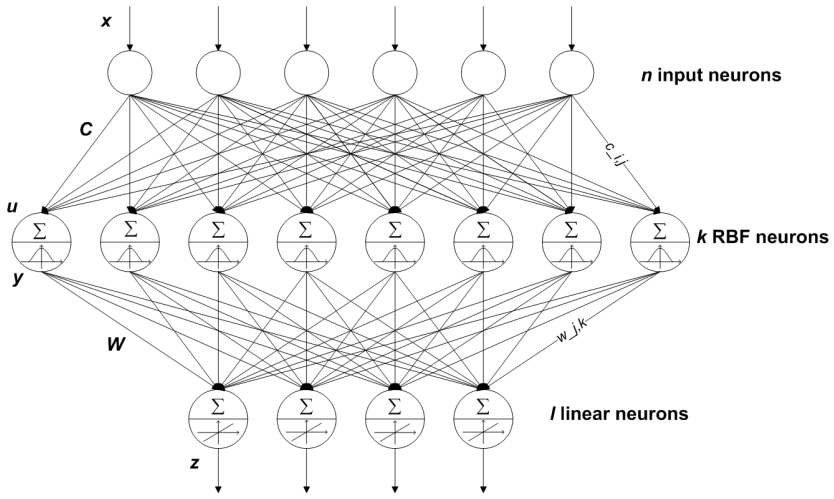

Figure 2.3: *Radial Basis Function network. An RBF network is a three-layer network with n neurons in the input layer, k RBF neurons in the hidden layer and l linear neurons in the output layer.*

In the following the Gaussian and the inverse multi-quadric function are used as RBF.

The output of the RBF prototypes y_j is calculated by applying the transfer function h, the RBF, to the activation of the RBF neuron that is calculated as the distance between the presented data point x and the centre c_j of the RBF prototype. The RBF is defined by the parameter σ_j specifying the width of the RBF of the jth prototype. σ_j can be either be real-valued $\sigma_j \in \mathbb{R}$ or specified as a vector $\sigma_j \in \mathbb{R}^d$ or as a matrix $\sigma_j \in \mathbb{R}^{d \times d}$.

In case σ_j is real-valued and the RBF is the Gaussian function the output of the jth RBF prototype is calculated by

$$y_j = h_j(x) = e^{-\frac{\|x-c_j\|_2^2}{2\sigma_j^2}}, j = 1, 2...., k. \tag{2.32}$$

A vector-valued σ_j yields the following calculation of the output y_j

$$y_j = h_j(x) = e^{-\sum_{i=1}^{d} \frac{\|x_i-c_{i,j}\|_2^2}{2\sigma_{i,j}^2}}, j = 1, 2...., k \tag{2.33}$$

where $\sigma_{i,j}$ determines the width of the jth Gaussian function in the ith dimension of the feature space.

The output of the linear layer z_j is computed by a weighted sum of the outputs of the k RBF prototypes:

$$z_m = \sum_{j=1}^{k} w_{j,k} y_j, m = 1, 2...., l. \tag{2.34}$$

If applied to classification problems the number of classes to be categorised defines the size of the output layer. The different classes are represented as a one-out-of-l code where l is the number of classes. Thus z_m can be interpreted as the degree of affiliation of the presented pattern x to class m.

2.4.3 Initialisation Methods

There exists miscellaneous strategies for initialising the different weights of an RBF network.

A rather simple approach to initialise the RBF centres c_j is the usage of random values. Another simple strategy is the initialisation with randomly chosen data points. If the RBF network is used to solve a classification problem it is beneficial to utilise the class information for the initialisation procedure. Hence a more sophisticated way to determine the initial centers $c_{i,j}$ of the RBF neurons is to perform class specific k-means clusterings. For each class $\omega \in \Omega$ a k-means clustering is conducted with all samples of this class. The resulting k-means prototypes are then used to initialise the RFB centres. LVQ clustering can also be used to initialise the RBF weights. The RBF centres are initialised with the LVQ prototypes.

The widths parameters σ_j of the RBFs can either be initialised randomly or estimated by local heuristics, e.g., by setting the σ_j of each prototype c_j equal to a multiple of the average distance to the p nearest other prototypes or by estimating the variance of the data points belonging to the respective clusters. The choice of the width parameters σ_j is a crucial factor for the classification quality of an RBF network. It has a direct impact on the degree of smoothness of the function approximated by the network. The smaller the widths are the less smoother are the realised functions.

The output weights $w_{j,k}$ can also be initialised randomly. When applied to a classification problem a slightly more complex approach is the initialisation according to the class distribution of the corresponding RBF prototype. The weight $w_{j,k}$ is set to the rate of samples in cluster j belonging to class k. A similar strategy only considers the class k^* represented most in cluster j. The weight w_{j,k^*} is set to one, all other weights are set to zero.

2.4.4 Training Methods

Let $\tau = \{(x^\mu, t^\mu) : \mu = 1, 2, ..., M\}$ be a set of M labelled training samples where x^μ is a d-dimensional feature vector and t^μ is the corresponding m-dimensional target vector specifying the class of the sample x^μ as an one-out-of-m code.

2.4.4.1 Pseudo Inverse Learning

The pseudo inverse [64] learning is one way to calculate the output weights $w_{j,m}$.

The output of the jth prototype when being presented the μth sample x^μ is $H^j_\mu = h_j(x^\mu)$. The outputs of all k prototypes for all M samples form the matrix $H = (H^j_\mu)^{\mu=1,2,...,M}_{j=1,2,...,k}$. The mth element of the target vector t^μ of the μth sample x^μ is given by t^μ_m. The matrix of the target vectors of all samples is given by $T = (t^\mu_m)^{\mu=1,2,...,M}_{m=1,2,...,l}$.

The matrix of the output layer weights $W = (w_{j,m})^{j=1,2,...,k}_{m=1,2,...,l}$ can be calculated using the pseudo inverse H^+:

$$W = H^+ T \tag{2.35}$$

where H^+ is the pseudo inverse matrix of the matrix H. The pseudo inverse matrix H^+ is calculated as

$$H^+ = \lim_{\alpha \to 0+} (H^T H + \alpha I)^{-1} H^T \tag{2.36}$$

where I is the identity matrix.

If the matrix $H^T H$ is invertible the calculatation of the pseud inverse matrix reduces to

$$H^+ = (H^T H)^{-1} H^T. \tag{2.37}$$

The weight matrix W minimises the error function

$$E(W) = \|HW - T\|^2. \tag{2.38}$$

2.4.4.2 Error Backpropagation

This gradient descent optimisation realises a combined training of the complete network. The parameters $c_{i,j}$, $\sigma_{i,j}$ and $w_{j,m}$ are adapted to minimise an error function E which gives a measurement for the difference between the network

output z and a teacher signal t specifying the desired output. A commonly used function is the mean square error:

$$E = \frac{1}{2} \sum_{\mu=1}^{M} \sum_{m=1}^{l} (t_m^\mu - z_m^\mu)^2. \tag{2.39}$$

From the error function E learning rules for the different weights $c_{i,j}$, $\sigma_{i,j}$ and $w_{j,m}$ can be derived as follows:

The output weights $w_{j,m}$ are updated using the following learning rule:

$$w_{j,m} = w_{j,m} - \eta_w h_j(x^\mu)(t_m^\mu - z_m^\mu) \tag{2.40}$$

where η_w is the corresponding learning rate.

The learning rules of the weights $c_{i,j}$ and $\sigma_{i,j}$ depend on the RBF used. When the Gaussian function is used as RBF the following learning rules for the weights $c_{i,j}$ and $\sigma_{i,j}$ are obtained:

$$c_{i,j} = c_{i,j} - \eta_c h_j(x^\mu) \frac{x_i^\mu - c_{i,j}}{\sigma_{i,j}^2} \sum_{m=1}^{l} w_{j,m}(x^\mu)(t_m^\mu - z_m^\mu) \tag{2.41}$$

$$\sigma_{i,j} = \sigma_{i,j} - \eta_\sigma h_j(x^\mu) \frac{(x_i^\mu - c_{i,j})^2}{\sigma_{i,j}^3} \sum_{m=1}^{l} w_{j,m}(x^\mu)(t_m^\mu - z_m^\mu) \tag{2.42}$$

where η_c and η_σ are the respective learning rates.

The usage of the inverse multi-quadric RBF leads to the following learning rules:

$$c_{i,j} = c_{i,j} - \eta_c h_j(x^\mu)^3 \frac{x_i^\mu - c_{i,j}}{\sigma_{i,j}^2} \sum_{m=1}^{l} w_{j,m}(x^\mu)(t_m^\mu - z_m^\mu) \tag{2.43}$$

$$\sigma_{i,j} = \sigma_{i,j} - \eta_\sigma h_j(x^\mu)^3 \frac{(x_i^\mu - c_{i,j})^2}{\sigma_{i,j}^3} \sum_{m=1}^{l} w_{j,m}(x^\mu)(t_m^\mu - z_m^\mu). \tag{2.44}$$

For other types of RBF the corresponding learning rules can be deduced by minimising the respective error functions with regards to the different weights $c_{i,j}$ and $\sigma_{i,j}$.

2.4.4.3 Learning Strategies

The training of the weights of the RBF network can be performed in different stages [75]. The three different learning schemes described in the following exploit

this. The individual learning procedures implement a different number of learning phases, where the computational complexity and classification performance that can be achieved increases with the number of learning phases conducted.

2.4.4.3.1 One-Phase Learning

This learning scheme adjusts only the output weights $w_{j,k}$ by a supervised optimisation of an error function. This can be accomplished e.g. by pseudo inverse learning (see equation 2.35) or by gradient descent optimisation (see equation 2.40). The centres c_j of the RBF prototypes are chosen randomly from the set of training samples. The scaling parameters σ_j are set equally for all prototypes to a predetermined value $\sigma \in \mathbb{R}$.

2.4.4.3.2 Two-Phase Learning

The usage of radial basis function that only respond locally allows for decoupling the estimation of the weights into a two-stage process as this ensures that only few radial basis functions will considerably be activated at a given time. Thus the hidden and the output layer of the RBF network are trained consecutively. In the first step the centres c_j of the RBF neurons and the corresponding scaling parameters σ_j are calculated utilising class-specific k-means, LVQ or decision trees (see section 2.4.3). The so obtained parameters of the radial basis functions are then kept fix, i.e. the transformation between the network input and the output of the hidden layer is fixed. Thus the network can now be treated like a single-layer network with a linear output layer. In the second step the output weights $w_{j,m}$ for the previously calculated centres c_j and widths σ_j can then be determined by pseudo inverse learning (see equation 2.35) or by gradient descent optimisation (see equation 2.40). This learning scheme is efficient and provides good classification results.

2.4.4.3.3 Three-Phase Learning

This learning scheme consists of three stages. The first two stages are identical to the steps conducted in two-phase learning. There the hidden and the output layer are trained separately. In the following learn step all parameters of the network c_j, σ_j and $w_{j,m}$ are further optimised simultaneously using error-backpropagation (see equations 2.41 and 2.42 or 2.43 and 2.44). This learning scheme utilises non-linear optimisation and is computationally expensive but yields improved classification results compared to the two-stage learning scheme.

2.5 Associative Memories

Associative **m**emories [86] (AM) are content-addressable structures that store a set of M associations between specific binary input representations $u^\mu \in \{0,1\}^n$ and specific binary output representations $v^\mu \in \{0,1\}^m$ with $\mu = 1, 2, ..., M$. After storing the M pairs of pattern $\{(u^\mu, v^\mu), \mu = 1, 2, ..., M\}$ an address pattern $\tilde{u}$ can be used to retrieve an associated pattern $\tilde{v}$, where the recall of data is based on the resemblance between the address pattern and the stored patterns. Associative memories show a certain error-correction ability and are fault-tolerant and robust.

A distinction is made between two types of associative memories: hetero-associative memories and auto-associative memories. In hetero-associative memories the output patterns are in general different from the input patterns, i.e. patterns are associated to other pattern, whereas in associative memories only pairs of identical pattern are stored, i.e. the patterns are associated with themselves. Auto-associative memories retrieve perviously stored pattern that most closely match the address pattern. Hence this type of associative memory allows for pattern correction or completion in case of addressing with noisy or incomplete pattern.

An associative memory is represented by synaptic connections between two neuron populations namely the address population associated with the input patterns u^μ and the retrieval population corresponding to the output patterns v^μ. The memory matrix A embodies the synaptic connections where the synaptic weight connecting neuron i of the address population to neuron j of the retrieval population is specified by the matrix entry $A_{i,j}$ which is determined from the superposition of the outer products of the input and output pattern. There are different ways of determining the entries of the memory matrix A when storing pairs of pattern. The most common approach uses clipped Hebbian learning yielding a binary memory matrix $A \in \{0,1\}^{n \times m}$:

$$A_{i,j} = \min(1, \sum_{\mu=1}^{M} u_i^\mu v_j^\mu) \in \{0,1\}. \tag{2.45}$$

Another learning strategy in use is the linear additive Hebbian learning:

$$A_{i,j} = \sum_{\mu=1}^{M} u_i^\mu v_j^\mu \in \mathbb{N}_0. \tag{2.46}$$

A modification of the linear additive Hebbian learning which has been applied in the context of this work and restrains the accumulation of the synaptic weights

following the geometric series:

$$A_{i,j} = g(q, \sum_{\mu=1}^{M} u_i^\mu v_j^\mu - 1) \in [0, 2] \tag{2.47}$$

with

$$g(q,n) = \begin{cases} n+1, & q = 1 \\ \frac{q^{n+1}-1}{q-1}, & otherwise \end{cases} \tag{2.48}$$

where $q = \frac{1}{2}$.

Once the M pairs of pattern are stored the associative memory can be addressed with a pattern $\tilde{u}$ in order to retrieve the output pattern $\tilde{v}$. If the address pattern $\tilde{u}$ sufficiently resembles a stored pattern u^μ the output pattern $\tilde{v}$ will ideally equal v^μ. Thus the associative memory can be addressed with noisy versions of the stored patterns and is nevertheless able to retrieve the correct output pattern. A common way of obtaining the output pattern is the application of a threshold Θ to the outer product of the address pattern $\tilde{u}$ with the memory matrix A:

$$\tilde{v_j} = \begin{cases} 1 & : \sum_{i=1}^{n} \tilde{u_i} A_{i,j} \geq \Theta \\ 0 & : otherwise \end{cases} \tag{2.49}$$

The threshold Θ is determined by the number of ones in the address pattern $\tilde{u}$:

$$\Theta = \sum_{i=1}^{n} \tilde{u_i}. \tag{2.50}$$

A more general way of retrieving the output pattern determines the maximum of the outer product of the address pattern $\tilde{u}$ with the memory matrix A:

$$\tilde{v_j} = \begin{cases} 1 & : \sum_{i=1}^{n} \tilde{u_i} A_{i,j} \geq \max_{j=1,2,\ldots,m}(\sum_{i=1}^{n} \tilde{u_i} A_{i,j}) - \theta \\ 0 & : otherwise \end{cases} \tag{2.51}$$

where $\theta \geq 0$ controls the number of ones in the retrieved pattern $\tilde{v}$: the higher θ is chosen the more ones contains the output pattern. Thus the most restrictive case is $\theta = 0$.

When using associative memories for classification the patterns to be classified are stored in a hetero-associative memory where the data forms the input patterns and the one-out-of-m coded classes constitute the output patterns. Hence the positions of the ones in the output patterns corresponds to the associated classes. If in the retrieval phase the output vector contains more than a single one a selection needs to be performed identifying the most likely class.

2.6 Discussion

There are various types of neural networks. Criteria for the differentiation of neural networks are the network topology, the characteristics of the used artificial neurons, the utilised training strategy and the way how the activation of the hidden layer is calculated. Either the scalar product of the input vector and a weight vector or the distance between the input vector and a prototype can serve as input to the neurons in the hidden layer. Multi-Layer Perceptrons are an example of the former class of artificial neural networks. Radial Basis Function networks and Learning Vector Quantisation networks form examples of the latter. All used networks are prototype-based networks that learn a representation of the training data in the feature space. The used networks also differ in their architecture. The RBF networks are three-layered feed-forward networks, consisting of an input layer, a hidden layer and an output layer with unidirectional connections between the layers, whereas the LVQ networks and the k-means networks are two-layered networks consisting of an input layer and a hidden layer performing a clustering task. Within an RBF network the neurons in the hidden layer use radial basis functions as transfer functions and the neurons in the output layer have linear transfer functions. The LVQ networks and the k-means identify the classifier output according to the k-nearest neighbour rule. These networks are competitive networks and the learning is performed utilising a winner-takes-all strategy. The learning in RBF networks is performed by means of gradient descent methods. The k-means network and the LVQ network differ insofar as the former one is trained using an unsupervised learning algorithm where as the latter one is trained supervised.

A major advantage of the k-NN classifiers and fuzzy k-NN classifiers is the marginal number of parameters to be specified despite showing reasonable and robust classification performance. The 1-NN classifier does not require parametrisation at all. The k-NN classifier only requires the specification of the number k of prototypes included in the decision process. For the fuzzy k-NN classifier only the additional parameter α needs to be selected. Moreover the provision of membership values is an eligible property of the fuzzy k-NN classifier. Another advantage is the fact that no training is required. A disadvantage of this approach is the long computational time required in the classification phase.

The outputs of an RBF network can be interpreted as an estimation of the a posteriori probabilities. A disadvantage of RBF networks is the large number of parameters that have to be adjusted and which have a crucial impact on the classification performance. The time required for classification is shorter than that required by the nearest neighbour classifiers as the number of prototypes is considerably lower. A disadvantage is the extensive training time.

A rather different approach are associative memories. They store the training

data by means of sparse binary patterns. Thus they require sparse binary representations of the data to be classified. These codes usually can not be generated straightforwardly. If the codes are generated in a proper way associative memories show a high and robust classification performance.

In the scope of this work different network types were used fulfilling diverse tasks. They were used for the hierarchy generation and for initialising classifiers and they were used as classifiers. The k-means algorithm was used within the hierarchy generation phase. LVQ networks and k-means clustering was utilised to initialise RBF networks. As classifiers within the hierarchy RBF networks, k-NN classifiers and fuzzy k-NN classifiers were used. Hetero-associative memories were employed as classifiers in connection with sparse similarity preserving codes generated from the classifier hierarchies.

3 Uncertainty

Dealing with uncertainty is an important concept in artificial intelligence. There are several approaches addressing this problem such as probability theory, fuzzy set theory, possibility theory and belief theory. Due to its straightforward applicability to the domain of hierarchical neural networks, belief theory was chosen above other frameworks for representing uncertainty.

In the following several concepts for representing and dealing with uncertainty are briefly described. For reasons of simplicity only finite universes Ω are considered. This is justified insofar as within the scope of classification problems, which are the domain of application in this work, the universe is represented by the finite set of classes $\Omega = \{1, 2, ..., l\}$.

3.1 Probability Theory

The probability theory is a well-established approach to pattern recognition. Requirements for the applicability of this approach are the posing of the decision problem in terms of probabilities and the availability of all relevant probability values. The Bayes' rule is used within this framework for the fusion of information.

The sample space $\Omega = \{\omega_1, ..., \omega_l\}$ is a finite set of l mutually exclusive atomic hypotheses. Within the scope of the probability theory framework these hypotheses are interpreted as the possible outcomes of a random experiment and they are called elementary events. A subset $A \subseteq \Omega$ is called an event. The powerset of Ω is denoted by 2^Ω and contains all possible subsets of Ω.

A function $p : \Omega \rightarrow [0, 1]$ is called probability distribution if $\sum_{\omega_i \in \Omega} p(\omega_i) = 1$. Given such a probability distribution p the corresponding probability measure

$P : 2^{\Omega} \rightarrow [0, 1]$ is uniquely defined by:

- $P(\{\omega_i\}) = p(\omega_i)$
- $P(A) \in [0, 1], \forall A \subseteq \Omega$
- $P(\Omega) = 1$
- $A \cap B = \emptyset \Rightarrow P(A \cup B) = P(A) + P(B)$ (additivity)

The probability $P(A)$ specifies the likeliness of the occurrence of the event A. In case of ignorance, i.e. if nothing is known about the probability of the elementary events $\omega_i \in \Omega$, each elementary event is assigned the probability $p(\omega_i) = \frac{1}{l}, \forall \omega_i \in \Omega$. This is also the case if all elements are equally probable.

A probability measure P shows the following additional properties:

- $P(\emptyset) = 0$
- $A \subseteq B \Rightarrow P(A) \leq P(B)$
- $P(A) = 1 - P(\overline{A})$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

The conditional probability[1] $P(A|B)$ of an event A given B is defined as

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)} \tag{3.1}$$

with $P(B) > 0$.

Applied to the problem of classification the following equation results

$$p(\omega_i|x) = \frac{p(x|\omega_i)p(\omega_i)}{p(x)} = \frac{p(x|\omega_i)p(\omega_i)}{\sum_{i=1}^{l} p(x|\omega_i)p(\omega_i)} \tag{3.2}$$

where $p(\omega_i)$ is the a priori probability of the event or class ω_i, $p(\omega_i|x)$ is the a posteriori probability of the correct class being ω_i given x, $p(x|\omega_i)$ is the class conditional probability of x for the class ω_i and x is the information provided by the classifier. $p(x)$ acts as a normalisation factor ensuring that the a posteriori probabilities sum up to one.

[1] If $P(B) = 0$ the conditional probability $P(A|B)$ is not defined. If $A \subseteq B$ the conditional probability yields $P(A|B) = \frac{P(A)}{P(B)}$. Moreover it holds $P(A|A) = P(A)$

By means of the Bayesian combination rule information from different sources x_1 and x_2 can be combined.

$$p(\omega_i|x_1, x_2) = \frac{p(x_1|\omega_i, x_2)p(x_1|\omega_i)p(\omega_i)}{p(x_1, x_2)}. \tag{3.3}$$

In case of independent sources the rule can be simplified to

$$p(\omega_i|x_1, x_2) = \frac{p(x_2|\omega_i)p(x_2|\omega_i)p(\omega_i)}{p(x_1)p(x_2)}. \tag{3.4}$$

The combination rule for more than two sources is given by

$$p(\omega_i|x_1, ..., x_k) = \frac{p(x_k|\omega_i, x_1, ..., x_{k-1})...p(x_1|\omega_i)p(\omega_i)}{p(x_1, ..., x_k)}) \tag{3.5}$$

where k is the number of sources to be fused.

In case of independence between the sources the rule can analogously be reduced to

$$p(\omega_i|x_1, ..., x_k) = \frac{p(\omega_i)\prod_{j=1}^{k} p(x_j|\omega_i)}{\prod_{j=1}^{k} p(x_j)}. \tag{3.6}$$

This fusion operator is only applicable if the k sources are completely reliable. If this is not the case, the fusion operator has to be adapted incorporating knowledge about the reliability of the individual sources [71]. The reliability of sources is often estimated utilising the classification performance of the source.

3.2 Fuzzy Set Theory

The fuzzy set theory [89] is a generalisation of the classical set theory which allows the representation and handling of imprecise knowledge. An essential feature is the possibility of specifying partial membership.

A set A in the classical sense can be defined by listing all elements belonging to this set $A = \{a_1, a_2, ..., a_n\}$. The set A can also be represented by its *characteristic function* $\mu_A : \Omega \rightarrow \{0, 1\}$, that takes the value one if ω_i is an element of the set A and zero otherwise:

$$\mu_A(\omega_i) = \begin{cases} 1, & \omega_i \in A \\ 0, & \omega_i \notin A \end{cases} \tag{3.7}$$

Sets whose characteristic function $\mu_A : \Omega \rightarrow [0,1]$ can take any value in the interval $[0,1]$ is called a fuzzy set A where $\mu_A(\omega_i)$ represents the grade of membership of the element $\omega_i \in \Omega$ to the set A and μ_A is called *membership function* or degree of membership. Hence the classic sets or so called crisp sets are a special case of fuzzy sets.

In the following the basic features of fuzzy sets as well as the operations on fuzzy sets are briefly described.

The set of all fuzzy sets of Ω is denoted by $\mathcal{F}(\Omega)$. Let $A \in \mathcal{F}(\Omega)$ be a fuzzy set defined on Ω. The degree of membership of an element $\omega_i \in \Omega$ to the fuzzy set A is denoted by $\mu_A(\omega_i)$.

The *support* $supp(A)$ of the fuzzy set A is the crisp set that contains all elements ω_i of Ω which belong to A with a positive degree of membership:

$$supp(A) = \{\omega_i \in \Omega | \mu_A(\omega_i) > 0\}. \tag{3.8}$$

The *kernel* $ker(A)$ of the fuzzy set A is the crisp set that contains all elements ω_i of Ω which belong to the fuzzy set A with degree of membership one:

$$ker(A) = \{\omega_i \in \Omega | \mu_A(\omega_i) = 1\}. \tag{3.9}$$

A set A is a crisp set iff $A = supp(A) = ker(A)$.

The *boundary* $bnd(A)$ of the fuzzy set A is the crisp set that contains all elements ω_i of Ω which belong to the support $supp(A)$ but not to the kernel $ker(A)$ of the fuzzy set A:

$$bnd(A) = suoo(A) \setminus ker(A) = \{\omega_i \in \Omega | 0 < \mu_A(\omega_i) < 1\}. \tag{3.10}$$

The boundary of a crisp set is the empty set.

The *height* $hgt(A)$ of the fuzzy set A is the highest degree of membership with which an element ω_i of Ω belongs to the fuzzy set A, i.e. it is the highest value of the characteristic function μ_A:

$$hgt(A) = \max_{\omega_i \in \Omega} \mu_A(\omega_i). \tag{3.11}$$

If the height $hgt(A)$ of a fuzzy set A equals one it is called *normal*, otherwise it is called *subnormal*. Thus a fuzzy set A is *normal* if at least one element ω_i of Ω exists that belongs absolutely to the fuzzy set A, i.e. the degree of membership of the element ω_i equals one.

A crisp set is always normal.

The cardinality $|A|$ of a fuzzy set A is the overall degree with which the elements ω_i of Ω belong to the fuzzy set A:

$$|A| = \sum_{\omega_i \in \Omega} \mu_A(\omega_i). \tag{3.12}$$

The cardinality of a crisp set is the number of its elements.

Two fuzzy sets $A \in \mathcal{F}(\Omega)$ and $B \in \mathcal{F}(\Omega)$ that are both defined on Ω are *equal* if their membership functions μ_A and μ_B are equal, i.e. they take the same value for each element $\omega_i \in \Omega$:

$$A = B \Leftrightarrow \mu_A(\omega_i) = \mu_B(\omega_i), \forall \omega_i \in \Omega. \tag{3.13}$$

Two fuzzy sets $A \in \mathcal{F}(\Omega)$ and $B \in \mathcal{F}(\Omega)$ that are both defined on Ω are regarded *similar* if their kernels and their supports are equal:

$$A \approx B \Leftrightarrow ker(A) = ker(B) \text{ and } supp(A) = supp(B). \tag{3.14}$$

A fuzzy set $A \in \mathcal{F}(\Omega)$ is *included* in the fuzzy set $B \in \mathcal{F}(\Omega)$ if all elements ω_i of Ω that belong to the set A also belong to the set B at least to the same degree:

$$A \subseteq B \Leftrightarrow \mu_A(\omega_i) \leq \mu_B(\omega_i), \forall \omega_i \in \Omega. \tag{3.15}$$

The *complement* $\overline{A}$ of a fuzzy set $A \in \mathcal{F}(\Omega)$ is givn by:

$$\mu_{\overline{A}} = 1 - \mu_A, \forall \omega_i \in \Omega. \tag{3.16}$$

The complement fulfills the De Morgan's laws known from classical set theory, i.e. $\overline{A \cap B} = \overline{A} \cup \overline{B}$ and $\overline{A \cup B} = \overline{A} \cap \overline{B}$ as well as the involution $\overline{\overline{A}} = A$. In contrast to the classical set theory the equations $A \cap \overline{A} = \emptyset$ and $A \cup \overline{A} = \Omega$ are no longer valid when using fuzzy sets.

The *intersection* $i(A, B)$ of two fuzzy sets $A \in \mathcal{F}(\Omega)$ and $B \in \mathcal{F}(\Omega)$ is defined via a mapping $i : [0,1] \times [0,1] \rightarrow [0,1]$ such that $\mu_{A \cap B}(\omega_i) := i(\mu_A(\omega_i), \mu_B(\omega_i)), \forall \omega_i \in \Omega$. The function [2] $i : [0,1]^2 \rightarrow [0,1]$ is required to fulfill the following conditions

[2] Formally speaking the function i is realised by t-norms[73] [88] [17] [85].

$\forall a, b, c \in [0, 1]$:

- i_1: $i(a, 1) = a$ (closure condition)
- i_2: $b \leq c \Rightarrow i(a, b) \leq i(a, c)$ (monotonicity)
- i_3: $i(a, b) = i(b, a)$ (commutativity)
- i_4: $i(a, i(b, c)) = i(i(a, b), c)$ (associativity)
- i_5: i is a continuous function (continuity)
- i_6: $i(a, a) < a$ (sub-idempotency)
- i_7: $a_1 < a_2$ and $b_1 < b_2 \Rightarrow i(a_1, b_1) < i(a_2, b_2)$ (strict monotonicity)

Note that i is not uniquely determined by these conditions.

The conditions i_1 - i_4 are mandatory conditions whereas the conditions i_5 - i_7 are additional conditions limiting the amount of possible fuzzy intersections. The axioms i_2 and i_3 ensure that the intersection will not increase the degree of membership when there is a decrease of the degree of membership of the fuzzy set A or B. Moreover condition i_3 ensures the symmetry of the intersection. Condition i_4 ensures that any number of fuzzy sets can be intersected regardless of the order. Condition i_5 ensures that a small change of the degree of membership of the fuzzy set A or B will not yield a large change of the degree of membership of the intersection. Condition i_6 ensures that in case of identical degrees of membership a of set A and B the degree of membership of the intersection $A \cap B$ does not surpass the degree of membership a. This requirement is weaker than idempotency ($i(a, a) = a$).

The most common functions fulfilling these conditions are $\forall a, b \in [0, 1]$:

- $i_s(a, b) = \min(a, b)$ (standard intersection)
- $i_p(a, b) = a \cdot b$ (algebraic product)
- $i_{di}(a, b) = \max(0, a + b - 1)$ (limited difference)
- $i_d(a, b) = \begin{cases} a, & b = 1 \\ b, & a = 1 \\ 0, & otherwise \end{cases}$ (drastic intersection)

These functions are related such that
$i_d(a, b) \leq i_{di}(a, b) \leq i_p(a, b) \leq i_s(a, b), \forall a, b \in [0, 1]$.

The *union* $u(A, B)$ of two fuzzy sets $A \in \mathcal{F}(\Omega)$ and $B \in \mathcal{F}(\Omega)$ is defined via a mapping $u : [0,1] \times [0,1] \rightarrow [0,1]$ such that $\mu_{A \cup B}(\omega_i) = u(\mu_A(\omega_i), \mu_B(\omega_i)), \forall \omega_i \in \Omega$. The function [3] $u : [0,1]^2 \rightarrow [0,1]$ is required to satisfy the following conditions $\forall a, b, c \in [0,1]$:

- u_1: $u(a, 0) = a$ (closure condition)
- u_2: $b \leq c \Rightarrow u(a, b) \leq u(a, c)$ (monotonicity)
- u_3: $u(a, b) = u(b, a)$ (commutativity)
- u_4: $u(a, u(b, c)) = u(u(a, b), c)$ (associativity)
- u_5: u is a continuous function (continuity)
- u_6: $u(a, a) > a$ (super-idempotency)
- u_7: $a_1 < a_2$ and $b_1 < b_2 \Rightarrow u(a_1, b_1) < u(a_2, b_2)$ (strict monotonicity)

The justification of the conditions $u_1 - u_7$ is analogous to the justifications of the conditions $i_1 - i_7$. Compared to the conditions $i_1 - i_7$ only the boundary conditions specified by the conditions i_1 and u_1 and the sub-idempotency and super-idempotency conditions specified by the axioms i_6 and u_6 are different.

The most common functions fulfilling these conditions are $\forall a, b \in [0,1]$:

- $u_s(a, b) = \max(a, b)$ (standard union)
- $u_{sum}(a, b) = a + b - a \cdot b$ (algebraic sum)
- $u_b(a, b) = \min(1, a + b)$ (boundary sum)
- $u_d(a, b) = \begin{cases} a, & b = 0 \\ b, & a = 0 \\ 1, otherwise \end{cases}$ (drastic union)

These functions are related such that
$u_s(a, b) \leq u_{sum}(a, b) \leq u_b(a, b) \leq u_d(a, b), \forall a, b \in [0,1]$.

The *α-cut* A_α of a fuzzy set A is the crisp set $A_\alpha \in \Omega$ that contains all elements ω_i of Ω that belong to the fuzzy set A with at least the degree of membership $\alpha \in [0,1]$.

$$A_\alpha = \{\omega_i \in \Omega | \mu_A(\omega_i) \geq \alpha\} \tag{3.17}$$

[3] Formally speaking the function i is realised by s-norms or t-conorms.

It holds $\alpha_1 \leq \alpha_2 \Rightarrow A_{\alpha_1} \subseteq A_{\alpha_2}$. Moreover the α-cut fulfills the property $A \cap B \Rightarrow A_\alpha \cap B_\alpha$. Furthermore the α-cut is commutative with respect to intersection and union, i.e. $(A \cap B)_\alpha = A_\alpha \cap B_\alpha$ and $(A \cup B)_\alpha = A_\alpha \cup B_\alpha$. Furthermore, it holds $A_{\alpha=0} = \Omega$ and $A_{\alpha=1} = ker(A)$.

3.3 Possibility Theory

Possibility theory is an extension of the fuzzy set theory and is an alternative to probability theory. It is a framework for dealing with non-probabilistic concepts of uncertainty. This framework provides means for representing both inaccuracy and uncertainty. Inaccurate information can be represented utilising the concept of fuzzy sets. Uncertainty can be quantified by means two measures: possibility and necessity.

A possibility distribution $\pi : \Omega \rightarrow [0, 1]$ can be interpreted as the membership function of a normal fuzzy set.

The universe of discourse $\Omega = \{\omega_1, ..., \omega_l\}$ is a finite set of l mutually exclusive atomic hypotheses. The powerset of Ω is denoted by 2^Ω and contains all possible subsets of Ω.

The *possibility* Π over a universe of discourse Ω is a function $\Pi : 2^\Omega \rightarrow [0, 1]$ such that $\Pi(\{\omega_i\}) = \pi(\omega_i)$. It is a measure for the likeliness of a hypotheses or a set of hypotheses (so called events), where possibility $\Pi(A) = 0$ means that the event A is completely impossible and possibility $\Pi(A) = 1$ expresses that the event A is completely possible. The relation to probability theory is stated by the fact that an event that is probable is also possible. This is expressed by the inequality $P(A) \leq \Pi(A)$. The possibility fulfills the following conditions:

$$\Pi(\emptyset) = 0 \tag{3.18}$$

$$\Pi(\Omega) = 1 \tag{3.19}$$

$$\Pi(A \cup B) = \max(\Pi(A), \Pi(B)), \forall A, B \in 2^\Omega \tag{3.20}$$

$$\Pi(\bigcup_{i=1,2,...} A_i) = \max_{i=1,2,...} (\Pi(A_i)), \forall A_i \in 2^\Omega \tag{3.21}$$

The fact that no possibility can be assigned to the empty set $\emptyset$ expresses the closed world assumption, i.e. the universe of discourse Ω is assumed to be an exhaustive description of the world and no belief is given to elements outside Ω.

The condition that the universe of discourse Ω is completely possible specifies the assumption that there is no conflict between the evidences from which the possibility Π was built. This implicates that at least one element in Ω is completely possible. The definition of the possibility of the union of two possibilities implies that the occurrence of one of two events obtains the same possibility as the occurrence of the most possible event of those two. Moreover it can easily be deduced that at least the event $A \in 2^{\Omega}$ or its complement $\overline{A}$ must be completely possible, i.e. $\max(\Pi(A), \Pi(\overline{A})) = 1, \forall A \in 2^{\Omega}$ and hence it follows $\Pi(A) + \Pi(\overline{A}) \geq 1, \forall A \in 2^{\Omega}$. Thus complete ignorance about the occurrence of A can be expressed when the event A as well as its complement $\overline{A}$ are both completely possible, i.e. $\Pi(A) = 1$ and $\Pi(\overline{A}) = 1$. The certain occurrence of A can be specified when its complement $\overline{A}$ is completely impossible as this implies that the event A is completely possible, i.e. $\Pi(A) = 1$ and $\Pi(\overline{A}) = 0$. There is an interaction between the possibility $\Pi(A)$ of the event A and the possibility $\Pi(\overline{A})$ of its complement $\overline{A}$: If either of them is not completely possible, i.e. the possibility is less than one, the other must be completely possible, i.e. the possibility must equal one.

The possibility measure on finite or countably infinite sets is determined by the possibility of the atomic elements it is composed of:

$$\Pi(A) = \max_{\omega_i \in A} \Pi(\omega_i). \tag{3.22}$$

As the possibility of a union can be determined from the possibility of each component, the possibility measure is compositional with respect to the union operator.

Regarding the intersection operator the following proposition can be derived:

$$\Pi(A \cap B) \leq \min(\Pi(A), \Pi(B)), \forall A, B \in 2^{\Omega}. \tag{3.23}$$

This implies that despite two events A and B are possible, i.e. $\Pi(A) > 0$ and $\Pi(B) > 0$, their coinstantaneous occurrence might be impossible, i.e. $\Pi(A \cap B) = 0$.

In order to be able to completely describe the uncertainty of an event A the possibility theory utilises the necessity measure N which is a complement of the possibility measure Π.

The *necessity* N over a universe of discourse Ω is a function $N : 2^{\Omega} \rightarrow [0, 1]$. It specifies the degree with which the occurrence of an event A is certain. It fulfills

the following conditions:

$$N(\emptyset) = 0 \tag{3.24}$$

$$N(\Omega) = 1 \tag{3.25}$$

$$N(A \cap B) = \min(N(A), N(B)), \forall A, B \in 2^{\Omega} \tag{3.26}$$

$$N(\bigcap_{i=1,2,\ldots} A_i) = \min_{i=1,2,\ldots} (N(A_i)), \forall A_i \in 2^{\Omega} \tag{3.27}$$

These conditions imply that N is monotone:

$$A \subseteq B \Rightarrow N(A) \leq N(B), \forall A, B \in 2^{\Omega}. \tag{3.28}$$

The following property of the union operator can be derived:

$$N(A \cup B) \geq \max(N(A), N(B)), \forall A, B \in 2^{\Omega}. \tag{3.29}$$

Regarding the necessity of on event A and the necessity of its complement $\overline{A}$ it holds $\min(N(A), N(\overline{A})) = 0$ and $N(A) + N(\overline{A}) \leq 1$. Moreover there exits a relation between the necessity of the event A and the necessity of its complement $\overline{A}$ such that if either of them is greater than zero the other must be zero and if one of them equals zero, the other can take any value.

The necessity function can be constructed from the possibility distribution as follows:

$$N(A) = \min_{\omega_i \notin A}(1 - \Pi(\omega_i)). \tag{3.30}$$

There exists a duality between possibility Π and necessity N such that:

$$N(A) = 1 - \Pi(\overline{A}) \tag{3.31}$$

where $\overline{A}$ is the complement of A with respect to Ω.

This duality expresses the fact that A is the more certain the more $\overline{A}$ is impossible. The greater the necessity of the event A is, the less is its complement possible, i.e. the more certain is the occurrence of the event A. The occurrence of an event A is completely certain if and only if its complement $\overline{A}$ is completely impossible,

which implicates that the event A is completely possible: $N(A) = 1 \Leftrightarrow \Pi(\overline{A}) = 0 \Rightarrow \Pi(A) = 1$.

This duality is also expressed in the conditions $\Pi(A) \geq N(A)$ and $\max(\Pi(A), 1 - N(A)) = 1$. These conditions imply that any event A whose occurrence is even slightly certain is completely possible: $N(A) > 0 \Rightarrow \Pi(A) = 1$. Furthermore they implicate that if an event is not completely possible but only relative possible there is no certainty about the occurrence of this event at all: $\Pi(A) > 1 \Rightarrow N(A) = 0$.

Certain pairs of possibility and necessity can be interpreted as follows:

- $N(A) = 1 \Rightarrow \Pi(A) = 1$: The event A is certainly true.
- $\Pi(A) = 0 \Rightarrow N(A) = 0$: The event A is certainly false.
- $N(A) = 0$ and $\Pi(A) = 1$: This states the indetermination that nothing is known about the event A.

There are various methods for fusing individual possibility distributions. These fusion strategies differ considerably in their behaviour, outcome and applicability. Examples for fusion strategies are:

- Conjunctive fusion:
  $\pi_{conj}(\omega_i) = \min(\pi_1(\omega_i), \pi_2(\omega_i)), \forall \omega_i \in \Omega$
- Renormalised conjunctive fusion:
  $\pi_{renorm}(\omega_i) = \frac{\pi_{conj}(\omega_i)}{h(\pi_1,\pi_2)} = \frac{\min(\pi_1(\omega_i),\pi_2(\omega_i))}{\max_{\omega_i \in \Omega}(\min(\pi_1(\omega_i),\pi_2(\omega_i)))}, \forall \omega_i \in \Omega$
- Disjunctive fusion:
  $\pi_{disj}(\omega_i) = \max(\pi_1(\omega_i), \pi_2(\omega_i)), \forall \omega_i \in \Omega$
- Adaptive fusion:
  $\pi_{ad}(\omega_i) = \max(\pi_{renorm}(\omega_i), \min(1 - h(\pi_1, \pi_2), \pi_{disj}(\omega_i))), \forall \omega_i \in \Omega$
- Quantified adaptive fusion:
  $\pi_{adq}(\omega_i = \max(\pi_{renorm_{(n)}}, \min(1 - h(n), \pi_{disj_{(m)}})), \forall \omega_i \in \Omega$

where k is the total number of sources to be fused, J is any subset of sources, $h(J) = \max_{\omega_i \in \Omega}(\min_{j \in J}(\pi_j(\omega_i)))$ is the degree of consensus of the subset J of the sources, $m = \max\{|J|, h(J) = 1\}$ is the pessimistic estimation of the number of reliable sources and $n = \max\{|J|, h(J) > 0\}$ is the optimistic estimation of the number of reliable sources.

The conflict c between the individual possibility distributions to be fused is expressed by $c = 1 - h(\pi_1, \pi_2)$. Thus $h(\pi_1, \pi_2)$ is a measure for the consensus between the sources providing the different possibility distributions. The sources

are considered reliable if $h(\pi_1, \pi_2) = 1$, i.e. no conflict exists between the sources. They are regarded as totally contradictory if $h(\pi_1, \pi_2) = 0$. If $0 < h(\pi_1, \pi_2) < 1$ there exists some conflict between the sources which is the stronger the lower $h(\pi_1, \pi_2)$ is.

3.4 Belief theory

The Dempster-Shafer evidence theory or belief theory is a mathematical theory of evidence based on belief functions and plausible reasoning. It provides means of representing and combining measures of evidence.

Major advantages of the belief theory are the possibility to discriminate between ignorance and uncertainty, i.e. a lack of knowledge caused e.g. by unknown objects can be distinguished from doubtful situations where several alternatives are almost equally likely, the ability to easily represent evidence at different levels of abstraction, i.e. it allows for associating evidence to intervals and sets, and the possibility to combine information from multiple sources and to give an indiction of the conflict between the individual sources. The theory allows for only representing the actual knowledge without enforcing more detailed propositions in case of ignorance.

This theory is a generalisation of the Bayesian probability theory, but is more flexible than probability theory in case of incomplete knowledge and when dealing with uncertainty and ignorance.

In the following the basic concepts of the Dempster-Shafer evidence theory are briefly explained.

3.4.1 Basic Concepts

The *frame of discernment* is a finite set of l mutually exclusive atomic hypotheses $\Omega = \{\omega_1, ..., \omega_l\}$ representing the universe of discourse. Let 2^Ω denote the power set of Ω.

In Dempster-Shafer theory there are three principal functions:

1. the basic probability assignment function or mass function m
2. the belief function Bel
3. the plausibility function Pl

A *basic probability assignment* or mass function m over a frame of discernment

Ω is a function $m : 2^\Omega \to [0, 1]$ that satisfies the following two conditions:

$$m(\emptyset) = 0 \tag{3.32}$$

and

$$\sum_{A \subseteq \Omega} m(A) = 1. \tag{3.33}$$

By definition, the mass of the empty set $\emptyset$ is zero, i.e. no evidence is assigned to the impossible event $\emptyset$ and the masses of the remaining elements of the power set 2^Ω sum up to one, i.e. the total evidence of a source sums up to a total of one. The subsets $A \subseteq \Omega$ with $m(A) > 0$ are called focal elements of m.

The fact that no mass can be assigned to the empty set $\emptyset$ expresses the closed world assumption. The frame of discernment Ω is regarded as being a complete description of the world. Hence no evidence pointing to some hypothesis not in Ω is allowed.

The belief $Bel(A)$ is defined as the sum of all masses of the proper subsets of the set A. With m being a basic probability assignment the *belief function* $Bel : 2^\Omega \to [0, 1]$ is defined as follows:

$$Bel(A) = \sum_{B:B \subseteq A} m(B). \tag{3.34}$$

The plausibility $Pl(A)$ is defined as the sum of all masses of the sets B that intersect the set A. If m is a basic probability assignment the *plausibility function* $Pl : 2^\Omega \to [0, 1]$ is defined as:

$$Pl(A) = \sum_{B:A \cap B \neq \emptyset} m(B). \tag{3.35}$$

Basic probabilities cannot only be assigned to atomic hypothesis $\omega_i \in \Omega$ but also to sets $A \subseteq \Omega$ of atomic hypotheses. The basic probability assignment $m(A)$ specifies the degree of belief that is assigned to exactly the set $A \subseteq \Omega$ and not to any subset of A. The value of $m(A)$ is only related to the set A. It makes no additional assertion about any subset of A as each subset of A has its own mass. In particular unlike to probability theory it is not required to specify explicitly the basic probability assignment of each element of the frame of discernment Ω. If A is a non-atomic hypothesis $m(A)$ reflects ignorance or partial knowledge as it is not possible to further subdivide the belief in A among the subsets of A. The mass $m(A)$ specifies the belief supporting A. It could support any subset of A given further information indicating this, but the available information justifies only the support of A. The belief $Bel(A)$ also takes into account the degree of belief assigned to events that are subsets of A. It specifies the total belief

that is certainly assigned to A. The plausibility $Pl(A)$ is a measure for the belief that could potentially be assigned to A, i.e. the belief assigned to some set B that is consistent with A. It specifies the degree of belief that is not assigned to events that falsify A. The belief interval $[Bel(A), Pl(A)]$ represents the complete information about the degree of belief in A. This probability interval contains the actual probability $P(A)$ of the set A: $Bel(A) \leq P(A) \leq Pl(A), \forall A \subseteq \Omega$.

The vacuous basic probability assignment m_{vac} is used to represent the extreme case of total ignorance. It is specified by $m_{vac}(\Omega) = 1$ and $m_{vac} = 0, \forall A \subset \Omega$. The corresponding plausibilities are all zero and the corresponding beliefs are all equal to one. Thus the resulting belief intervals are $[0, 1]$ for all subsets $A \subset \Omega$.

If an event ω_i is known to be definitely true this is represented by $m(\omega_i) = 1$ and $m(A) = 0, \forall A \neq \omega_i \subseteq \Omega$. If an event $A \subset \Omega$ is known to be definitely false this is represented by assigning a basic probability assignment $m(A)$ corresponding to the belief $Bel(\overline{A}) = 1$ or the plausibility $Pl(A) = 0$.

If evidences are available for all elementary events ω_i the basic probability assignment m is reduced to a classical probability. Then $\sum_{\omega_i \in \Omega} m(\omega_i) = 1$ holds and $m(A) = 0, \forall A \in \Omega$ with $|A| > 1$.

Both Bel and Pl are monotone functions. If $A \subseteq B$ then $Bel(A) \leq Bel(B)$ and $Pl(A) \leq Pl(B)$. Furthermore both functions are non-additive.

There exists a duality between $Bel(A)$ and $Pl(A)$ such that $Pl(A) = 1 - Bel(\overline{A})$ and $Bel(A) = 1 - Pl(\overline{A})$ where $\overline{A}$ is the complement of A with respect to Ω.

Multiple sources that provide different assessments for the same frame of discernment can be combined within the framework of belief theory. The basic probability assignments m_1 and m_2 of two independent sources can be combined via the orthogonal sum $m_{1,2} = m_1 \oplus m_2$ which is given by:

$$m_{1,2}(C) = K^{-1} \sum_{A,B:A \cap B = C} m_1(A) \cdot m_2(B) \tag{3.36}$$

where K is a scale factor and $1 - K$ is a measure for the conflict between the two sources. The scale factor K is given by:

$$K = 1 - \sum_{A,B:A \cap B = \emptyset} m_1(A) \cdot m_2(B) = \sum_{A,B:A \cap B \neq \emptyset} m_1(A) \cdot m_2(B). \tag{3.37}$$

The orthogonal sum $m_1 \oplus m_2$ does only exists if $K \neq 0$ and the result $m_{1,2}$ is then a basic probability assignment. Otherwise the two sources are said to be totally contradictory.

The operator $\oplus$ is commutative and associative. Thus the orthogonal sum can be generalised to combine evidence from multiple sources by sequentially using

3.36. The combination of k sources is defined as:

$$m_{1,2,...,k} = m_1 \oplus m_2 \oplus ... \oplus m_k = \bigoplus_{i=1,2,...,k} m_i. \tag{3.38}$$

The combined basic probability assignment $m_{1,2,...,k}$ exists if at least two of the k sources are not totally contradictory. It is then given by

$$m_{1,2,...,k}(C) = K^{-1} \sum_{A_1 \cap ... \cap A_k = C} \prod_{i=1}^{k} m_i(A_i) \tag{3.39}$$

where

$$K = 1 - \sum_{A_1 \cap ... \cap A_k = \emptyset} \prod_{i=1}^{k} m_i(A_i) = \sum_{A_1 \cap ... \cap A_k \neq \emptyset} \prod_{i=1}^{k} m_i(A_i). \tag{3.40}$$

3.4.2 Transferable Belief Model

The transferable belief model [79] is an interpretation of the Dempster-Shafer theory of evidence.

Within the transferable belief model positive masses can be assigned to the empty set $\emptyset$. The mass function m is then just required to fulfill the following condition:

$$\sum_{A \subseteq \Omega} m(A) = 1. \tag{3.41}$$

The rule for combining evidences from different sources is changed to:

$$m_{1,2}(C) = \sum_{A,B:A \cap B = C} m_1(A) \cdot m_2(B), \forall C \subseteq \Omega \tag{3.42}$$

entailing unnormalised basic probability assignments [78].

A high value for the mass of the empty set $\emptyset$ indicates a high conflict between the sources.

The possible assignment of positive masses to the impossible event $\emptyset$ reflects the fact that the frame of discernment Ω might not include all possible hypotheses. This assumption is referred to as open-world-assumption in contrast to the closed-world-assumption where Ω is assumed to completely describe all possible states and thus no belief is assigned to the impossible event $\emptyset$ and normalisation is required when combining evidences.

The orthogonal sum is a conjunctive fusion method and is applicable if the sources to be fused are reliable. If this can not be guaranteed a disjunctive fusion method

is appropriate. This fusion method is realised by utilising the union operator instead of the intersection operator:

$$m_{1\cup 2}(C) = \sum_{A,B:A\cup B=C} m_1(A) \cdot m_2(B), \forall C \subseteq \Omega. \tag{3.43}$$

For basic probability assignments from k sources the fusion method is given by

$$m_{1\cup 2\cup \ldots \cup k}(C) = \sum_{A_1\cup \ldots \cup A_k=C} \prod_{i=1}^{k} m_i(A_i). \tag{3.44}$$

3.5 Comparison of Theories for Representing Uncertainty

A major advantage of the fuzzy set theory is the capability of modelling imprecise or vague knowledge. However, it is not possible to handle both imprecision and uncertainty in the same framework. The possibility theory framework provides means for dealing with uncertainties on imprecise knowledge as it allows for reasoning on inaccurate knowledge.

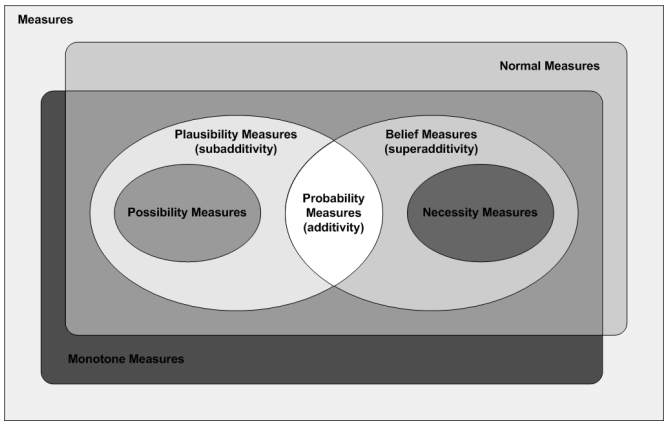

Figure 3.1: *Typology of the different uncertainty measures [33].*

Probability theory and possibility bear resemblance as they are both based on set theory, but within the framework of possibility theory a pair of dual set functions, possibility and necessity, is used whereas probability theory only uses one set function.

	Probability	Possibility	Belief	Fuzzy Sets
Universe	sample space Ω	universe of discourse Ω	frame of discernment Ω	universe of discourse Ω
Distribution	$p : \Omega \rightarrow [0, 1]$	$\pi : \Omega \rightarrow [0, 1]$	-	$\mu : \Omega \rightarrow [0, 1]$
Measures	probability $P : 2^{\Omega} \rightarrow [0, 1]$	possibility $\Pi : 2^{\Omega} \rightarrow [0, 1]$, necessity $N : 2^{\Omega} \rightarrow [0, 1]$	mass $m : 2^{\Omega} \rightarrow [0, 1]$, belief $Bel : 2^{\Omega} \rightarrow [0, 1]$, plausibility $Pl : 2^{\Omega} \rightarrow [0, 1]$	membership $\mu : \Omega \rightarrow [0, 1]$
Constraints	$P(\emptyset) = 0$, $P(\Omega) = 1$, $P(A) = \sum_{\omega_i \in A} p(\omega_i)$	$\Pi(\emptyset) = 0$, $N(\emptyset) = 0$, $\Pi(\Omega) = 1$, $N(\Omega) = 1$, $\Pi(A) = max_{\omega_i \in A}(\pi(\omega_i))$	$\sum_{A \subseteq \Omega} m(A) = 1$ (closed-world assumption: $m(\emptyset) = 0$)	$-$
Monotony	$A \subseteq B \Rightarrow P(A) \leq P(B)$	$A \subseteq B \Rightarrow \Pi(A) \leq \Pi(B)$, $A \subseteq B \Rightarrow N(A) \leq N(B)$	$A \subseteq B \Rightarrow Bel(A) \leq Bel(B)$, $A \subseteq B \Rightarrow Pl(A) \leq Pl(B)$	$A \subseteq B \Rightarrow \mu_A(\omega_i) \leq \mu_B(\omega_i), \forall \omega_i \in \Omega$
Union $A \cup B$	$P(A \cup B) = P(A) + P(B) - P(A \cap B)$	$\Pi(A \cup B) = \max(\Pi(A), \Pi(B))$, $N(A \cup B) \geq \max(N(A), N(B))$	$Bel(A \cup B)$	$\mu_{A \cup B} = \max(\mu_A, \mu_B)$ or $\mu_{A \cup B} = \mu_A + \mu_B - \mu_A \mu_B$
Intersection $A \cap B$	$P(A \cap B) = P(A)P(B)$	$N(A \cap B) = \min(N(A), N(B))$	$Bel(A \cup B) \geq Bel(A) + Bel(B)$, $Pl(A \cup B) \leq Pl(A) + Pl(B)$	$\mu_{a \cap b}(\omega_i) = \min(\mu_a(\omega_i), \mu_b(\omega_i))$

Table 3.1: *Comparison of the different theories for representing uncertainty.*

	Probability	Possibility	Belief	Fuzzy Sets
Ignorance	$p(\omega_i) = \frac{1}{\|\Omega\|}, \forall \omega_i \in \Omega$	$\pi(\omega_i) = 1, \forall \omega_i \in \Omega$	$m(\Omega) = 1, m(A) = 0, \forall A \subset \Omega$	$\mu(\omega_i) = 0, \forall \omega_i \in \Omega$
Conflict	$-$	$c = 1 - h(\pi_1, \pi_2)$	$c = \sum_{A,B:A \cap B = \emptyset} m_1(A) \cdot m_2(B)$	$c = 1 - h(\mu_1, \mu_2)$
Duality	$P(A) + P(\overline{A}) = 1$	$\Pi(A) + N(\overline{A}) = 1$	$Pl(A) + Bel(\overline{A}) = 1$	$\mu_A(\omega_i) + \mu_{\overline{A}}(\omega_i) = 1$
Special Additivity	$P(A) + P(\overline{A}) = 1$	$\Pi(A) + \Pi(\overline{A}) \geq 1$, $N(A) + N(\overline{A}) \leq 1$, $\max(\Pi(A), \Pi(\overline{A})) = 1$, $\min(N(A), N(\overline{A})) = 0$	$m(A) + m(\overline{A}) \leq 1$, $Bel(A) + Bel(\overline{A}) \leq 1$	$\mu_A(\omega_i) + \mu_{\overline{A}}(\omega_i) = 1$
Dominance	$P(A)$	$N(A) \leq P(A) \leq \Pi(A)$	$Bel(A) \leq P(A) \leq Pl(A)$	$-$
Conjunctive Fusion	$p(\omega_i \| x_1, ..., x_k) = \frac{p(\omega_i) \prod_{j=1}^{k} p(x_j \| \omega_i)}{\prod_{j=1}^{k} p(x_j)}$	$\pi_{conj}(\omega_i) = \min_{j=1}^{k} \pi_j(\omega_i), \forall \omega_i \in \Omega$	$m_{1,2,...,k}(C) = \sum_{A_1 \cap ... \cap A_k = C} \prod_{i=1}^{k} m_i(A_i)$	$\mu_{1,2,...,k}(\omega_i) = \min_{j=1,2,...,k} \mu_j(\omega_i)$
Disjunctive Fusion	$-$	$\pi_{disj}(\omega_i) = \max_{j=1}^{k} \pi_j(\omega_i), \forall \omega_i \in \Omega$	$m_{1 \cup 2 \cup ... \cup k}(C) = \sum_{A_1 \cup ... \cup A_k = C} \prod_{i=1}^{k} m_i(A_i)$	$\mu_{1,2,...,k}(\omega_i) = \max_{j=1,2,...,k} \mu_j(\omega_i)$

Table 3.2: *Comparison of the different theories for representing uncertainty.*

In contrast to the other theories for handling uncertainty facilitates the belief theory, in particuar the Transferable Belief Model, an open-world assumption, i.e. the universe of discourse is not assumed to be exhaustive.

Probabilistic product fusion, possibilistic minimum fusion and the orthogonal sum show all a conjunctive behaviour [6] and are all associative and commutative.

Figure 3.1 relates the different uncertainty measures to each other.

The different concepts of the explicated frameworks for dealing with uncertainty are contrasted in the tables 3.1 and 3.2.

3.6 Discussion

For the evaluation of hierarchical classifier Dempster-Shafer evidence theory has been chosen over possibility theory, fuzzy set theory and probability theory as it provides means for representing evidence for sets of events, for dealing with uncertainty and ignorance and for combining separate pieces of information or evidence.

Probability theory does not allow to differentiate between uncertainty and ignorance. Both cases are represented in an identical manner. In contrast the belief theory allows to represent both cases in a different manner. Ignorance is represented by assigning evidence to the frame of discernment Ω. In case of uncertainty evidence can be assigned to the individual hypotheses one is in doubt about or to the set of these hypotheses. Moreover probability theory requires the assignment of a probability value to each element of the sample space Ω, i.e. the knowledge to be represented needs to be complete whereas belief theory allows for the representation of partial knowledge. Furthermore within probability theory a prerequisite for the fusion of information from different sources is the total reliability of the sources. Within the belief theory framework the disjunctive fusion rule can be used to combine not totally reliable sources.

Though comprising methods for combining evidence of different sources, possibility theory does not facilitate the discrimination of doubt and ignorance. Moreover possibility theory proceeds on the closed-world assumption. In contrast belief theory facilitates also an open-world assumption.

The classifier hierarchy inherently provides evidence at different levels of abstraction due to the coarse to fine hierarchical class grouping. In probability theory there is no straightforward way of assigning evidence to a set of events.

Moreover, not all classifiers within the hierarchy have been trained with all classes. Thus when presenting a sample x of a certain class c to all classifiers within the hierarchy some of the classifiers have not been trained with this class and therefore their response should indicate ignorance, i.e. the sample is an outlier or belongs to

an unknown class. But this representation of ignorance should be distinguishable from the representation of uncertainty or doubt. In probability theory both cases are represented in an identical manner and are thus indistinguishable: One cannot distinguish between a lack of belief and disbelief, or in other words: belief cannot be withhold from a proposition without assigning that belief to the negation of this proposition.

The individual classifiers within the hierarchy each form a separate source of information providing evidence about the real class c of a sample x. Hence it is necessary to combine this information in a suitable manner in order to obtain a collective classification result. Belief theory provides means for combining evidence from different sources in a straightforward way.

Table 3.3 summarises the discussed advantages and drawbacks and compares the different theories for handling uncertainty with respect to constraints emerging in the context of hierarchical neural network classifiers. This comparison shows the suitability of the belief theory for being applied to classifier hierarchies as it in contrast to the other theories fulfills all requirements and thus justifies the selection of the belief theory.

Requirement	Probability	Possibility	Belief	Fuzzy Sets
Assign belief only to the level of detail that is justified	−	−	+	−
Open world assumption	−	−	+	−
Different representation for uncertainty and ignorance	−	−	+	−
Conflict	−	+	+	+
Combination rule	+	+	+	+

Table 3.3: *Comparison of the different theories for representing uncertainty with respect to requirements arising when applying the theories in the context of classifier hierarchies. The table indicates for each theory whether the individual requirements are facilitated or not.*

4 Preprocessing Methods

The preprocessing of the input data of a neural network is an important factor for the performance of such a network. Preprocessing may involve a transformation of the input data as well as a reduction of the dimensionality of the input data. The former is performed to ensure that the input data span similar ranges. The latter counteracts the curse of dimensionality [4], which is the fact that in a high-dimensional feature space a vast number of data points is required to cover the space appropriately. Hence previously mapping high-dimensional data into a space of lower dimensionality can yield a potential improvement of the network performance.

4.1 Data Transformation

In order to ensure that the data does not differ significantly with respect to the individual dimensions of the feature space a linear transformation is performed which treats the dimensions as independent. The data is normalised to have zero mean and unit standard deviation. Therefore the mean $\overline{x_i}$ and the variance σ_i^2 with respect to the training data set τ are calculated for each dimension $i = 1, 2, ..., d$ as

$$\overline{x_i} = \frac{1}{M} \sum_{\mu=1}^{M} x_i^\mu \tag{4.1}$$

and

$$\sigma_i^2 = \frac{1}{M-1} \sum_{\mu=1}^{M} (x_i^\mu - \overline{x_i})^2 \tag{4.2}$$

where M is the number of instances in the data set and x_i^μ is the value of the μth feature vector in the ith dimension with $i = 1, 2, ..., d$.

The data is then rescaled using the following transformation

$$\tilde{x_i^\mu} = \frac{x_i^\mu - \overline{x_i}}{\sigma_i}. \tag{4.3}$$

4.2 Reduction of Dimensionality

A reduction of the dimensionality of the input data entails a loss of information. The dimensionality reduction therefore aims at retaining as much relevant information as possible. The goal is to map the d-dimensional input space to a $\tilde{d}$-dimensional space with $\tilde{d} < d$.

The principal component analysis or Karhunen-Loève transformation is a linear transformation that reduces the dimensionality d of the original feature space where the reduced feature space comprehends the maximum possible variance of the original feature space. It represents significant features as linear combinations of the original features.

In order to be able to perform the principle component analysis the data $X = (x_i^\mu)_{\mu=1,2,...,M}^{i=1,2,...,d}$ has to be mean rectified such that for each dimension the mean value over the M samples equals zero. This is achieved by calculating for each dimension $i = 1, 2, ..., d$ the mean $\overline{x_i}$:

$$\overline{x_i} = \frac{1}{M} \sum_{\mu=1}^{M} x_i^\mu. \tag{4.4}$$

Afterwards the means are subtracted from the original data:

$$x_i^{\mu\prime} = x_i^\mu - \overline{x_i} \tag{4.5}$$

with $i = 1, 2, ..., d$ and $\mu = 1, 2, ..., M$.

Therefore the eigenvectors and the eigenvalues of the covariance matrix of the mean rectified data X' (see equation 4.5) are calculated.

The resulting matrix of the mean rectified data $X' = (x_i^{\mu\prime})_{\mu=1,2,...,M}^{i=1,2,...,d}$ is used to calculate the covariance matrix C as follows:

$$C = \sum_{\mu=1}^{M} x^\mu (x^\mu)^T. \tag{4.6}$$

where the individual components $c_{i,j}$ of the covariance matrix are given by

$$c_{i,j} = \frac{1}{M} \sum_{\mu=1}^{M} x_i^\mu x_j^\mu. \tag{4.7}$$

The covariance matrix C is a symmetric $d \times d$ matrix, i.e. $c_{i,j} = c_{j,i}$. The elements $c_{i,i}$ on the diagonal of the covariance matrix denote the variance of the data along the ith dimension.

From the covariance matrix C the corresponding eigenvalues u_i and eigenvalues λ_i are calculated where u_i is the eigenvector belonging to the eigenvalue λ_i. The eigenvectors u_i are called principal components. The eigenvectors u_i belonging to the largest eigenvalues λ_i correlate to the dimensions that show the strongest variance. As the matrix C is symmetric all eigenvalues λ_i are real-valued and the eigenvectors u_i are pairwise orthogonal.

The eigenvalues and the corresponding eigenvectors are then sorted such that $\lambda_1 \geq \lambda_2 \geq ... \geq ...\lambda_i \geq ... \geq \lambda_d \geq 0$. The corresponding eigenvectors are sorted accordingly. To yield the intended reduction in dimensionality the data matrix X' is transformed using the $\tilde{d}$ eigenvectors corresponding to the $\tilde{d}$ strongest eigenvalues by projecting the data vectors $x^{\mu\prime}$ onto the eigenvectors u_i.

4.3 Discussion

Preprocessing may comprise according to requirements miscellaneous operations of different complexity such as linear transformation of the input data, reduction of the dimensionality of the input data, expurgating the input data if the data suffers from outliers, incorrect teacher signals or missing input values, or usage of prior knowledge. Within the context of this work only the former two methods were applied.

The linear rescaling yielding similar values of the input data in all dimensions of the feature space is essential inter alia for nearest-neighbour classifiers as this avoids that one dimension of the feature space prevails the other dimensions. The values of the different dimensions of the feature space might considerably differ in magnitude but this might not reflect the importance of the different features for the classfication. In case of RBF networks the input normalisation allows for the usage of a scalar value width parameter σ as the data shows the same variance in all dimensions.

The reduction of the input dimensionality obviates the sparseness resulting from too few samples in too high dimensional feature spaces which is a problem particularly for RBF networks.

Besides the features used for classification the preprocessing of the data has a considerable influence on the performance of neural network classifiers. Applying a reasonable preprocessing transformation to the input data before it is presented to the neural network is advantageous in most instances and leads to improved classification performances [5].

5 Evaluation Methods

In order to compare different learning algorithms it is necessary to estimate their performance. As the performance of learning algorithms shows a certain variance, results on the basis of a single run of the algorithm are neither reliable nor meaningful. Moreover is it not sufficient to only evaluate the learning algorithm on the training data as this results are too optimistic. Thus the usage of a test data set different from the training data set is necessary for the more realistic assessment of a learning algorithm. A method accounting for this is the cross-validation approach. This approach estimates the classification accuracy of the evaluated learning algorithm. The estimated classification accuracies can than be compared by means of statistical significance tests which evaluate whether the results of two learning algorithms differ considerably.

5.1 Cross-Validation

Cross-validation is a common technique for assessing the capability of learning algorithms when only a limited amount of data is available for evaluation. The idea behind it is not to use the complete data set for training the algorithm but to use only a part of the data set for training and the remaining part of the data set for testing the performance of the algorithm.

To conduct one cross-validation run the data is randomly permutated and divided into k subsets of equal size where it might not always be possible to split the data into subsets of exactly the same size. These k subsets are called folds. The number of folds is naturally limited to $2 \leq k \leq M$ where M is the total number of samples in the data set. Then k experiments are performed in each of which one of the k subsets is respectively used as test set and the remaining $k-1$ subsets are used as training set. If the permutation and splitting of the data is iterated more than once this is referred to as repeated cross-validation. One iteration is called a run. There are r runs. Thus for r-times k-fold cross-validation $r \times k$ experiments are conducted where $a_{i,j}$ is the accuracy of the evaluated algorithm

for fold i in run j with $1 \leq i \leq k$ and $1 \leq j \leq r$. In this experiment all data except the data in fold i of run j is used to train the algorithm. The data in fold i of run j is employed for testing.

The $r \times k$ accuracies $a_{i,j}$ can then be used to calculate a mean accuracy $a = \frac{1}{rk} \sum_{i=1}^{k} \sum_{j=1}^{r} a_{i,j}$. If two algorithms A and B are to be compared each of the $r \times k$ experiments is conducted with both algorithms and the respective accuracies $a_{i,j}$ and $b_{i,j}$ are gained so that exactly the same training and test data is used to obtain both $a_{i,j}$ and $b_{i,j}$. Thus the accuracies are paired and the differences of the accuracies $d_{i,j} = a_{i,j} - b_{i,j}$ can be used as input for paired statistical significance tests.

Although for one cross-validation run there is no overlap of the k different test data sets, the data sets used for training overlap considerably as each two training sets always consist of $k - 2$ identical folds. Considering different runs, there is also overlap as well for the training data as for the test data. This violates the independence assumption most significance tests require.

Depending on the choice for k there are different variants of cross-validation. The *holdout method* is the simplest form of cross-validation. It chooses $k = 2$, which means that the data set is split into two sets, the training and the test set. The most costly form of cross-validation is *leave-one-out cross-validation* with $k = M$, i.e. the number of folds is equal to the number of data points in the data set. With $2 < k < M$ the variant is called k-fold cross-validation.

In order to account for potential differences of the class frequencies in the data set so-called stratified cross-validation is used which considers the relative class frequencies when splitting the data set into folds such that the relative class frequencies in each fold are the same as in the complete data set.

5.2 Testing for Significance

Significance tests are used to statistically detect differences or effects on the basis of observed values. Previously formulated hypotheses are examined where the null hypothesis H_0 assumes no difference or effect and the alternative hypothesis H_1 assumes a difference or an effect.

Significance tests can distinguish an observed result from chance with a low probability of error. This probability of error is defined by the significance level α and confines the error probability to reject the null hypothesis although the null hypothesis is correct.

The quality of a statistical test is typically valuated on the basis of type I and type II errors. A type I error is the erroneous rejection of the null hypothesis, i.e. detecting a difference when actually no difference exists. The probability of

committing a type I error is specified by the significance level α. A type II error corresponds to the erroneous acceptance of the null hypothesis, i.e. indicating that there is no difference when actually a difference exists. The probability of the occurrence of a type II error is denoted by β. There exists an interdependency between the two types of errors. The reduction of the probability of making one error increases the probability of the occurrence of the other error. The size of a statistical test is the probability of a type I error. The power of a statistical test is defined by the probability of correctly rejecting a false null hypothesis. The power is defined as $1 - \beta$.

Significance tests usually provide a test statistic T by means of which the statistical significance is assessed. The distribution of the test statistic specifies the probability that the test statistic takes a certain value depending on the number of observations and the utilised test procedure. The corresponding probability is the so-called p-value. This value is used to decide whether the observed difference is statistically significant or not. The observation is regarded as statistically significant if the p-value is smaller than the perviously defined significance level α. The null hypothesis H_0 can than be rejected in favour of the alternative hypothesis H_1. A small p-value supports the fact that there is statistical evidence for a difference of unspecified strength.

Statistical significance test can be distinguished into parametric and non-parametric tests. Parametric tests, such as the t-test, require the observed values to follow a particular distribution and thus rely on the estimation of parameters specifying this distribution. Non-parametric or distribution-free test, such as the maximum test, the sign test or the signed rank test, make no requirements concerning the distribution the data follows.

When comparing two learn algorithms A and B by means of statistical significance tests a common approach is to conduct a r-times k-fold cross-validation experiment for each algorithm. From the individual classification accuracies a_i and b_i $n = rk$ differences $d_i = a_i - b_i$ with $i = 1, 2, ..., n$ are calculated. These difference form the input for the statistical significance tests.

5.2.1 t-Test

The t-test is a parametric significance test requiring the observed values to follow a normal distribution and to be independent.In the following the pairwise t-test based on r-times repeated k-fold cross validation is described. An r-times k-fold cross-validation comprises r runs and k folds.

Each of the two algorithms A and B to be compared are tested using the same splitting of the data sets based on the r-times k-fold cross validation. The corresponding individual results $a_{i,j}$ and $b_{i,j}$ are pairwise subtracted resulting in $n = rk$ differences $d_{i,j} = a_{i,j} - b_{i,j}$. On the basis of these distances the mean m and the

variance $\hat{\sigma}^2$ could be estimated as:

$$m = \frac{1}{kr} \sum_{i=1}^{k} \sum_{j=1}^{r} d_{i,j} \tag{5.1}$$

and

$$\hat{\sigma}^2 = \frac{1}{kr-1} \sum_{i=1}^{k} \sum_{j=1}^{r} (d_{i,j} - m)^2. \tag{5.2}$$

Under the assumption that the distances $d_{i,j}$ are independent the test statistic t is then given by

$$t = \frac{m}{\sqrt{\frac{1}{kr}\hat{\sigma}^2}} \tag{5.3}$$

and follows a t-distribution with $df = kr - 1$ degrees of freedom.

This test statistic t is then compared against the Student's t-distribution to determine the corresponding p-value.

5.2.2 Corrected Repeated k-Fold Cross Validation t-Test

The corrected repeated k-fold cross validation t-test [7] is a pairwise t-test based on r-times repeated k-fold cross validation with a variance correction to compensate the highly violated independence assumption.

As the independence assumption is violated in case of cross-validation experiments and thus the variance is underestimated, the standard t-test is not applicable to these experiments.

The test statistic $\tilde{t}$ is calculated analogously to the t-value of the t-test utilising a r-times k-fold cross validation. It only differs in an addiional variance correction resulting in the following calculation of the corrected test statistic $\tilde{t}$:

$$\tilde{t} = \frac{m}{\sqrt{(\frac{1}{kr} + \frac{n_2}{n_1})\hat{\sigma}^2}}. \tag{5.4}$$

Here n_1 is the number of samples in the training data set and n_2 is the number of samples in the test data set.

This test statistic $\tilde{t}$ is then used to determine the corresponding p-value according to a t-distribution $df = kr - 1$ degrees of freedom.

A suitable choice for the values r and k are: $r = 10$ and $k = 10$.

5.2.3 Maximum Test

The maximum test [84] is a simple non-parametric test for comparing two paired data series. The differences are sorted according to their absolute value. The number of the absolute highest differences which have the same sign determines the p-value. If two differences have the same absolute value but different signs, they are sorted so that an existing sequence of identical signs is decreased in length.

The null hypothesis, which states that the paired differences are symmetrically distributed around zero, can be rejected at a significance level $\alpha = 2^{-t+1}$ if the t absolute highest differences have the same sign.

The maximum test can be used to independently verify the well established t-test (see section 5.2.1) though being no replacement for the t-test.

5.2.4 Sign Test

The sign test is a non-parametric significance test. This test only considers the signs of the individual differences. The magnitude of the differences are not taken into account.

The null hypothesis for the sign test states that the differences of paired observations on average do not differ from zero or the median of the distribution of the differences equals zero. It is expected that approximately half of the differences are smaller than zero, i.e. have a negative sign, and half of the difference are greater than zero, i.e. have a positive sign. Thus the underlying distribution of both the positive and the negative differences is the binomial distribution with success probability $p = \frac{1}{2}$ and number of trial $n = \tilde{n}$, where $\tilde{n}$ is the number of non-zero differences as only the positive and the negative differences are considered. Zero differences are ignored.

The test statistic T is determined as the maximum of the number of positive differences $T^+ = \sum_{d_i>0} 1$ and the number of negative differences $T^- = \sum_{d_i<0} 1$:

$$T := \max(T^+, T^-). \tag{5.5}$$

The number of considered non-zero differences $\tilde{n} = T^+ + T^-$ can thus be smaller than the number of total differences n.

The p-value is obtained by determining the probability of observing a value of at least T under the binomial distribution with $p = \frac{1}{2}$ and $n = \tilde{n}$ and doubling this probability.

5.2.5 Wilcoxon Matched Pairs Signed Rank Test

A more powerful test than the sign test is the Wilcoxon signed rank test. It is a non-parametric alternative to the paired t-test (see section 5.2.1) to test the difference between paired data. The test checks whether the distribution underlying the differences of matched pairs of observations from two samples is symmetric with zero median $\tilde{\mu}_d = 0$. Thus the null hypothesis H_0 states that the paired differences d_i emanate from a population with the distribution function $F(d)$ where $F(+d) + F(-d) = 1$ or the density $f(d)$ where $f(+d) = f(-d)$. A rejection of the null hypothesis therefore implies that either the population is not symmetrical around the median, i.e. the median of the differences does not equal zero, or the two samples have different distributions.

The test considers the direction of the differences of the value pairs, i.e. the sign as well as the amount of deviation.

No assumptions about the form of the distribution of the data is required. Thus the test can be used if the distributional assumption that underlies the paired t-test is violated.

To perform the test the differences d_i are sorted regardless of their sign in ascending order, i.e. the absolute difference values $|d_i|$ are sorted, and rank numbers $\tilde{R}_i = rank(|d_i|) \in [1, n]$ are assigned to the sorted differences where the rank of a value corresponds to the position of this value in the ordered list of values such that the rank 1 is assigned to the lowest value and the rank n is assigned to the highest value. Zero differences $d_i = 0$ will be disregarded, i.e. only pairs with different values will be considered. If any differences are equal their ranks are averaged. Thus the rank $\tilde{R}_i$ of the distance d_i indicates how many distances are smaller than or equal to the distance d_i. As zero differences $d_i = 0$ are excluded the number of ranked differences $\tilde{n}$ might be smaller than the total number of differences $n \geq \tilde{n}$.

The positive and the negative rank numbers are summed separately yielding

$$T^+ = \sum_{d_i>0} \tilde{R}_i \in [0, \frac{\tilde{n}(\tilde{n}+1)}{2}] \tag{5.6}$$

and

$$T^- = \sum_{d_i<0} \tilde{R}_i \in [0, \frac{\tilde{n}(\tilde{n}+1)}{2}] \tag{5.7}$$

where $T^+ + T^- = \sum_{i=1}^{\tilde{n}} i = \frac{\tilde{n}(\tilde{n}+1)}{2}$.

The smaller rank sum is used as test statistic T:

$$T = \min(t^+, t^-). \tag{5.8}$$

The mean and variance of the test statistic T are $\frac{\tilde{n}(\tilde{n}+1)}{4}$ and $\frac{\tilde{n}(\tilde{n}+1)(2\tilde{n}+1)}{24}$, respectively. For large sample sizes (approximately $\tilde{n} > 25$) the standard normal distribution with mean $\mu_t = \frac{\tilde{n}(\tilde{n}+1)}{4}$ and variance $var_t = \frac{\tilde{n}(\tilde{n}+1)(2\tilde{n}+1)}{24}$ is an approximation of the test statistic T.

The p-value is determined by calculating the value $\tilde{z}$ and comparing this value against the standard normal distribution.

$$\tilde{z} = \frac{|t - \frac{\tilde{n}(\tilde{n}+1)}{4}|}{\sqrt{\frac{\tilde{n}(\tilde{n}+1)(2\tilde{n}+1}{24}}} \tag{5.9}$$

5.2.6 Quantile-Quantile Plot

A technique for graphically ascertaining whether two data sets origin from populations with identical distributions is the quantile-quantile plot. Therefore the quantiles of the first data set are plotted against the quantiles of the second data set with a quantile being a measure which realises a subdivision of a distribution into equidistant percentage points such that the resulting partitions comprise equal proportions of the distribution.

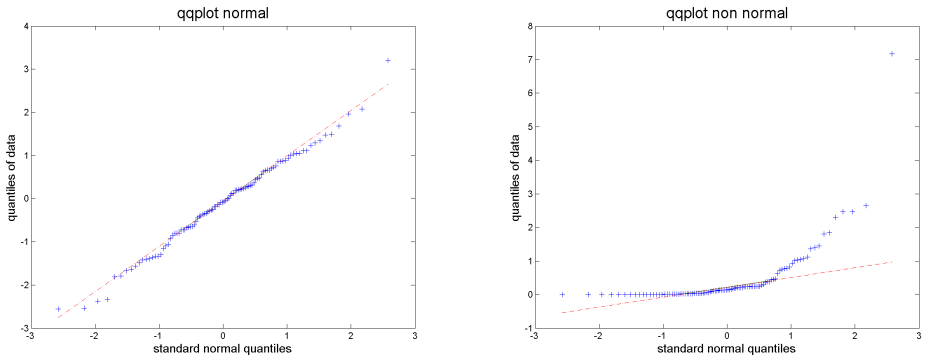

Figure 5.1: *Examples of quantile-quantile plots. The left plot charts data originating from a normal distribution whereas the right plot depicts data originating from a non normal distribution.*

The value of a specific quantile, the p-quantile, specifies the value that separates the lower $p \cdot 100$ percent of the data points and the upper $(1-p) \cdot 100$ percent of the data points, i.e. it gives the point of the distribution below which fall $p \cdot 100$ percent of the data. If the two data sets follow the same underlying distribution the qq-plot is approximately linear, i.e. the plotted points approximately yield a straight line.

Quantile-quantile plots can be used to verify whether the data fulfills the normality requirement.

5.3 Discussion

Within the scope of this thesis the four significance test described above are used to statistically evaluate the proposed approach.

The sign test is a simple and rather insensitive significance test. No assumptions are required for this test, which makes it easy to apply.

The signed rank test is similar to the sign test but shows a much greater sensitivity. For large number of samples this test shows nearly the same sensitivity as the t-test. For unknown distributions and small number of samples this test shows a higher sensitivity than the t-test. Although not assuming normally distributed data, this test requires a symmetric distribution. Thus it is a more powerful alternative to the sign test, but it is more stringent.

The t-test is the most sensitive significance test of the tests used within the scope of this thesis, but it is also the test with the most stringent requirements as it assumes normally distributed data.

The maximum test is simple significance test which can be used to independently validate the t-test.

When applying significance tests to classification results yielding from cross-validation experiments the independence assumption is violated and the normality requirements cannot be guaranteed. The corrected t-test makes allowance for the former. The latter can be verified using quantile-quantile plots.

If the requirements for a significance test are not fulfilled completely, which in case of classification results from cross-validation experiments cannot be guaranteed, the outcome of the significance test can only be considered as indication.

Part II: Developed Methods

This part describes the developed approach of hierarchical neural network classifiers and the different aspects examined within the scope of this thesis.

6 Hierarchical Neural Networks

This chapter deals with the different aspects of hierarchical neural network classifiers which form the basic of this thesis. Initially the basic concept of hierarchical neural networks is described. Afterwards the different stages that are relevant for hierarchical neural network classifiers are explained. After delineating the developed strategies for building and training classifier hierarchies, the information fusion aspect is dealt with in terms of the different methods deployed in the classification phase. Another aspect of classifier hierarchies also treated is the estimation of the classification quality with the main focus of detecting outlier. Finally the features and benefits as well as the disadvantages of hierarchical neural network classifiers in general and the presented aspects in particular are discussed.

6.1 Basics of Hierarchical Neural Networks

Hierarchical neural networks consist of several simple neural networks that are combined in a tree or more general in a rooted directed acyclic graph, i.e. the nodes within the hierarchy represent individual neural networks. In the context of this thesis hierarchical neural network classifiers were investigated were the nodes within the hierarchy are clasifiers. Different neural classifiers were used such as RBF networks, LVQ networks or k-NN classifiers.

The basic idea of hierarchical neural networks is the hierarchical decomposition of a complex classification problem into several less complex ones. This yields hierarchical class grouping whereby the decision process is split into multiple steps exploiting rough to detailed classification. The hierarchy emerges from recursive partitioning of the original set of classes C into several disjoint subsets C_i until subsets consisting of single classes result. C_i is the subset of classes to be classified by node i, where i is a recursively composed index reflecting the path from the root node to node i. The subset C_i of node i is decomposed into s_i disjoint subsets $C_{i,j}$, where $C_{i,j} \subset C_i$, $C_i = \cup_{j=0}^{s_i-1} C_{i,j}$ and $C_{i,j} \cap C_{i,k} = \emptyset$ for $j \neq k$. The total

set of classes $C_0 = C$ is assigned to the root node. Consequently nodes at higher levels of the hierarchy classify between larger subsets of classes whereas nodes at the lowest level discriminate between single classes.

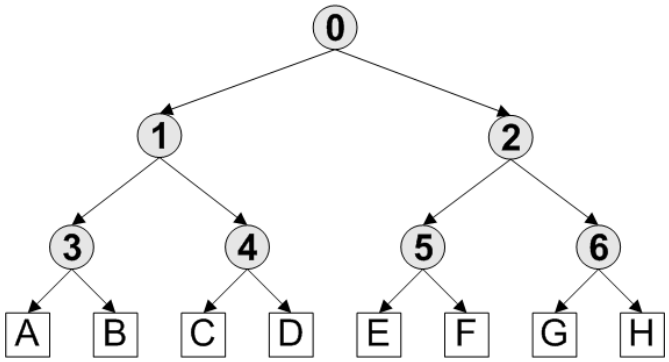

Figure 6.1: *Example of a hierarchical neural network classifier for the classification of eight classes $A, B, ..., H$. Each of the seven nodes $0, 1, 2, ..., 6$ within the hierarchy represents an individual classifier.*

This divide-and-conquer strategy yields several simple classifiers that are more easily manageable, instead of one extensive classifier. These simple classifiers can be amended much more easily to the decomposed simple classification tasks than one classifier could be adapted to the original complex classification task. In case of a binary tree architecture the hierarchy consists of $l - 1$ internal nodes which represent a neural classifier and l leaf nodes which represent the classes to be classified, where l is the number of classes to be classified. If the binary tree is balanced the hierarchy depth is $\log_2 l$. In case of an unbalanced tree the maximal depth is $l-1$. Furthermore different feature types $\mathcal{X}_i$ are used within the hierarchy, i.e. the classifiers work on different feature spaces. For each classification task the feature type that allows for the best discrimination is chosen. An example of such a hierarchy is shown in figure 6.1.

The proposed approach of hierarchical neural network classifiers comprises three independent phases:

1. Hierarchy generation
2. Hierarchy training
3. Classification

In the generation phase the structure of the hierarchy, which is defined by the hierarchical class grouping, is determined and the suitable feature types are selected. In the following training phase the types of classifiers used within the hierarchy are selected and the classifiers are trained. In the last phase the combined classification result is retrieved by applying a certain fusion strategy.

This separation of the individual stages allows for a sizeable flexibility. The same hierarchy generated during the generation phase can e.g. be trained with different classifier types. Moreover in the classification phase several retrieval strategies can be applied to the same hierarchy. In the following the three phases are explained in more detail.

6.2 Hierarchy Generation

The hierarchy is generated by unsupervised k-means clustering [83]. In order to decompose the set of classes C_i assigned to one node i into s_i disjoint subsets a k-means clustering is performed with all data points $X_i = \{x^\mu | t^\mu \in C_i\}$ belonging to the classes under consideration. Depending on the distribution of the classes across the k-means clusters s_i disjoint subsets $C_{i,j}$ are formed. One successor node j corresponds to each subset. For each successor node j again a k-means clustering is performed to further decompose the corresponding subset $C_{i,j}$. The k-means clustering is performed for each feature type. The different clusterings are evaluated and the clusterings which group data according to their class labels are preferred. Since the k-means algorithm depends on the initialisation of the clusters, k-means clustering is performed several times per feature type. Out of these clusterings the best clustering is chosen using a valuation function (see equation 6.1). This clustering then determines not only the partitioning of the classes but also the used feature type.

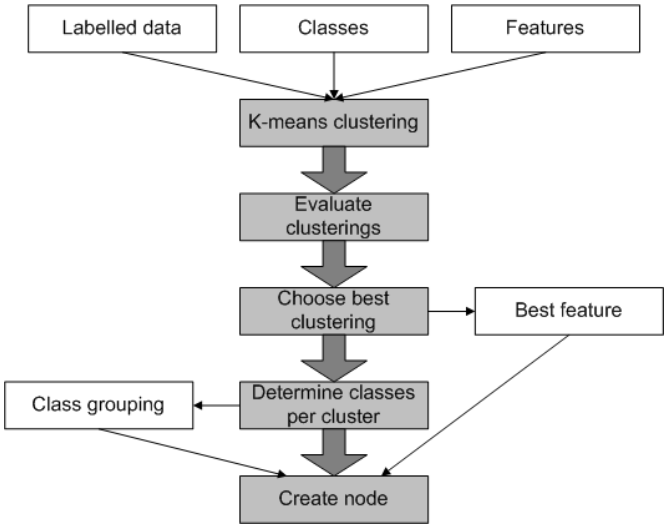

Figure 6.2: *Hierarchy generation.*

The number of clusters k must be at least the number of successor nodes or the number of subsets s respectively but can also exceed this number. If the

number of clusters is higher than the number of successor nodes, several clusters are grouped together so that the number of groups equals the number of successor nodes. All possible groupings determined and evaluated.

In the following all equations only refer to clusterings for reasons of simplicity, i.e. the number of clusters k equals the number of successor nodes s. A valuation function is used to rate the clusterings or groupings respectively. The valuation function prefers unambiguous clusterings that group data according to their class labels. A clustering is regarded as unambiguous if the data of one class are for the most part assigned to one cluster. Ideally all samples of one class are assigned to the same cluster. Clusterings where data is uniformly distributed across clusters notwithstanding their class labels receive low ratings, i.e. it is rewarded if the data of one class is basically assigned to one cluster and it is penalised if the data of one class is spread over several clusters. Furthermore clusterings are preferred which result in balanced hierarchies. This is the case if the sets of classes are divided into subsets of approximately the same size. Thus clustering that evenly divide the classes with respect to the number of data points favoured. Thus the valuation function rewards unambiguity regarding the class affiliation of the data assigned to a prototype as well as uniform distribution regarding the number of data points assigned to each prototype.

The valuation function $V(p)$ consists of two terms regulated by a scaling parameter $\lambda \in [0, 1]$ where p is a stochastic matrix giving the distribution of the data points across the different clusters and classes. The first term $E(p)$ calculates the entropy [69] of the distribution of each class across the different clusters and is a measure for the pureness of the clusters. This accounts for unambiguous distribution of the data considering the corresponding classes, i.e. it attempts to group data of the same class into the same cluster. The term $E(p)$ becomes minimal if it is ensured for all classes that all data belonging to one class is assigned to the same cluster. It becomes maximal if all data belonging to one class is uniformly distributed across all clusters. In this term also other homogeneity measures such as the Gini index [69] could be used instead of the entropy replacing the term $E(p)$ with $G(p)$ showing the same properties as the term $E(p)$. The second term $D(p)$ computes the deviation from the uniform distribution. It rewards clusterings where the same number of classes is assigned to each cluster. This term becomes minimal if the clusters have equal cardinality. This supports the even division of the classes into subsets. During the hierarchy generation phase clusterings that minimise the valuation function $V(p)$ are looked for. The influence of the respective term is regulated by the scaling parameter λ. Both terms are normalised so that they return values in the interval $[0, 1]$. This adjustment of the scaling parameter is e.g. reasonable if there is a considerable difference in the number of similar objects best being represented by a non-balanced hierarchy.

The valuation function $V(p)$ is given by

$$V\,(p) = (1-\lambda)\frac{1}{l\log_2(k)}E(p) + \lambda\frac{1}{l(k-1)}D(p) \to \min\,. \tag{6.1}$$

The following concepts are necessary for the definition of the terms $E(p)$ and $D(p)$.

The Voronoi cell [18] defined by cluster j is denoted by:

$$\mathscr{R}_j = \{x \in \mathcal{X} | j = \text{argmin}_{i=1,2,...,k}\|x - c_i\|\} \tag{6.2}$$

where c_i is the center of cluster i.

The set of data points that were assigned to cluster j is given by

$$\mathscr{C}_j = \mathscr{R}_j \cap X. \tag{6.3}$$

The set of data points that belong to class i is defined by:

$$X^i = \{x_\mu | \mu = 1, 2, ..., M; t^\mu = i\} \subseteq X. \tag{6.4}$$

The rate of patterns from class i, that belong to cluster j is given by:

$$p_i^j = \frac{|X^i \cap Z_j|}{|X_i|}. \tag{6.5}$$

The two elementary terms the valuation function is composed of are then calculated as:

$$E(p) = -\sum_{i=1}^{l}\sum_{j=1}^{k} p_i^j \log_2(p_i^j) \tag{6.6}$$

and

$$D(p) = \sum_{j=1}^{k} |\sum_{i=1}^{l} p_i^j - \frac{l}{k}|. \tag{6.7}$$

If the Gini index is used as homogeneity measure the valuation function $V(p)$ becomes:

$$V\,(p) = (1-\lambda)\frac{1}{l(1-\frac{1}{k})}G(p) + \lambda\frac{1}{l(k-1)}D(p) \to \min\,. \tag{6.8}$$

The term assessing the pureness of the clusters is then calculated as:

$$G(p) = \sum_{i=1}^{l}(1 - \sum_{j=1}^{k}(p_i^j)^2). \tag{6.9}$$

The valuation function $V(p)$ uses more or less sub-symbolic criteria for assessing the different clusterings. Other criteria than the pureness of the clusters, the decidedness of the class distribution and the uniformness of the distribution of the different classes could be used which could as well be on a sub-symbolic level, such as measures for the separateness of the clusters, e.g expressed by means of the inter-class variance and the intra-class variance and rewarding clusters that lie clearly apart in the feature space, or on a symbolic level, such as task specific criteria, taking similarity other than in the feature space into account. The classes could, e.g. be grouped in a manner suitable for the task at hand according to functional similarity of the objects to be classified or be arranged in an optimal way for the code generation such that more frequent classes have shorter codes.

The best clustering, i.e. the one that minimises the valuation function $V(p)$, is chosen and used for determining the division of the set of classes into subsets. Moreover this also determines which feature type $\mathcal{X}_i$ will be used for the corresponding classifier. So each classifier within the hierarchy can potentially use its own feature type. To identify which classes will be added to which subset the distribution of the data across the clusters is considered. The division in subsets C_j is carried out by maximum detection. The set of classes belonging to subset C_j is defined as:

$$C_j = \{i \in C | j = \text{argmax}\{q_{i,1}, ..., q_{i,k}\}\} \tag{6.10}$$

where $q_{i,j} = \frac{|X_i \cap Z_j|}{|Z_j|}$ denotes the rate of class i in cluster j. For each class it is determined to which cluster j^* the majority of data points belonging to this class were associated. The class label will then be added to the corresponding subset C_{j^*}.

The decomposition into disjoints subsets of classes does not necessarily have to be performed utilising unsupervised clustering. Instead of determining clusters, an alternative approach being rather simple and straightforward is considering the class centroids $\overline{x_\omega}$:

$$\overline{x_\omega} = \frac{1}{l_\omega} \sum_{\{x^\mu \in X_i : t^\mu = \omega\}} x^\mu \tag{6.11}$$

where l_ω is the number of data points belonging to class $\omega \in \Omega$.

The two centroids having the greatest distance are identified. The distance to these two centroids $\overline{x_\omega}^*$ and $\overline{x_\omega}^{**}$ is used to split the classes into two disjoint subsets by determining for each centroid $\overline{x_\omega}$ the closest of the two centroids $\overline{x_\omega}^*$ and $\overline{x_\omega}^{**}$.

This simple method does not consider possibly distributed localisations of samples of one class.

To generate the hierarchy at first the set of all classes is assigned to the root node. Starting with a clustering on the complete data set the set of classes is divided into subsets. Each subset is assigned to a successor node of the root node. Now the decomposition of the subsets is continued until no further decomposition is possible or until the decomposition does not lead to a new division.

6.3 Hierarchy Training

The hierarchy is trained by separately training the individual classifiers with the data $\{x^\mu \in X_i | t^\mu \in C_i\}$ that belong to the subsets of classes assigned to each classifier. For the training the respective feature type X_i identified during the hierarchy generation phase is used. In order to be able to train the classifiers with the s_i classes corresponding to the class grouping the data will be relabelled such that all data points of the classes belonging to one subset $C_{i,j}$ have the same label j, i.e. $\tilde{t}^\mu = j, x^\mu \in X_i, t^\mu \in C_{i,j}$. The number of input neurons of the single classifiers is defined by the dimension d_i of the respective feature type X_i assigned to the corresponding node i. The number of output nodes equals the number of successor nodes s_i. The classifiers are trained using supervised learning algorithms. The classifiers within the hierarchy can be trained independently, i.e. all classifiers can be trained in parallel.

Within the hierarchy different types of classifiers can be used. Examples of classifiers would be Radial Basis Function networks, learning vector quantisation networks, k-nearest neighbour classifiers, multi-layer perceptrons networks or support vector machines.

6.4 Classification within the Hierarchy

After the classifiers within the hierarchy have been trained they can be used to classify samples of which the corresponding class is not known. Therefore the sample to be classified is presented to the different classifiers within the hierarchy that individually provide provide classification results on different levels of abstraction. There are various ways to fuse these classification results of the individual classifiers within the hierarchy to one collective result. These strategies

differ inter alia in complexity, performance and output type. A different number of classifiers is involved in the fusion process for the different fusion strategies.

The different evaluation strategies all work on the same hierarchy. In the following the different retrieval strategies are developed in the context of this work are described.

6.4.1 Evaluate Hierarchy Analogous to Decision Tree

A simple and fast way to obtain the classification result is to evaluate the hierarchy similar to the retrieval process in decision trees [18] where a path from the root node of the hierarchy to the leaf node that specifies the resulting class is determined following the strongest evidence at each node. Starting with the root node the classification results of the individual classifiers are used to decide which classifier at the next lower level will be looked at next, i.e. to which successor node the decision will be delegated. Therefore the respective feature vector of the object to be classified is presented to the trained classifier. By means of the classification output the next classifier to categorise the data point is determined, i.e. the classifier j^* corresponding to the highest output value $o(j^*)$ is chosen such that $j^* = \text{argmax}_{j=1..s_i}\{o(j)\}$. Classifier i that discriminates between s_i disjoint subsets $C_{i,j}$ decides to which of these subsets C_{i,j^*} the presented sample most likely belongs. As a result the j^*th successor node is the next classifier looked at. This is successively repeated until a leaf node is reached. Thus a path through the hierarchy from the root node to an end node is obtained which not only represents the class of the object but also the subsets of classes to which the object most likely belongs. Hence the data point is not presented to all classifiers within the hierarchy and the hierarchical decomposition of the classification problem yields additional intermediate information.

This evaluation method only considers a subset of the classifiers within the hierarchy. Figure 6.3 visualises this decision process and shows which classifiers are involved.

The result of this decision process is only the resulting class as well the information to which supersets of classes the sample to be classified belongs. No estimation of the degree of memberships to the other classes of the presented sample is given.

This method features a simple way of combining the results of multiple classifiers. It yields good classification results in rather short classification time, but a major disadvantage is the missing ability to correct misclassifications that occur at higher levels of the hierarchy. The consideration of only a part of the classifiers within the hierarchy has on the one hand a positive effect on the computation time but on the other hand this might result in disregarding potentially relevant information. Hence it would be beneficial not only to take a single path within the hierarchy into account but to consider all classifiers of the hierarchy. However,

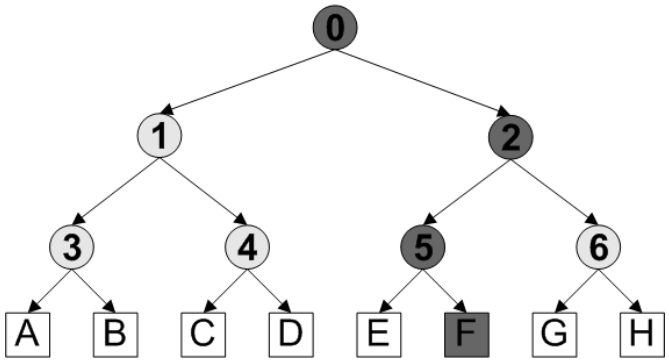

Figure 6.3: *Retrieval of the classification result analogous to decision trees. A path through the hierarchy is determined leading to the resulting class. The highlighted path (in dark grey) shows the nodes activated during the classification of a sample that is classified as class F.*

in contrast to a simple classifier the the classifier hierarchy features the usage of different feature types as well as the availability of class estimations at different levels of abstraction.

6.4.2 Evaluate End Nodes

Another simple way of evaluating the hierarchy is to take only the outputs of the classifiers that are end nodes into account. As the hierarchy is generated by splitting the set of classes into disjoint subsets, each class is only represented by one leaf node. Thus when considering the end node classifiers for each class $\omega \in \Omega$ there is one corresponding classifier output $\tilde{z} \in \mathbb{R}^l$. The collective classification result is the class $j* = \text{argmax}_{j=1\ldots l}\{\tilde{z}_j\}$ corresponding to the highest classifier output.

Figure 6.4 depicts the retrieval of the classification result considering only the end nodes. The nodes that are evaluated, i.e. the end nodes, are highlighted.

The result of this fusion strategy is not only the class that the presented sample most likely belongs to, but also an estimation of how likely the presented sample belongs to the remaining classes as all end nodes are considered and thus classifier outputs for all classes are available.

The fact that only part of the classifiers need to be considered which even can be evaluated in parallel allows for a fast retrieval of the classification result. A disadvantage of this method is the fact that the end nodes during retrieval are presented samples of classes they have nt been trained with as the sample to be classified is presented to all end nodes but each end node has only been trained with a small subset of classes. Thus in general only one end node has been trained with samples of the class of the presented sample, whereas all other end

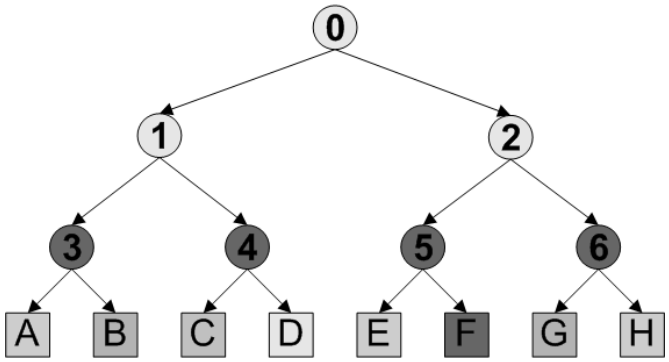

Figure 6.4: *Retrieval of the classification result evaluating the end nodes of the hierarchy. Only the end nodes are considered when calculating the classification result.*

nodes are presented a sample of an unknown class. Unfortunately it cannot be guaranteed that the classifiers always show low responses when being presented a sample of an unknown class. These potentially erroneous classifier responses can superimpose the result of the actually responsible classifier.

6.4.3 Evaluate Hierarchy Utilising a Voting Scheme

Another approach for combining the results of the individual classifiers within the hierarchy is a majority voting principle. Each classifier within the hierarchy votes for a class or a set of classes. The voting for sets of classes is equally divided among the elements of the respective set. The classification result is the class that gains the most votes.

This voting scheme is similar to the voting principle used to combine the classification results of pairwise coupled binary classifiers [32].

Figure 6.5 shows the voting fusion scheme. The classifiers involved are highlighted.

The result of the fusion is the winning class as well as an estimation of the probabilities of the presented sample to belong to the individual classes as the classifiers overall can vote for all classes.

For this approach all classifiers have to be evaluated which however can be performed in parallel. Another severity is the fact that during retrieval many classifiers are presented samples of classes they have not been trained with. These classifiers also vote for a class or set of classes. In case the votes of these classifiers do not diverge sufficiently their votes might possibly outvote the actually responsible classifiers.

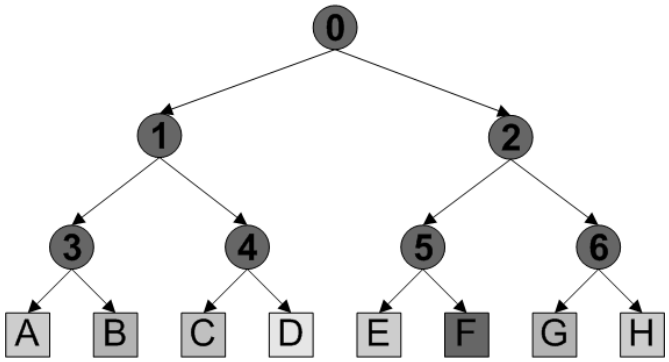

Figure 6.5: *Retrieval of the classification result utilising a simple voting scheme. All classifiers are considered when calculating the classification result.*

6.4.4 Evaluate Hierarchy Utilising Dempster-Shafer Evidence Theory

A more complex way of combining the results that involves all classifiers of the hierarchy is an approach utilising the belief theory. The sample to be classified is presented to all classifiers within the hierarchy and the individual results are then combined to one collective result. The strengths of the individual results are incorporated by this method. The combination is performed utilising Dempster-Shafer theory of evidence.

Figure 6.6 depicts which classifiers are considered for this decision process.

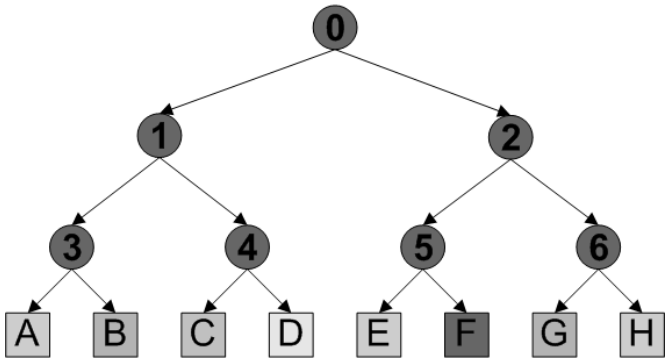

Figure 6.6: *Retrieval of the classification result utilising Dempster-Shafer evidence theory. All classifiers are considered when calculating the classification result.*

In order to apply Dempster-Shafer theory for the evaluation of the classifier hierarchy it is at first necessary to derive basic probability assignments m_i from the outputs of the individual classifiers within the hierarchy expressing the likelihood that the presented sample belongs to the corresponding subsets of classes. Not all neural classifiers produce output that satisfies the conditions for probability assignments (see equation 3.41). In these cases a transformation of the outputs

is necessary. The output of fuzzy k-nearest neighbour classifiers $\Xi_j(x)$ fulfils the conditions for basic probability assignments as the class memberships satisfy the conditions $\Xi_j(x) \in [0,1]$ and $\sum_{j=1}^{s_i} \Xi_j(x) = 1$ where s_i is the number of classes the classifier discriminates between whereas the output of radial basis function networks $z_j(x)$ does not necessarily do so. To enforce the fulfillment of the condition $z_j(x) \in [0,1]$ a ramp function

$$\Theta(z_j(x)) = \begin{cases} 0, & x < 0 \\ x, & 0 \leq x \leq 1 \\ 1, & x > 1 \end{cases} \tag{6.12}$$

is is applied to the classifier output setting all negative values to zero and all values greater than 1 to 1. This is justified insofar as typically only a negligible number of output values violate this condition. In order to account for ignorance which is represented by low classifier outputs the difference to one is assigned to Ω. If the sum of the classifier outputs is equal to or greater than one nothing is assigned to Ω. In the latter case the output is then normalised to sum up to one. Hence in either case the total evidence assigned by one classifier sums up to one.

These transformations are applied if necessary to the outputs of all classifiers and then the resulting basic probability assignments m_j of all classifiers are combined using the orthogonal sum without normalisation (equation 3.42).

According to the hierarchy structure each classifier provides evidence for the specific subsets of Ω between which the classifier discriminates and for Ω itself. In case of ignorance strong evidence is assigned to Ω.

Furthermore, a discounting technique is used propagating the classifier responses at higher levels of the hierarchy down: Classifier responses along pathes that at a higher level contain a classifier that assigned low responses are weakened strongly whereas pathes below classifiers with strong output are hardly weakened. The discounting is realised by successively multiplying the classifier responses with the classifier output of the respective predecessor node. Hence the root node is not discounted. The discounting accounts for the fact that within the hierarchy there are a not negligible number of classifier that have to provide results for samples belonging to classes they have not been trained with. Hence low classifier responses, as would be desired, cannot be guaranteed in that cases. The discounting thus weakens insular strong responses, which are likely to be caused by a classifier that has been presented a sample of an unknown class, whereas if only one classifier within a specific path shows a low response but all other classifiers responses are high this leads only to a moderate attenuation. The discounting is applied directly after the transformation of the classifier outputs to basic probability assignments. As a multiplication with the discounting factors $d_i \in [0,1]$ decreases the basic probability assignments if $d_i < 1$, their sum $\sum_{j=0}^{s_i-1} d_i m_i(C_{i,j})$ is then smaller than one. The difference to one originating from

this is then assigned to Ω: $m_i(\Omega) = 1 - \sum_{j=0}^{s_i-1} d_i m_i(C_{i,j})$.

This fusion strategy yield not only the class to which the presented sample most likely belongs but also an estimate of the class memberships of this sample.

6.4.5 Evaluate Hierarchy Utilising Similarity Preserving Codes

Another way of retrieving the combined classification results also incorporating all classifiers within the hierarchy is a method utilising similarity preserving codes generated from the classifier hierarchy (see chapter 8). This strategy involves all classifiers as the codes used are generated from the activation of all classifiers within the hierarchy. This is illustrated in figure 6.7.

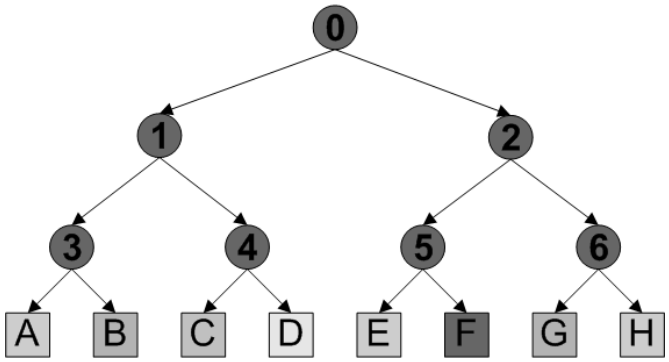

Figure 6.7: *Retrieval of the classification result utilising similarity preserving codes. All classifiers are considered when calculating the classification result.*

For each sample of the training data set τ a binary code is composed by presenting the individual samples x^μ to the trained hierarchy and encoding the activation of the neural networks within the hierarchy. The so obtained codes and the corresponding class affiliation are then either stored in a hetero-associative memory or used to train a simple nearest neighbour classifier. The classification result is then attained by generating the codes from the samples using the trained hierarchy and afterwards classifying the code by means of the trained associative memory or nearest neighbour classifier respectively.

The classification result is then obtained by presenting the sample to be classified to the trained hierarchy and retrieving the activation vector. The memory matrix A is then addressed with the so calculated activation vector. The positions of the ones in the retrieved output vector corresponds to the associated classes. If the output vector contains more than one element that equals one, the appropriate class needs to be identified by a suitable selection procedure.

The result of this complex fusion strategy is merely the resulting class. No information about the class membership of the sample to be classified is provided.

This approach is rather complex as it involves the evaluation of all classifiers, the generation of the code vector as well as the classification of the code vector but it benefits from the incorporation of the information provided by all classifiers.

6.4.6 Inter-State Decision Templates

A slightly different approach closely related to decision templates directly utilises the activations of the neural networks instead of binarised codes. This technique is very similar to the fusion method incorporating the similarity preserving codes. It also involves all classifiers within the hierarchy. This is sketched in figure 6.8.

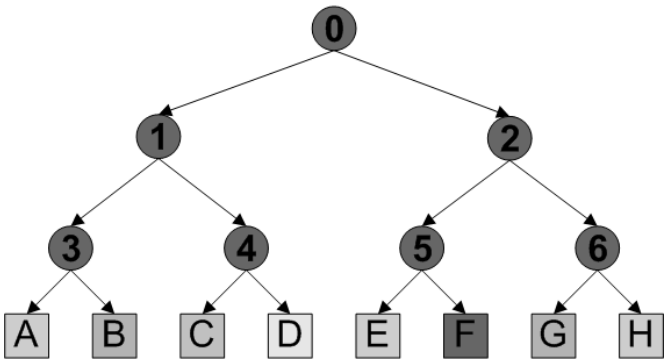

Figure 6.8: *Retrieval of the classification result utilising inter-state decision templates. All classifiers are considered when calculating the classification result.*

The activation vectors are obtained from the trained hierarchy by presenting the samples x^μ of the training data set τ and by concatenating the activations r_i^μ of the individual classifiers within the hierarchy. These M activation vectors r^μ form the activation matrix R. A memory matrix A that can be used analogously to an associative memory is then calculated using the pseudo inverse:

$$A = R^+ T \tag{6.13}$$

where T is the matrix containing the classes t^μ of the samples x^μ as one-out-of-l code.

The classification result is retrieved analogously to the retrieval of the fusion strategy that utilises similarity preserving codes.

This complex fusion strategy exploits the entire information available within the classifier hierarchy at a rather detailed level. The usage of this information makes the approach rather powerful but also time-consuming as not only all classifiers are involved in the decision process, but also the activation vector has to be composed and a downstream classification of the activation vector needs to be performed.

6.5 Outlier Detection

Outliers, i.e. samples of classes the classifier has not been trained with, can massively deteriorate the classification performance of a classifier. Therefore it is beneficial to identify such samples and to reject them.

There are different approaches to determine the classification quality for the individual evaluation strategies which take the strength of the classifier responses or the combined evidences respectively into consideration.

In the case of RBF networks low classifier outputs, i.e. all outputs are below a certain threshold θ_{lower} indicate that it is not likely that the sample to be classified belongs to the corresponding class, whereas high classifier outputs support the option that the sample presumably belongs to the corresponding class. If there are several outputs of approximately similar strength that differ at most by an amount of θ_{dist} this gives rise to the supposition that the classifier is in doubt, i.e. no clear decision between the corresponding classes is possible. The decision is ambiguous and is considered as uncertain. A singular high response being higher than θ_{upper} is an indication for a confident decision. If all classifier outputs are weak it is likely that the presented sample belongs to neither of the classes between which the classifier discriminates. The sample is considered to be an outlier and to belong to an unknown class.

This suggests the usage of three thresholds θ_{lower}, θ_{upper} and θ_{dist}, with $\theta_{lower} \leq \theta_{upper}$. The threshold θ_{lower} is used to identify outliers. If all outputs $z_i(x^\mu)$ of a classifier with $i = 1, 2, ..., l$ where l is the number of classes the classifier discriminates between are lower than θ_{lower} the sample x^μ is regarded to belong to an unknown class. If the highest classifier output $z_{i^*}(x^\mu)$ exceeds the threshold θ_{upper} and the distance $d_{i^*,i^{**}}(x^\mu) = z_{i^*}(x^\mu) - z_{i^{**}}(x^\mu)$ to the second highest output $z_{i^{**}}(x^\mu)$ is greater than θ_{dist} the decision is reckoned an unambiguous decision. Otherwise the decision is regarded as uncertain.

These thresholds can be determined by evaluating the classifier outputs on the training data set τ. The strength of the winner outputs $z_{i^*}(x^\mu)$ for the correctly classified samples are looked at and the threshold θ_{upper} is set to the lowest of these values: $\theta_{upper} := \min_{\mu=1,2,...,M} z_{i^*}(x^\mu)$. In order to account for outliers the threshold θ_{upper} can also be determined as p-quantile, where p has to be chosen appropriately. The threshold θ_{dist} is determined analogously: The distances $d_{i^*,i^{**}}$ between the highest and the second highest output for the correctly classified samples are evaluated and the threshold θ_{dist} is set to the minimal value $\theta_{dist} = \min_{\mu=1,2,...,M} d_{i^*,i^{**}}(x^\mu)$. Here the usage of the p-quantile might be advisable as well. The threshold θ_{lower} is calculated by examining the strength of the winner outputs $z_{i^*}(x^\mu)$ for the incorrectly classified samples if there are any. Otherwise the threshold θ_{lower} is set to an arbitrary value between zero and θ_{upper}, e.g. $\theta_{lower} = \frac{\theta_{upper}}{2}$.

When using the decision-tree like strategy to retrieve the combined classification result in a classifier hierarchy the individual classification qualities of the involved classifiers are taken into account. If the decisions of all considered classifiers are unambiguous the overall decision is also considered as unambiguous. If all involved classifiers indicate that the sample belongs to an unknown class, the presented sample is regarded as an outlier. Otherwise the decision is considered as uncertain.

The belief theory framework as utilised within the scope of this thesis offers a straightforward way for estimating the classification quality. The evidence assigned to the impossible event $\emptyset$ and to the frame of discernment Ω give indications for the quality of the classification result. The evidence $m(\emptyset)$ assigned to the empty set $\emptyset$ is a measure for the conflict between the different sources of information. If the information to be combined is inconsistent a large part of the evidence will be assigned to the empty set $\emptyset$. The evidence $m(\Omega)$ dedicated to the frame of discernment Ω represents ignorance as this is the information that cannot further be subdivided onto the subsets of Ω.

6.6 Discussion

6.6.1 Features and Benefits of Hierarchical Networks

The hierarchical approach makes intermediate classification results available. If only intermediate results are of interest it is not necessary to evaluate the complete path. In order to solve a task it might be sufficient to know whether the object to be recognised belongs to a set of classes and the knowledge of the specific category of the object might not add any value. If the task for example is to grasp a cup, it is not necessary to distinguish between red and green cups. Moreover, when looking for a specific object it might in some cases not be necessary to retrieve the final classification result if a decision at a higher level of the hierarchy already excludes this object.

The hierarchy also facilitates a link between symbolic information and sub-symbolic information processing. The classification itself is performed using feature vectors which represent sub-symbolic information, whereas symbolic knowledge can be provided concomitantly via the information about the affiliation to certain subsets of classes. The usage of neural networks allows the representation of uncertainty of the membership to these classes since the original output of the neurons is not discrete but continuous.

Moreover similarity preserving sparse binary codes can easily be generated from the neural hierarchy. Since the hierarchy is generated using features which are based on the appearance of the objects such as orientation or colour information

it primarily reflects visual similarity. Thus it allows the generation of a sparse similarity preserving representation of the objects. A straightforward approach is the usage of binary vectors of length corresponding to the total number of neurons in the output layer of all networks in the hierarchy. The code is created identifying the strongest activated output neurons for each node. The corresponding elements of the code vector are then set to 1, the remaining elements are set to 0. These properties are extremely useful in the field of neuro-symbolic integration [10] [63] [53]. For the limited task of object localisation and classification a similarity preserving code may not be relevant, but when integrating the object classification system into an overall cortical model realised by associative memories (like e.g. in [41] [23]) this is an important aspect.

6.6.2 Comparison of Hierarchy Evaluation Methods

The evaluation of the classifier hierarchy by means of Dempster-Shafer evidence theory as well as the fusion strategy utilising similarity preserving codes and the inter-state decision template approach yields improved classification results compared to other evaluation methods. When compared directly the evidence-theoretic approach outperformed the decision tree like approach in all tested cases.

The evidence-theoretic approach and the inter-state decision template approach do not require further adjustment of parameters which makes them easily applicable, whereas in order to achieve good classification results when applying the fusion strategy utilising similarity preserving codes it is necessary to adapt certain parameters, such as the learning strategy for the associative memory, the coding strategy and the sparseness of the generated codes, to the given problem. If the parameters are chosen unfortunately the approach might yield poor classification results.

With respect to computation time the Dempster-Shafer alternative, the fusion strategy utilising silimarity preserving codes and the inter-state decision template method are the most expensive as all classifiers within the hierarchy are used and additional calculations for transforming the classifier outputs and combining the individual classification results are needed. For the decision-tree-like method and the end node method only part of the classifiers are evaluated. For the voting approach all classifiers have to be evaluated, but no time consuming calculations need to be performed thereafter.

Thus in time critical applications an efficient approach would be to first use the simple and faster decision-tree-like method that already yields good classification results to classify the objects in question. If this method does not yield unambiguous results, the more time consuming methods such as the evidence-theoretic approach, the inter-state decision templates or the fusion strategy based on gen-

erated codes should be used. If time is no critical factor, the usage of the more exhaustive approaches is justified and recommended.

Since the individual classifiers within the hierarchy can be evaluated independently of each other, it would be possible to evaluated all classifiers in parallel. Thus the difference in time on multi-processor machines is solely determined by the additional calculation required to combine the classifier outputs. Thus if all classifiers are evaluated in parallel the time aspect becomes less significant.

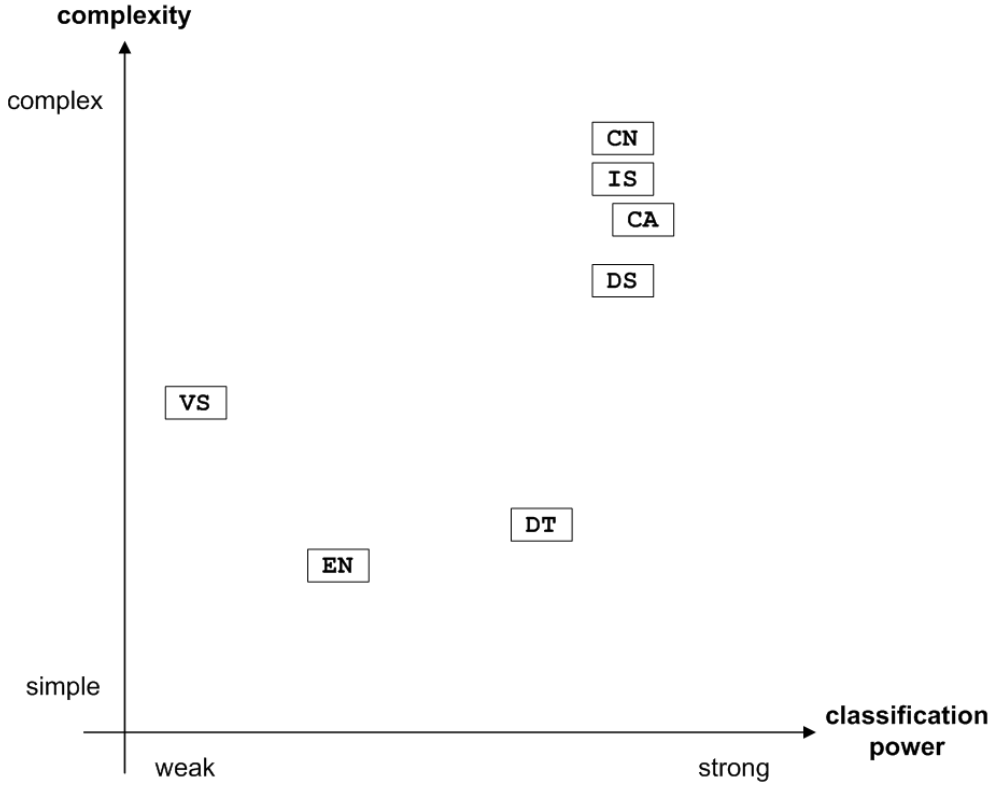

Figure 6.9: *Comparison of the different fusion strategies with regards to complexity and classification power. The strategies evaluated were the Dempster-Shafer method (DS), the decision tree method (DT), the voting scheme method (VS), the end node method (EN), the retrieval strategy utilising similarity preserving codes and associative memories (CA), the retrieval strategy utilising similarity preserving codes and non-hierarchical nearest-neighbour classifiers (CN) and the inter-state decision template method (IS).*

When not using the decision tree approach the advantage of the availability of intermediate classification outputs and the resulting savings of computation time do no longer apply as all classifiers within the hierarchy or all end nodes respectively need to be evaluated. However, the Dempster-Shafer approach provides not only the resulting class but also a measure how likely the presented samples belongs to the specific classes. The voting scheme method and the end node evaluation

approach also provide measures for all classes. The inter-state decision template approach and the approach utilising similarity preserving codes only provide the resulting class and if any the alternatives that are also likely.

A major drawback of the decision-tree-like evaluation method is the fact that there is no possibility to later on correct misclassifications that occur at higher levels of the hierarchy. As the evidence based approach considers all classifiers within the hierarchy a misclassification at higher levels of the hierarchy can be compensated for if the decisions made by the classifiers at the lower levels are correct.

The evidence theoretic approach can only compensate misclassifications at higher levels of the hierarchy. If the misclassification takes place at the responsible leaf node, this wrong decision cannot be corrected any more. The evidence theoretic approach can also not compensate for misclassifications where the majority of the classifiers supports the wrong decision.

In figure 6.9 the different retrieval strategies developed within the context of this work are assess with respect to classification performance and computational complexity.

7 Adaptive Incremental Learning of Novel Classes

Within the context of this thesis methods for incrementally learning new classes have been developed. This chapter first sketches the basic principles when incrementally extending classifier hierarchies. Afterwards the different strategies for effectively adding new classes to the hierarchy are presented. Another aspect of incremental learning dealt with is the incremental adaption of RBF networks and the retraining of incrementally learnt hierarchies in order to improve the classification performance. This chapter concludes with a discussion of the developed adaptive incremental learning approach.

7.1 Incremental Learning of New Classes

An important aspect of object recognition systems that are deployed in non-trivial real-world environments which are subject to numerous changes is the ability of the system to adapt to new situations. Thus the ability to incrementally learn new classes during run-time is an essential skill for such systems.

Requirements for the incremental learning are on the one hand the attainment of a sufficient classification quality for the newly learnt classes while not negatively affecting the classification quality of the previously learnt classes and on the other hand fast training times.

In the following an approach is proposed in which the incremental learning of new objects is performed in two stages. In the first stage fast but less sophisticated methods are used to obtain initial results, i.e. the novel objects are learnt but the recognition rate might be weak. In this first stage the recognition rate can be improved by using a similar method to retrain the new object with additional data. In a second stage more complex algorithms are used to adapt the system and to further improve the classification results.

For the incremental learning of a novel object N data samples $\tilde{S} := \{(x^\nu, \tilde{c}), \nu = 1, 2, ..., N\}$ of the new class $\tilde{c} \notin C$ are needed, where C is the set of classes known so far and the number of samples N available for the new class $\tilde{c}$ is considerably lower than the number of samples used for learning the already known classes C.

Before learning new objects it is necessary to identify whether the presented object is already known or whether it is in fact a new object. This is accomplished by presenting the new data x^ν to the trained classifier and taking the strength of the classifier response into account. Thereby a strong response is considered as an unambiguous decision and weak responses indicate a dubious decision which could be evoked by unknown classes or if the object to classify bears resemblance to more than one class. The thresholds for this are derived from the classifier responses when testing the known data. If a significant majority of the new data is unambiguously classified as a certain class c it is assumed that the object is already known and belongs to class c, hence nothing is learnt. Otherwise the object is regarded as a hitherto unknown object. If an object is identified as unfamiliar it is learnt by fitting it into the hierarchy and if necessary retraining the affected nodes.

Within the scope of this thesis two different methods for fitting the new classes into the hierarchy were developed. These methods are presented in the following two sections.

7.2 Incremental Learning of New Classes by Adding New Leaves

When learning new classes by adding leaves the new class $\tilde{c} \notin C$ associated with the unknown object is inserted into the hierarchy as a new leaf. The position of the new leaf is determined by classifying all new data x^ν and evaluating the corresponding paths through the hierarchy. The leaf will be inserted where the paths start to diverge. As complete identicalness for all data cannot be presumed even at the root node since the network has not been trained with this data a certain variance needs to be considered. Otherwise the new leaf would in most instances be added at the root node. Thus the common path is successively determined similar to retrieving the classification result. Starting with the root node all new samples are presented to the corresponding classifier. Depending on the classification results either the next node in the path is determined or the search is stopped and the new class is added as a new leaf at this node.

If all new samples x^ν are assigned to one successor node j by classifier i node i is added to the common path without retraining and classifier j is the next classifier. If the classification result of one classifier is not completely consistent, i.e. not all new samples but a significant majority of the data points were assigned

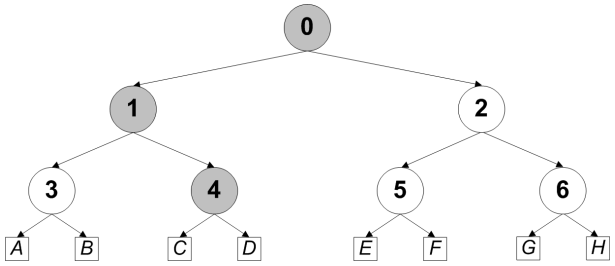

Figure 7.1: *Identification of the common path which all new samples take through the hierarchy. All new samples of the unfamiliar object are presented to the hierarchy. Their pathes through the hierarchy are compared and the least common path starting with the root node is determined.*

to the same successor node j, this classifier is retrained using the samples of the known objects S or in case of large training data sets only a subset of S as well as the new samples $\tilde{S}$ and node i is then added to the common path and the classification results for this node are obtained. If there is no clear decision, i.e. no significant majority of the samples is assigned to the same successor node, or if an end node is reached, the new class $\tilde{c}$ is inserted as an additional leaf of this node and the node is afterwards retrained.

The identification of the common path is illustrated in fugure 7.1. Figure 7.2 depicts how a new class is inserted into the hierarchy by means of the proposed incremental learning approach at the end of the so identified common path.

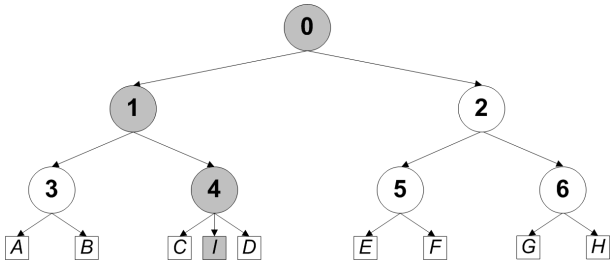

Figure 7.2: *Example of the incremental learning of a new class I by adding a leaf. At first the position where to insert the new class is determined. Then the new class is added as a new leaf and the affected nodes are retrained. Here the new class is added to node 4. The classifiers 0 and 1 are retrained with the samples of the additional class I and the samples used for previous training. One additional node is added to the output layer of classifier 4 and the classifier is then retrained.*

This incremental learning approach can also be used for non-hierarchical neural networks. However, here it is not necessary to determine the position were to insert the new leaf as non-hierarchical networks can be regarded as hierarchical networks consisting only of one node, but a retraining of the complete network takes place whereas only parts of the hierarchical network is amended.

This approach is a fast method for adding new clsses to existing classifier hierarches. Nevertheless this method does not further expand the hierarchy structure, i.e. only leaves are added to the existing hierarchy structure and the hierarchy depth is not increased.

7.3 Incremental Learning of New Classes by Adding New Nodes

An alternative approach for adding new classes to the hierarchy more resembling the hierarchy generation process is not to simply add the new classes as new leaf, but to add new nodes to the hierarchy by utilising unsupervised k-means clustering. At first the common path and the nodes on this path that need to be retrained are determined as described in section 7.2. Instead of adding a new leaf the part of the hierarchy emerging from the last common node is reconstructed analogue to the hierarchy generation (see section 6.2). This is depicted in figure 7.3.

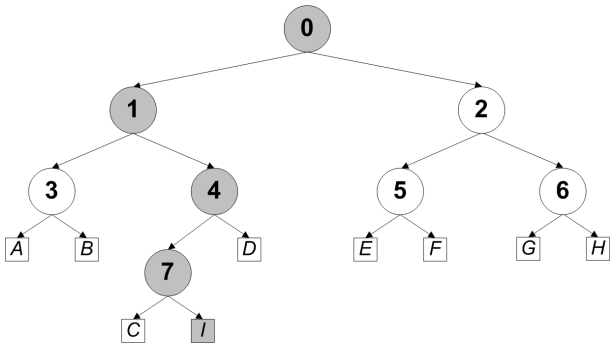

Figure 7.3: *Example of the incremental learning of a new class I by adding new nodes. At first the position where to insert the new class is determined by identifying the common path which all new samples take through the hierarchy. Then the new class groupings are calculated by means of unsupervised k-means clustering, new nodes are added accordingly and the affected nodes are retrained. Node 4 is rebuilt and a new node 7 is inserted. The classifiers 0 and 1 are retrained.*

This alternative approach is more time consuming than the previous approach, but further adapts the hierarchy structure as new nodes are generated. This allows for an incremental building of the hierarchy.

7.4 Incremental Training of Radial Basis Function Networks

The retraining or incremental training of the classifiers is conducted by adding a new neuron to the hidden layer and then retraining the output weights with the joint sample set of old and new samples $\hat{S} = S \cup \tilde{S}$. The centre c_{k+1} of the new prototype is determined by the mean of all new samples: $c_{k+1} := \frac{1}{N} \sum_{\nu=1}^{N} x^{\nu}$ (here k is the number of prototypes of the RBF classifier before learning). The width σ_{k+1} of the corresponding gaussian function is set to the mean distance of the new samples to the new centre: $\sigma_{k+1} := \frac{1}{N} \sum_{\nu=1}^{N} x^{\nu} - c_{k+1}$, and the new output weights are learnt by calculating the pseudo-inverse [64], which is a fast method already yielding good classification results.

The incremental training of the classifiers by adding a single prototype works well for data not spread widely. In order to account for scattered data the usage of more prototypes could yield better results. Hence two other ways of rapidly adding initial prototypes have been exploited, namely using all new samples as prototypes and performing a k-means clustering with $k = N/2$ clusters where N is the number of new samples. For each classifier to be retrained it must then be decided which method to use. Initially the method of adding one prototype is used. If optimal results, i.e. all samples are classified correctly, are achieved with this method, nothing more is done. Otherwise the k-means method and the method of adding all samples as prototypes are also performed and the one achieving the best classification results is chosen. To evaluate this the mean classification error per class

$$MCEC(\hat{S}) = \frac{1}{|C|+1} \sum_{i \in C \cup \tilde{c}} \frac{1}{|\{x^{\mu} \in \hat{S} : t^{\mu} = i\}|} \sum_{\{x^{\mu} \in \hat{S} : t^{\mu} = i, K(x^{\mu}) = i\}} 1 \qquad (7.1)$$

is calculated.

The incremental training of the RBF networks is used when adjusting the nodes during the incremental learning, but it is also used when simply retraining an already learnt class.

7.5 Retraining

Within the context of this work two strategies for further improving the classification performance of incrementally trained hierarchical neural network classifiers have been proposed. The first method is a fast online retraining of the hierarchy using additional samples that takes place during run-time and the second method

is a more extensive offline approach utilising sophisticated learning schemes and is applied after run-time.

A mechanism similar to the incremental learning approach is applied when retraining already learnt classes utilising additional training samples. The only differences are that no additional leaf or node is added and that the path through the hierarchy of the new class is already known. The single classifiers on this path are retrained if there are any incorrect or ambiguous decisions. This retraining can be used in order to further improve the classification results of the new classes learnt online that are only represented by few samples.

Scenarios for retraining an already learnt class could be when a class is only represented by few samples or the classification performance for this class is not satisfactory. The classification rate of novel classes is likely to be lower than the classification rate for the previously learnt classes. Thus a retraining of all affected nodes can yield improved classification results once sufficient additional samples for the novel class are available. Another reason could be that a new instance of a known class with noticeably different characteristics has to be learnt.

In order to improve the classification results a more elaborate and more time-consuming learning method can be applied to the incrementally learnt hierarchies after the first training took place as described above. This online learning phase therefore is followed by an offline learning phase where more sophisticated learning algorithms such as three phase learning (see section 2.4.4.3.3) are used which will further improve the classification performance. All nodes to which the novel classes have been assigned are newly trained using elaborate learning schemes. As these algorithms are very time consuming it might not be advisable to apply them during run time.

7.6 Discussion

As a consequence of the hierarchy composition when adding new classes to an existing hierarchy only parts of the hierarchy need to be amended while the rest remains unchanged. This makes classifier hierarchies applicative for adaptive incremental learning.

The adaptive incremental learning approach is a fast method for soundly learning new classes during run-time. The usage of fast training methods provides for rapid learning of the new classes which meets the requirements of a real-world robotic application. The fact that only parts of the hierarchy have to be adapted also adds to this.

The two different strategies for adding new classes to existing classifier hierarchies developed in the context of this thesis show different complexity. If only a limited number of new objects have to be learnt and if time is a critical factor, adding the

new classes as leaves would be the preferred method. In longer intermissions the hierarchy could then be retrained or even restructured in order to further improve the classification performance. If for the scenario at hand several classes have to be learnt and time is a non-critic factor, adding the novel classes by adding new nodes would be the preferred method.

A drawback of the usage of simple but fast training algorithm is the lower classification accuracy resulting from this. A lower classification performance is also due to the fact that not all information is available when constructing the hierarchy but is only gradually available. This only allows for a suboptimal generation of the hierarchy. However these drawbacks can be compensated for by retraining or even rebuilding the hierarchy in an intermission phase. The approach nevertheless facilitates fast learning of novel objects while still being able to recognise previously learnt objects.

8 Distributed Similarity Preserving Sparse Binary Codes

The problem of finding meaningful representations is an important question in the field of artificial neural networks. A special aspect of this problem is the conversion of non-binary data samples into sparse binary codes such that existing similarities between data samples are reflected in the generated codes. These similarity preserving sparse codes are of particular interest when dealing with associative memories. In the following different ways of generating such codes from hierarchical neural network classifiers are presented.

8.1 Generation of Code Vectors

When generating codes from artificial neural networks a straightforward approach is to use the neural activation of the different layers. In case of radial basis function networks the activations of the hidden layer and the activation of the output layer come into question. The number of neurons in the hidden layer is defined by the problem complexity and can be varied whereas the number of output neurons is determined by the number of classes of the classification problem to be solved and it cannot be chosen at liberty. Moreover the number of hidden neurons is usually notably larger than the number of output neurons. The activation of the output neurons is information at a more abstract level and is rather symbolic. The activation of the hidden neurons is closer to the data level and thus represents rather sub-symbolic information.

The characteristic of radial basis functions to respond strongly to input similar to the learnt input and with decreasing strength in case of increasing distance of the presented input to the learnt in feature space the usage of RBF classifiers is suggesting when attempting to represent similarities in the feature space. In [19] [20] an ensemble of RBF classifiers has been successfully used for categorising objects according to visual similarity. In the scope of this appearance-based approach the individual classifiers were each trained to recognise one class. The

combined output of the classifiers was found to encode similarity.

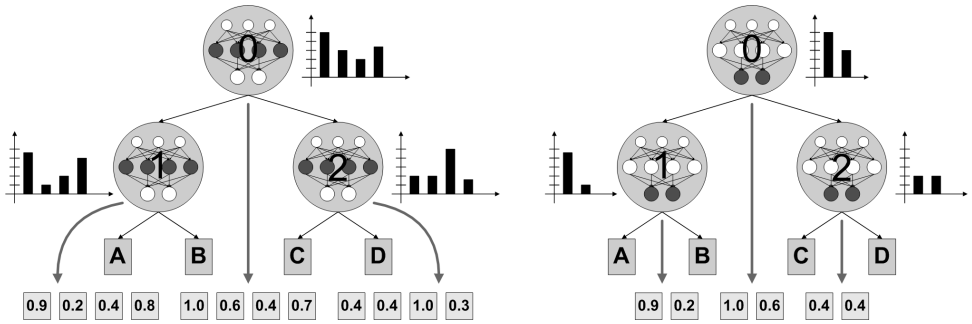

Figure 8.1: *Codes from classifier hierarchies can either be generated from the activation of the hidden layer of the individual classifiers (left) or from the activation of the output layer (right). Next to each classifier the activation of the considered layer is depicted as bar chart. Below the hierarchy the emerging vector containing all activations is shown. This vector forms the basis for the subsequent calculation of the code.*

The sample to be classified is presented to all classifiers within the hierarchy and their neural activations, either the output activation or the activation of the hidden layer, are then concatenated to form the code vector. The concatenation is carried out level-wise starting with the root node. Figure 8.1 depicts the generation of codes from the hidden and the output layer activation respectively.

There are diverse ways of controlling the sparseness of the generated codes. The strategies used in the scope of this work are defining the number or the percentage of active neurons or using a threshold. The different approaches for impacting the sparseness are equivalent and can be converted into each other. The choice of the technique for controlling the sparseness therefore only depends on the constraints of the given problem.

If the sparseness is controlled by defining the number $1 \leq a \leq t$ of active neurons the a strongest activations of the concatenated activation vector are kept and all other activations are set to zero. This way of controlling the sparseness of the codes is shown in figure 8.2. The number of active neurons can also be defined as a proportion $p \in [0, 1]$ of the total number of activations t. From this percentage the corresponding number of active neurons $a = \lfloor t \cdot p \rfloor$ can easily be calculated. A different way of manipulating the sparseness of the code vector is the usage of a threshold θ. All activations that are smaller than the threshold are set to zero. The activations equal to or greater then the threshold are retained. Figure 8.3 illustrates this. The predefined numbers, percentages or thresholds can used either globally referring to the complete hierarchy or locally referring to the individual levels or to the single classifiers.

In order to extenuate the abruptness of the suppression of the weaker activations activations that are only a predefined percentage of the weakest so far considered

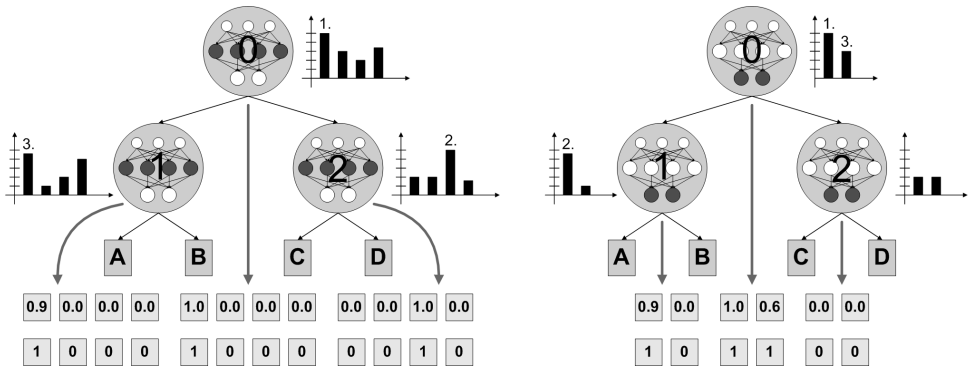

Figure 8.2: *Controlling the sparseness of codes generated from classifier hierarchies by defining the number of the strongest activations to be considered. The strongest activations are numbered in descending order according to their strength. This strategy can be applied to both the activation of the hidden layers (left) and the activation of the output layer (right).*

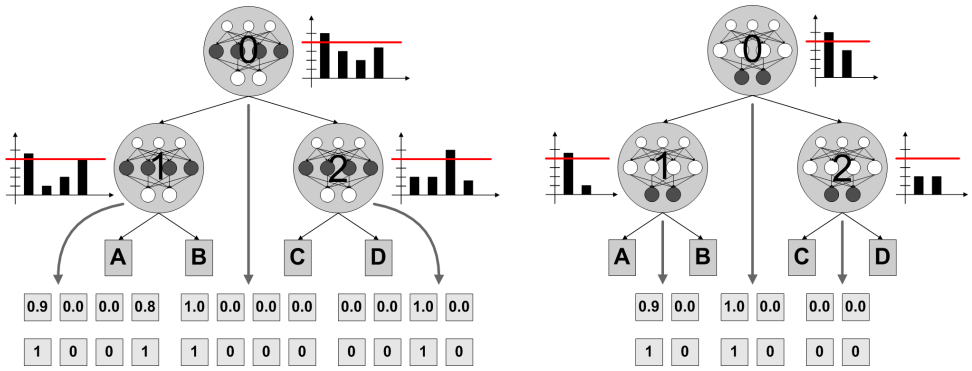

Figure 8.3: *Controlling the sparseness of codes generated from classifier hierarchies by applying a threshold. This threshold is marked in the bar charts as horizontal line. This strategy can be applied to both the activation of the hidden layers (left) and the activation of the output layer (right).*

activation lower are also considered. If there is a large distance between the weakest of the considered activations and the next lowest activation no modification is caused, whereas if there are activations approximately as strong as the weakest activation still retained it would be unnatural to suppress these activations.

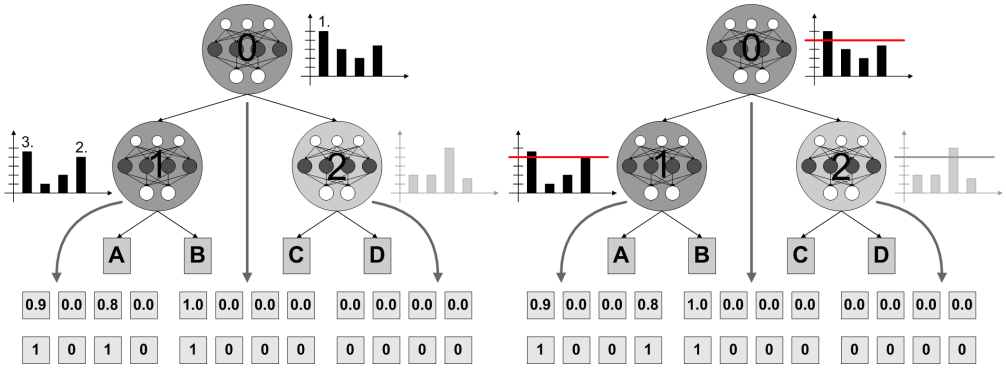

Figure 8.4: *Generating codes considering the classification path. The classifiers that do not belong to the classification path are greyed out as their activations do not contribute to the code generation.*

In order to attach more importance to the classification result only the activations of the nodes within the decision path that emerges when classifying the sample utilising the decision-tree retrieval strategy can be considered. The activations of all other nodes are set to zero. This is depicted in figure 8.4. From this coding scheme a simple strategy for generating a code vector can be derived, which directly encodes the classification result analogous to the decision tree retrieval strategy (see section 6.4.1). For each classifier along the classification path the output neuron activated strongest is retained. Figure 8.5 illustrates this.

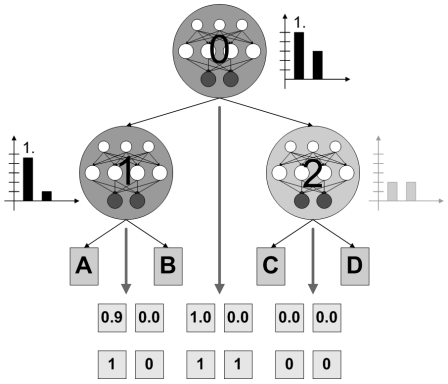

Figure 8.5: *Generating codes by directly encoding the classification path. For each classifier in the classification path the strongest output activation is considered.*

Another way of including the actual classification result in the generated code is to add l additional elements to the code vector, where l is the total number of

classes. These additional l elements are used to encode the resulting class l^* by means of a one-out-of-l coding scheme, i.e. all elements are set to zero except for the l^*st element which is set to one. This strategy is shown in figure 8.6.

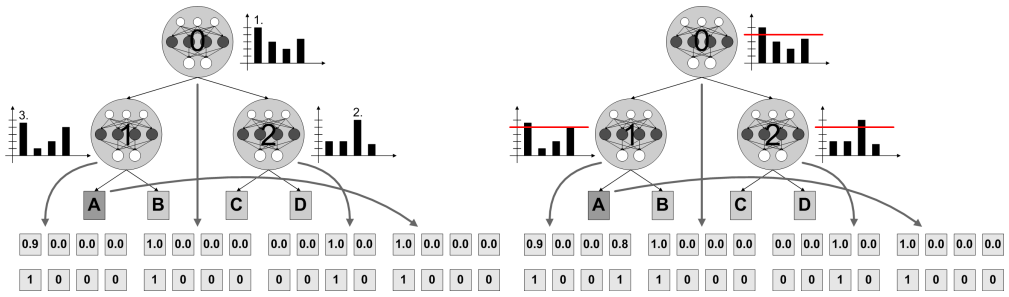

Figure 8.6: *Generating codes by inclusion of the classification result.*

If the classifiers provide additional information about the quality of the respective classification result this quality can also be encoded by adding two additional elements to the activation vector of each classifier. The classification quality is expressed by one of three states. The first state is related to the case if a clear decision for exactly one class with negligible uncertainty can be made. In the second case no clear decision in favour of one class can be made but several classes seem to be qualified. Thus the classifier is in doubt. The third state is assigned if no decision for any of the given classes can be made, i.e. none of the classes seems to be qualified. The sample to be classified is then likely to be an outlier and to belong to a different class. The two additional vector elements represent the doubt and the outlier state respectively. They are set to one if the classifier decision is dubious or it reveals nescience respectively and they are set to zero otherwise.

The different ways of generating the codes, such as using the activation of the hidden layer or the output layer, considering only nodes along the classification path or including the classification result or quality, and controlling the sparseness of the codes, such as using thresholds or defining the number of ones or the percentage of ones in the binary code either globally or locally, can be combined quite arbitrarily.

In a final step the codes can be binarised, i.e. the codes are transformed from $\mathbb{R}^t$ to $\{0,1\}^t$. This is achieved by setting all values greater than zero to one. The other values are set to zero. But if applicable the codes can also be used in their continuous form.

8.2 Discussion

There are various ways of generating meaningful codes from classifier hierarchies. Within the scope of this research a number of straightforward generating strategies have been developed. The different methods can be chosen according to the task they are needed for.

As the hierarchy is built exploiting similarity regarding the used features the codes generated from the hierarchy are similarity preserving.

When only the classification result is of interest the generation of codes from the hierarchy lacks motivation, but when e.g. the classification is embedded in a superordinated cortical model that is realised by means of associative memories the necessity of the usage of codes becomes obvious (see [41] [23]). These codes are then a suitable representation for subsequent processing with associative memories.

Part III: Application to 3D Object Recognition and Evaluation

In this part the functionality and the general applicability of the proposed approach is demonstrated by means of statistical experiments and by presenting different domains of applications.

9 Applications

The suitability and universality of the approach developed within the scope of this thesis could be shown in several classification experiments with different neural classifiers and data sets from various domains as well by successfully implementing the proposed approach on a robot and the effective testing of the implementation in a realistic scenario.

9.1 Visual Object Recognition

A special classification problem is visual 3D object recognition. The appearance of three-dimensional objects shows great variance depending on the pose of the object with respect to the beholder making the recognition of three-dimensional objects a complex endeavour. There are two diverging types of approaches to 3D object recognition: primitive-based (also called model- or structural-based) approaches and view-based approaches. The principle behind the primitive-based approaches is the decomposition of objects into volumetric components resulting in a three-dimensional viewpoint invariant representation of the object. These approaches are object centred. View-based approaches are viewpoint dependent. The three-dimensional objects are represented by a collection of two-dimensional characteristic views showing the object under different poses and illumination conditions. From these views abstract features are acquired. These approaches describe the object in a viewer centred manner by view-specific two-dimensional representations.

View-based recognition of three-dimensional objects is typically performed on information gathered from a single two-dimensional image. Therefore a set of features is extracted from this image and is compared against object models learnt by neural networks. Within the scope of this work the recognition process

is performed in the following three stages:

1. Object localisation
2. Feature extraction
3. Object classification

As the camera images do neither solely contain the object of interest but also other objects as well as background, nor is assured that the object of interest is always located in the centre of the image, it is at first necessary to localise the potential objects within the image. Thus initially a figure-ground segmentation is performed which determines regions of interest within the image potentially containing objects to be classified. A region of interest is the smallest rectangle containing such an object of interest. This upstream stage of localising the objects reduces the computational effort of the subsequent stages since thenceforward all calculations are conducted on a part of the complete image. Moreover it attains position and scale independence of the recognition process. As the figure-ground segmentation is not in the main focus of this thesis only simple not very sophisticated algorithms, such as thresholding or identifying regions of similar predefined colours, are used to determine the regions of interest.

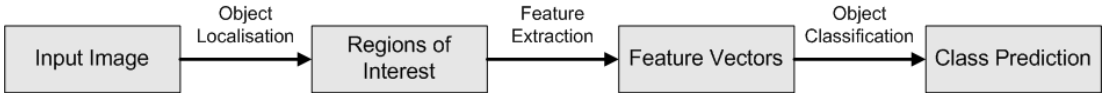

Figure 9.1: *Three-stage process for object recognition.*

Afterwards the different characteristic feature vectors such as orientation or colour histograms describing the three-dimensional objects are extracted from these regions of interest. For each region of interest all utilised feature types are extracted. The different feature types used within the scope of this thesis are explained in more detail in chapter 11. Using representative and meaningful features instead of the raw image data is reasonable insofar as the image data contains a great many of redundant and dispensable information. Moreover the usage of feature vectors yield a considerable reduction of the dimensionality as instead of using all pixels of the camera image condensed information is utilised.

In a final stage the extracted feature vectors are presented to a classifier trained to recognise the contemplable objects. The result of this stage are the predicted classes of the localised objects.

The individual stages and the interstage results, such as regions of interests, feature vectors and classification results, as well as their interrelationships are depicted in figure 9.1.

9.2 MirrorBot Project

The approach of hierarchical neural network classifiers developed in the scope of this thesis has successfully been implemented on a robot, an ActivMedia PeopleBot. This implementation was part of the MirrorBot project of the European Union (EU-IST-FET). In the context of the MirrorBot project a scenario has been defined where the robot has to perform object manipulating tasks such as grasping or pointing to certain objects. The contemplable objects lie on a table. The robot is situated near to the table and has to respond to spoken commands concerning these objects. The scenario is depicted in figure 9.2. The commands are formulated utilising a simple language with a restricted vocabulary and an elementary grammar. An example of such a command is "Bot show orange".

Figure 9.2: *MirrorBot test scenario. In the test scenario the robot is situated in front of a table. Different objects are lying on this table. The robot has to grasp or point to specified objects.*

In order to be able to fulfill the requested tasks the robot needs to be capable of recognising a certain set of objects. The fruits data set described in section 10.1.1 was generated in the scope of this project. It was recorded under conditions simulating the described scenario: Images of fruits placed on a table were taken from a robot's point of view. The data set depicts a large part of the objects used in the scenario. The object recognition is performed by means of hierarchical

neural network classifiers developed in this work which were trained to classify seven different fruits. For the training features extracted from images of the fruits were used. The features that proved most suitable for the recognition under challenging constraints of real-world robotic applications were orientation and colour histograms.

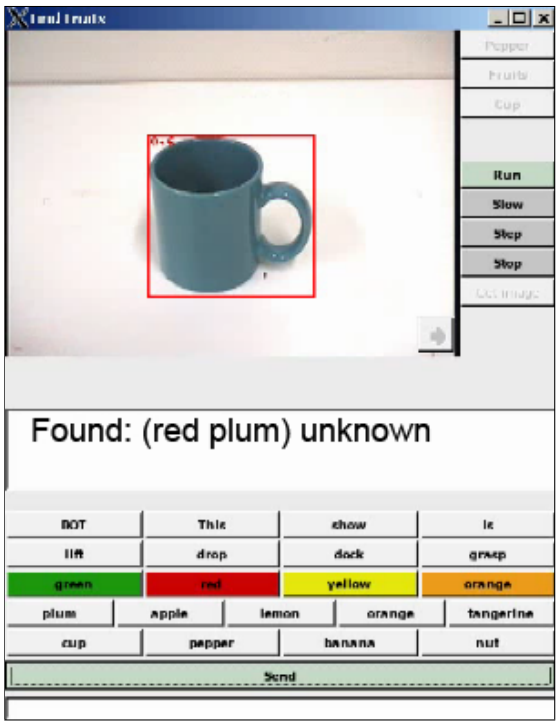

Figure 9.3: *MirrorBot control GUI.*

The conditions are characterised by changing lighting conditions, massively alternating cast shadows, strongly varying object positions, views and size, occlusion, only partially visible objects due to the limited visual field, cluttered backgrounds and numerous unfamiliar objects. The varying object views and size is dealt with by applying a figure-background separation stage before the actual object recognition stage. During this stage the object is localised and a region of interest solely containing the object is determined. This regions of interest are varied during the feature extraction process and thus yielding robustness against not perfectly determined regions of interest or only partially visible objects. The issue of unknown objects is addressed by not only giving the classification result, i.e. the class to which the object in the robot's visual field most likely belongs, but also the quality of the classification, i.e. whether the decision is certain or uncertain or whether the object is unfamiliar. Also the possibility of incrementally learning new objects during run-time was successfully implemented. The learning of new objects is triggered by the command "This is" followed by the object name. Then the robot is shown ten different views of the novel object. These views are then

learnt using the incremental learning approach. In typical scenarios it could be demonstrated that the previously learnt classes are still correctly recognised and the new objects can also be classified. The robotic platform also allowed to show that more than one new object can be successfully learnt.

Figure 9.3 shows the graphical user interface used to instruct the robot. The interface also shows the robots camera image, the regions of interest (in the example shown there is only one region of interest in the robot's visual field) and the outcome of the robot's classification process.

9.3 Discussion

The approach developed within the scope of this thesis has been applied to different problem areas showing the universality of this approach. The main problem domain was the visual recognition of three-dimensional objects. The feasibility of the proposed approach was not only demonstrated by statistical evaluation by means of previously recorded data sets but also by the implementation on a robot under real-world conditions where beyond the typical object recognition problems additional difficulties such as changing illumination conditions, inaccuracies of the visual sensors and varying environments arise. The approach can deal with these difficulties sufficiently well such that the approach could successfully be integrated on a mobile robot.

10 Data

Miscellaneous data sets from different domains were used to evaluate the proposed approach. One part of the data sets are two-dimensional images of three-dimensional objects. From these images different features as described in chapter 11 were extracted from these images and used for classification. Another part of the data sets originates from the domain of optical character recognition were the objects to be classified are letters and digits. These data sets are already preprocessed and provide a specific set of features each and as such were used as benchmarking data sets. In the following these data sets are described.

10.1 3D Data Sets

10.1.1 Fruits

The fruits data set consists of RGB colour images of the size 384×288 pixels of 7 different fruits. There are 120 images per object. The fruits were placed on a white table and were recorded from different views and at different positions. Figure 10.1 shows sample images of the seven different fruits.

Figure 10.1: *Fruits*

This data set was generated within the scope of this thesis.

10.1.2 Columbia Object Image Library (COIL)

The Columbia Object Image Library offers two data sets of different size. Both data sets consist of images of three-dimensional objects. The objects are articles

use such as toys, cups, cosmetics and drugs.

10.1.2.1 COIL-20

The COIL-20 data set [62] consists of 1440 size-normalised grey-scale images of 20 different objects in front of a unicoloured black background. The images were recorded under clearly defined conditions. The images are of size 128×128. There are 72 images per object from different views at pose intervals of 5 degree covering a total of 360 degrees. Figure 10.2 shows frontal views of the twenty different objects.

Figure 10.2: *COIL-20*

10.1.2.2 COIL-100

The COIL-100 data set [61] comprises 7200 size-normalised colour images of 100 different objects. The objects are recorded in front of a single-coloured black background. The recording conditions were clearly defined. The objects were recorded from different views at pose intervals of 5 degree covering a total of 360

degrees resulting 72 images per object. The image size is 128×128. Frontal views of the different objects are shown in figure 10.3.

Figure 10.3: *COIL-100*

10.2 Benchmarking Data Sets

10.2.1 Letter Image Recognition Data

The Letter Image Recognition data set [31] consists of 20,000 samples of the 26 capital letters in the English alphabet. From black and white images of the characters 16 primitive numerical features were derived representing simple statistical

characteristics of the pixel distribution. The features are integer valued. They are linearly scaled to a range from 0 to 15. To generate the images 20 different fonts were used and randomly distorted resulting in 20,000 unique samples. Figure 10.4 shows examples of the samples used.

Figure 10.4: *Letter Image Recognition Data*

This data set represents a rather complex classification problem due to the wide variety of the fonts used and the simplicity of the extracted attributes.

The sixteen different features are described in the following.

1. **x-ROI:** Horizontal position of the centre of the smallest rectangle completely containing the object, the so called region of interest. The pixels are counted from the left edge of the image.
2. **y-ROI:** Vertical position of the centre of the region of interest. The pixels are counted from the bottom of the image.
3. **width-ROI:** Width of the region of interest in pixels.
4. **height-ROI:** Height of the region of interest in pixels.
5. **object-pix:** Number of pixels in the image belonging to the object.
6. **x-mean:** Mean horizontal position of all pixels belonging to the object relative to the centre of the region of interest and normalised by the width

of the region of interest. For left-sided characters such as the letter "L" this feature will have a negative value.

7. **y-mean:** Mean vertical position of all pixels belonging to the object relative to the centre of the region of interest and normalised by the height of the region of interest.

8. **x-var:** Mean x variance, i.e. mean squared values of the horizontal pixel distances x-mean. For characters that show a wider separation in the horizontal direction such as the letters "M" and "W" this feature has a higher value.

9. **y-var:** Mean y variance, i.e. mean squared values of the vertical pixel distances y-mean.

10. **xy-correlation:** Mean x-y correlation, i.e. mean product of the horizontal and vertical pixel distances x-mean and y-mean for each object pixel. For diagonal lines oriented from bottom left to top right this feature takes positive values. For diagonal lines running from top left to bottom right it has negative values.

11. **x2y:** Mean of x * x * y, i.e. mean value of the squared horizontal distance times the vertical distance for each object pixel. This feature is a measure for the correlation between the horizontal variance and the vertical position.

12. **xy2:** Mean of x * y * y, i.e. mean value of the horizontal distance times the squared vertical distance for each object pixel. This feature is a measure for the correlation between the vertical variance and the horizontal position.

13. **x-edge:** Mean edge count left to right, i.e. mean number of edges detected from left to right over all vertical positions within the region of interest. An edge is defined as an object pixel directly above to the right of a background pixel or the image border. This feature allows to discriminate between letters like "M" and "W" or letters like "I" and "L".

14. **x-edge-y-correlation:** Correlation of mean edge count left to right with y, i.e. sum of the vertical positions of the edges x-edge detected in horizontal direction from left to right. For characters such as the letter "Y" which have more edges at the top of the region of interest, this feature will have a higher value.

15. **y-edge:** Mean edge count bottom to top, i.e. mean number of edges detected from bottom to top over all horizontal positions within the region of interest. An edge is defined as an object pixel directly above a background pixel or the image border.

16. **y-edge-x-correlation:** Correlation of mean edge count bottom to top with x, i.e. sum of the horizontal positions of the edges y-edge detected in vertical direction from bottom to top.

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M
Number of samples	789	789	736	805	768	775	773	734	755	747	739	761	792
Letter	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Number of samples	783	753	803	783	758	748	796	813	764	752	787	786	734

Table 10.1: *Class distribution of letter image recognition data. The number of samples per letter ranges from 734 to 813.*

Table 10.1 shows the class distribution of the data set, i.e. the number of samples per class.

The data set was evaluated inter alia in [31] and [21].

10.2.2 Handwritten Digits

The handwritten-digits data set [45] consists of 20,000 images of 10 handwritten digits from German postcodes. The grey valued images are of size 16×16 with 8 bit quantisation, i.e. 256 grey levels, and they are normalised in width and height. There are 2,000 images per class.

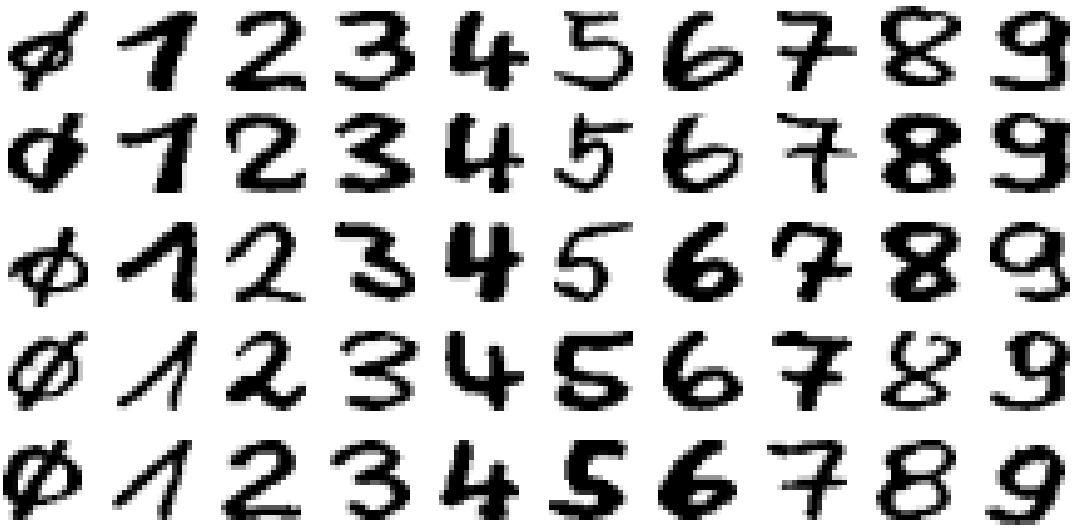

Figure 10.5: *STATLOG digits. Examples of the handwritten digits.*

The data set is divided into a predefined training and a test set containing 10,000 samples each. Each class is represented with 1,000 samples. This division is used in a couple of experiments in order to obtain benchmarking results.

The data set was evaluated inter alia in [45], [57] and [75].

Figure 10.5 shows some examples of the digits within this data set.

10.3 Discussion

The data used within the cope of this work differs widely inter alia in the domain of application, the type of patters to be classified, the number of classes, the number of samples, the number and types of features used and complexity. The tables 10.2 and 10.3 list the different properties [90] of the used data sets.

name	attribute type	number of attributes	data set size	number of classes	default accuracy [%]	entropy [bit]
letters	numeric (integer) [0, 15]	16	20000	26	4.06	0.6194
digits	numeric (integer) [0, 255]	256	20000	10	0.1	0

Table 10.2: *Classification of the different data sets used.*

name	data set size	number of classes	default accuracy [%]	entropy [bit]
fruits	840	7	0.1429	0
COIL-20	1440	20	0.05	0
COIL-100	7200	100	0.01	0

Table 10.3: *Classification of the different data sets used.*

The attribute type defines the data type of the elements of the feature vectors. Possible attribute types are binary, nominal, ordinal, numeric and mixed. The number of attributes gives the dimension d of the feature vectors. The data set size specifies the number of samples M within the data set. Another property is the number l of different classes that are represented within the data set. The default accuracy is the frequency of the most common class and reflects the complexity of the underlying classification problem. The entropy of a data set

calculates the expected amount of information for classifying a sample taking the class distribution of the data set into account:

$$E = -\sum_{i=1}^{l} P(C_i) \log_2 P(C_i). \tag{10.1}$$

The data sets consisting of images from three-dimensional objects, namely the fruits data set, the COIL-20 data set and the COIL-100 data set, were used to extract several different features from which define the attribute type and the number of attributes. The two-dimensional data sets, namely the digits data set and the letters data set, consist of a set of fixed features.

11 Features

One problem when classifying 3D objects from 2D camera images is the extraction of features representing the objects in a suitable way.

The selection of the features depends among other things on the objects to be classified. The features should be meaningful for the specific objects as different objects are characterised by different attributes. For the distinction of different fruits e.g. features representing colour and form of the object are an appropriate choice.

The described feature types differ in complexity, dimensionality, classification performance, discrimination power, expressiveness and suitability for the given classification task.

As in real world applications the camera images seldom contain solely the object of interest but also other objects as well as background it is necessary to first localise the object within the image and to separate it from the background. Once the pixels belonging to the object are identified the region of interest is placed around the object. The so called region of interest is the smallest rectangle containing the object. The features are not extracted from the whole image but only from this region of interest. As the region of interest is also rectangular the region of interest can be treated as an image $I(x, y)$. Some of the features are extracted from the mask $B(x, y)$ which is a binary image defining the object pixels.

This chapter describes the different feature types extracted from the regions of interest containing the object hypotheses.

11.1 Orientation Histograms

A feature type which represents the form of the object are orientation histograms [30][16], summing up all orientations within a given region of interest. Orientation histograms show the frequency of occurrence of the different directions of orientation in this region. When graphically visualising orientation histograms

the x-axis specifies the different orientations and the y-axis gives the frequency of occurrence of these orientations.

To calculate the orientation histograms the gradient in x and y direction of the grey value image $I(x, y)$ is calculated using an edge detector such as the Sobel or the Canny edge detector. The gradient angles are discretised therefor the gradient angle range is divided into b sections. The discrete gradient directions are weighted with the absolute gradient value and summed to form the orientation histogram. The orientation histogram provides information about the directions of the edges and their intensity.

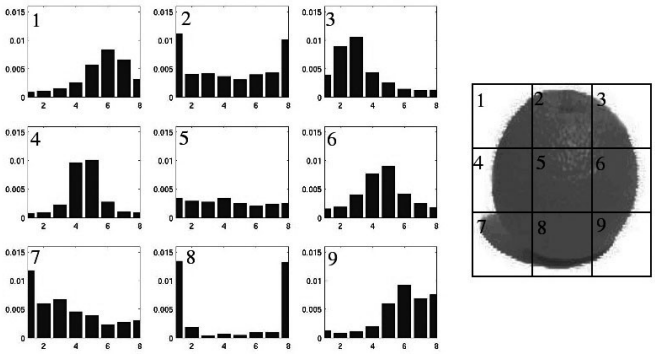

Figure 11.1: *The image is split into sub-images. For each sub-image an orientation histogram is calculated by summing up the orientations that occur in this sub-image. For reasons of simplicity non-overlapping sub-images are depicted.*

It has been found that the results achieved could be improved if not only one orientation histogram per region of interest is used to represent the form of the object but several orientation histograms are calculated from different parts of the region of interest. The region of interest is therefore split into $m \times m$ potentially overlapping parts of equal size. For each part a separate orientation histogram is calculated. The concatenated orientation histograms then form the $b \times m \times m$ dimensional feature vector. If the parts overlap the result improves further as this compensates for non-optimally detected regions of interest. On the one hand the number of orientation sections b should be high enough to be able to distinguish between the orientations occurring in the image, but on the other hand number of orientation sections b should be as low as possible in order to reduce computational costs. Figure 11.1 illustrates how an orientation histogram is generated.

The dimension of the feature vector depends only on the number of sub-images $(m \times m)$ and the number of sections b used to discretise the gradient angle. The orientation histogram is also largely independent of the resolution of the image used to extract the features.

For the data sets used suitable values for m are 1, 2, 3 and 4. An appropriate value for the number of different orientations b is 8. And an overlap of 20% gives

good results.

11.1.1 Orientation Histograms Utilising Sobel Operator for Edge Detection

In images edges are represented by areas with strong intensity contrasts, i.e. a strong ascent or descent of the intensity over a short distance. Thus the one-dimensional shape of an edge is a ramp. Consequently locating the maxima and minima of the first derivative of an image indicate the presence of an edge.

The Sobel operator [34] estimates the gradient $\nabla I(x, y)$ of a two-dimensional grey-value image $I(x, y)$. It consists of a pair of 3×3 convolution masks. One mask S_x estimates the gradient $I_x(x, y)$ in the x-direction (columns) and the other mask S_y estimates the gradient $I_y(x, y)$ in the y-direction (rows). These masks are shown in figure 11.2.

$$S_x = \frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array} \qquad S_y = \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}$$

Figure 11.2: 3×3 *Sobel masks.*

The grey-value image $I(x, y)$ is convolved with the two masks S_x and S_y respectively. The resulting images $I_x(x, y)$ and $I_y(x, y)$ give the horizontal and vertical orientations respectively:

$$I_x(x, y) = I(x, y) * S_x \tag{11.1}$$
$$I_y(x, y) = I(x, y) * S_y. \tag{11.2}$$

The gradient $\nabla I(x, y)$ is then given by

$$\nabla I(x, y) = \begin{pmatrix} I_x(x, y) \\ I_y(x, y) \end{pmatrix}, \tag{11.3}$$

the gradient direction θ is calculated as

$$\theta(x, y) = \arctan \frac{I_x(x, y)}{I_y(x, y)} \tag{11.4}$$

and the gradient strength $|\nabla I(x, y)|$ is defined as

$$|\nabla I(x, y)| = \sqrt{I_x(x, y)^2 + I_y(x, y)^2}. \tag{11.5}$$

To calculate the orientation histogram the gradient directions $\theta(x, y)$ are discretisised. The parameter b specifies the number of discrete orientations d_i and determines b orientation rages $[r_i^{low}, r_i^{high}]$. The orientation ranges are defined by $r_i^{low} = i\frac{180^\circ}{b}$ and $r_i^{high} = (i+1)\frac{180^\circ}{b}$ with $i = 0, 1, 2, ..., b-1$. The corresponding discrete orientations d_i are then given by $d_i = (i + \frac{1}{2})\frac{180^\circ}{b}$ with $i = 0, 1, 2, ..., b-1$. The gradient direction $\theta(x, y)$ is then assigned the discrete direction d_i if $r_i^{low} \leq \theta(x, y) \leq r_i^{high}$. For each discrete direction d_i the gradient strength $|\nabla I|$ of the pixels having this gradient direction is summed up.

11.1.2 Orientation Histograms Utilising Canny Operator for Edge Detection

Another more sophisticated way of calculating edges within an image is the Canny edge detector [12]. The gradients of the edges detected with the Canny edge detector are used to calculate an orientation histogram.

The detection of edges with the Canny operator is a multi-stage process.

Step 1: Noise reduction

In a first step the image is smoothed by convolving the image with a gaussian filter in order to reduce the noise. The result is a blurred but less noisy image. Figure 11.3 shows a sample gaussian mask.

$$\frac{1}{115}\begin{array}{|c|c|c|c|c|}\hline 2 & 4 & 5 & 4 & 2 \\ \hline 4 & 9 & 12 & 9 & 4 \\ \hline 5 & 12 & 15 & 12 & 5 \\ \hline 4 & 9 & 12 & 9 & 4 \\ \hline 2 & 4 & 5 & 4 & 2 \\ \hline \end{array}$$

Figure 11.3: *5×5 Gaussian filter. Discrete approximation of the two-dimensional gaussian function with $\sigma = 1.4$*

Step 2: Calculating the intensity gradient of the image

Afterwards the Sobel operator is used to calculate the intensity gradient of the smoothed image. As described above the gradient is estimated in horizontal and vertical direction. From these estimates the gradient strength and direction can be calculated.

Step 3: Non-maximum suppression

The aim of the non-maximum suppression is to find local maxima in the direction of the gradient and to suppress all others ensuring only one response to a single edge. The result of this non-maximum suppression are single pixel thin edges. The local maxima is found by comparing the strength of the gradient of a pixel with the strength of the gradient of its neighbors along the direction of the gradient. All other pixel are set to zero.

Step 4: Hysteresis

Pixels with high intensity gradients are more likely to belong to an edge. Thus a thresholding is performed eliminating pixels with low gradient strength. The so called hysteresis uses two thresholds t_{high} and t_{low} with $t_{high} > t_{low}$. All pixels whose gradient strength is greater than t_{high} are unconditionally accepted as edge pixels. Pixels with a gradient strength lower than t_{low} are immediately discarded and are set to zero. Pixels having a gradient value lower than t_{high} but higher than t_{low} are only regarded as edges if they are connected to a pixel already selected as an edge pixel.

The orientation histogram is then calculated alike to the orientation histogram utilising the Sobel operator on potentially overlapping subimages using different discretisations of the orientations. As the gradients of non-edge pixels are set to zero only gradients of edge pixels are taken into account.

11.1.3 Orientation Histograms Based on Opponent Colour Channels

Another complex feature type are orientation histograms calculated on opponent colour channels. Calculating the orientation histograms on the different channels of a colour space takes the colour information besides the form information into consideration. An appropriate colour space is the opponent colour space [11], where the different colour channels are based on colour information of opponent colours.

The initial trichromatic colour information is transformed into an achromatic (A: black / white) and two opponent chromatic channels (P: red / green and Q: yellow / blue). On each of these three channels an orientation histogram is calculated utilising the sobel edge detector as described in 11.1.1 resulting in three different feature types combining colour and form information.

The conversion from the RGB colour space to the APQ colour space is performed

by applying the following transformation:

$$\begin{bmatrix} A \\ P \\ Q \end{bmatrix} = \begin{bmatrix} 0.887 & 0.461 & 0.0009 \\ -0.46 & 0.88 & 0.01 \\ 0.004 & -0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{11.6}$$

The orientation histograms are then calculated on each of the three colour channels analog to the orientation histograms on the grey-value images utilising the Sobel edge detector. This results in three different feature types. As the previously described orientation histograms the orientation histograms on the different opponent colour channels can be calculated not only on the complete image but also on $m \times m$ potentially overlapping subimages. The orientation histograms of these subimages are then concatenated to $b \times m \times m$-dimensional feature vectors.

11.2 Colour Histograms

Colour histograms are calculated on the original RGB colour image and specify for each colour channel the frequency of occurrence of the different colour values, i.e. the number of pixels of the individual colour values are determined. For each of the three colour channels this frequency is calculated. In the graphical visualisation of colour histograms the x-axis specifies the colour value, the y-axis specifies the frequency of occurrence of the different colour values and the z-axis defines the colour channel.

The colour histograms are also calculated on $m \times m$ potentially overlapping subimages in order to make allowance for a certain amount of rotation of the object of interest. The colour histograms are $3 \times b \times m \times m$-dimensional feature vectors.

If the colour histograms are calculated on grey scale images simply a grey value histogram is calculated.

11.3 Curvature Histograms

The curvature histograms [52] are a feature type considering the cornerness. The x-axis shows the curvatures and the y-axis gives the frequency of occurrence of these curvatures.

The curvatures are calculated on the grey-value image using the structure tensor [36] [28]. The structure tensor is a measure which can discriminate between homogenous regions, edges and corners. The structure tensor J_0 is calculated by

the covariance matrix of the gradient $\nabla I(x, y)$ of the grey-value image:

$$J_0(x,y) = \nabla I(x,y) \cdot \nabla I(x,y)^T = \begin{bmatrix} I_x^2(x,y) & I_x(x,y) \cdot I_y(x,y) \\ I_x(x,y) \cdot I_y(x,y) & I_y^2(x,y) \end{bmatrix}. \tag{11.7}$$

The structure tensor J_0 is then weighted and averaged by convolving it with a two-dimensional gaussian filter G_ρ resulting in the matrix

$$J_\rho(x,y) = G_\rho(x,y) * J_0(x,y) = \begin{bmatrix} J_{xx}(x,y) & J_{xy}(x,y) \\ J_{xy}(x,y) & J_{yy}(x,y) \end{bmatrix} \tag{11.8}$$

where $*$ is the convolution operator.

The eigenvalues λ_1 and λ_2 of the smoothed structure tensor J_ρ are then calculated as

$$\lambda_{1,2}(x,y) = \frac{1}{2}\left(J_{xx}(x,y) + Jyy(x,y) \pm \sqrt{(J_{xx}(x,y) - Jyy(x,y))^2 + 4J_{xy}^2(x,y)}\right) \tag{11.9}$$

with $\lambda_1 \geq \lambda_2$.

Depending on the values for λ_1 and λ_2 the following can be stated about the image structure: If both eigenvalues are approximately zero ($\lambda_1 \approx \lambda_2 \approx 0$), no structure is given, i.e. the region observed shows constant intensity. The structure is intrinsically 0-dimensional. In case the greater eigenvalue λ_1 is clearly greater than zero and the smaller eigenvalue λ_2 is approximately zero ($\lambda_1 \gg \lambda_2 \approx 0$), the observed structure is intrinsically 1-dimensional, i.e. there is an edge in the image. If both eigenvalues are significantly greater than zero ($\lambda_1 \geq \lambda_2 \gg 0$), the observed structure is intrinsically 2-dimensional, i.e. the examined region shows e.g. a corner.

The curvatures c at position x, y are expressed as ratio of the two eigenvalues λ_1 and λ_2.

$$c(x,y) = \frac{\lambda_2(x,y)}{\lambda_1(x,y)}. \tag{11.10}$$

11.4 Orientation-Curvature Histograms

Orientation-curvature histograms [52] are multi-dimensional histograms relating orientations and local curvatures. When visualising orientation-curvature histograms the x-axis shows the orientations, the y-axis shows the strength of the

local curvatures and the z-axis gives the frequency of the occurrence of the respective orientations and curvatures.

The orientations and their strengths are determined utilising the Sobel edge detector (see section 11.1.1). The curvatures are calculated using the structure tensor as described in section 11.3.

11.5 Geometric Features

Geometric features [72] describe the two-dimensional shape of an object. They are calculated from the binary image $B(x, y)$ which discriminates pixels belonging to the object of interest O (value 1) and pixels belonging to the background (value 0). The resulting features are simple one-dimensional features. Five geometric features were used within the scope of this work which are described below: form factor f, roundness r, compactness c, extent e and aspect ratio of the bounding rectangle b. Moreover a concatenation of these five features forming a five-dimensional feature vector was also used.

The area A is defined by the number of pixels belonging to the object O.

$$A(O) = \sum_{p \in O} 1 = \sum_x \sum_y B(x, y). \tag{11.11}$$

The form factor f expresses the ratio of area A to perimeter P. The form factor of a circle is 1. For other shapes the form factor is smaller than 1. The form factor f is defined as

$$f = \frac{4\pi A}{P^2} = \frac{4\pi}{c} \tag{11.12}$$

where c is the compactness (see equation 11.15).

From the area A the equivalent circular diameter $D_{circular}$ can be calculated which is the diameter a circle with area A would have. It is defined as

$$D_{circular} = \sqrt{\frac{4}{\pi} A}. \tag{11.13}$$

The roundness r expresses the ratio of the squared equivalent circular diameter $D_{circular}$ to the squared maximal diameter D_{max}. The roundness of a circle equals 1 and is smaller than 1 for elongated shapes. The roundness r is defined as

$$r = \frac{D_{circ}^2}{D_{max}^2} = \frac{4A}{\pi D_{max}^2} \tag{11.14}$$

where D_{max} is the maximal diameter or the length.

The compactness c expresses the ratio of the equivalent circular diameter $D_{circular}$ to the maximal diameter D_{max}. The compactness of a circle is 1. The compactness c is defined as

$$c = \frac{D_{circular}}{D_{max}} = \frac{\sqrt{\frac{4}{\pi}A}}{D_{max}} = \frac{P^2}{A}. \tag{11.15}$$

Thus it holds $r = c^2$.

The area of the bounding rectangle A_{ROI}, the so called region of interest, is defined by the width w and the height h of the region of interest.

$$A_{ROI} = wh. \tag{11.16}$$

The extent e expresses the ratio of area A to the area of the region of interest A_{ROI}. The extend e is defined as

$$e = \frac{A}{A_{ROI}}. \tag{11.17}$$

The aspect ratio of the bounding rectangle b is determined by the width and the height of the bounding rectangle. It is defined as

$$b = \frac{\min(w, h)}{\max(w, h)}. \tag{11.18}$$

11.6 Hu Invariant Moments

The Hu invariant moments [37] are a set of absolute orthogonal two-dimensional moment invariants and one skew invariant moment. They can be applied to rotation, scale and translation invariant pattern recognition.

The seven Hu invariant moments were used as one-dimensional features as well as a concatenation of these seven moments forming a seven-dimensional feature vector.

The image $I(x, y)$ is considered as a probability density function.

A two-dimensional $(p + q)$th order moment m_{pq} of a grey-value image $I(x, y)$ of

size $M \times N$ is defined as

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q I(x,y) \tag{11.19}$$

where $p, q \in \mathbb{N}_0$.

The central $(p+q)$th order moment μ_{pq} of a grey-value image $I(x,y)$ is defined as

$$\begin{aligned} \mu_{pq} &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x-\overline{x})^p (y-\overline{y})^q I(x,y) \\ &= \sum_{m=0}^{p} \sum_{n=0}^{q} \binom{p}{m} \binom{q}{n} (-\overline{x})^{(p-m)} (-\overline{y})^{(q-n)} M_{mn} \end{aligned} \tag{11.20}$$

for $p, q \in \mathbb{N}_0$ where $\overline{x} = \frac{m_{10}}{m_{00}}$ and $\overline{y} = \frac{m_{01}}{m_{00}}$. The centroid of the image is $(\overline{x}, \overline{y})$.

The central moments are calculated using the image's centroid such that the calculation corresponds to the calculation of regular moments of an image that has been shifted such that its centre coincides with its centroid. Thus central moments are translation invariant.

The central moments of order zero to three are

$$\mu_{00} = m_{00} = \mu \tag{11.21}$$
$$\mu_{10} = 0 \tag{11.22}$$
$$\mu_{01} = 0 \tag{11.23}$$
$$\mu_{11} = m_{11} - \overline{x} m_{01} = m_{11} - \overline{y} m_{10} \tag{11.24}$$
$$\mu_{20} = m_{20} - \overline{x} m_{10} \tag{11.25}$$
$$\mu_{02} = m_{02} - \overline{y} m_{01} \tag{11.26}$$
$$\mu_{21} = m_{21} - \overline{x} m_{11} - \overline{y} m_{20} + 2\overline{x}^2 m_{01} \tag{11.27}$$
$$\mu_{12} = m_{12} - \overline{y} m_{11} - \overline{x} m_{02} + 2\overline{y}^2 m_{10} \tag{11.28}$$
$$\mu_{30} = m_{30} - 3\overline{x} m_{20} + 2\overline{x}^2 m_{10} \tag{11.29}$$
$$\mu_{03} = m_{03} - 3\overline{y} m_{02} + 2\overline{y}^2 m_{01} \tag{11.30}$$

By normalising the central moments with the γth power of the zeroth central moment μ_{00} scale invariance is achieved. The $(p+q)$th order normalised central moment ν_{pq} is defined as

$$\nu_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{11.31}$$

where $\gamma = \frac{p+q}{2} + 1$ with $p, q \in \mathbb{N}_0$ and $p + q \geq 2$.

The Hu invariant moments are calculated from normalised central moments up to order three. H_n is the nth Hu invariant moment. Utilising the second and third order normalised central moments the following six absolute orthogonal moments can be derived:

$$
\begin{aligned}
H_1 &= \nu_{20} + \nu_{02} && (11.32)\\
H_2 &= (\nu_{20} - \nu_{02})^2 + 4\nu 11^2 && (11.33)\\
H_3 &= (\nu_{30} - 3\nu_{12})^2 + (3\nu_{21} - \nu_{03})^2 && (11.34)\\
H_4 &= (\nu_{30} + \nu_{12})^2 + (\nu_{21} + \nu_{03})^2 && (11.35)\\
H_5 &= (\nu_{30} - 3\nu_{12})(\nu_{30} + \nu_{12})[(\nu_{30} + \nu_{12})^2 - 3(\nu_{21} + \nu_{03})^2] + && (11.36)\\
&\quad (3\nu_{21} - \nu_{03})(\nu_{21} + \nu_{03})[3(\nu_{30} + \nu_{12})^2 - (\nu_{21} + \nu_{03})^2]\\
H_6 &= (\nu_{20} - \nu_{02})[(\nu_{30} + \nu_{12})^2 - (\nu_{21} + \nu_{03})^2] + && (11.37)\\
&\quad 4\nu_{11}(\nu_{30} + \nu_{12})(\nu_{21} + \nu_{03})
\end{aligned}
$$

One additional skew orthogonal moment can be calculated:

$$
\begin{aligned}
H_7 &= (3\nu_{21} - \nu_{03})(\nu_{30} + \nu_{12})[(\nu_{30} + \nu_{12})^2 - 3(\nu_{21} + \nu_{03})^2] -\\
&\quad (\nu_{30} - 3\nu_{12})(\nu_{21} + \nu_{03})[3(\nu_{30} + \nu_{12})^2 - (\nu_{21} + \nu_{03})^2]
\end{aligned}
\qquad (11.38)
$$

11.7 Mean Colour Information

A simple feature representing the object colour are the mean colour values of the HSV representation of the detected object of interest.

To determine the mean colour values the camera image $I(x, y)$ is converted from RGB colour space to HSV colour space [80]. The different channels of the HSV colour space represent hue, saturation and value respectively.

For each of the three colour channels the mean value of the localised object within the region of interest is calculated. Therfore the colour values of the pixels (x, y) that belong to the object, i.e. $B(x, y) = 1$, are averaged for each colour channel separately.

$$\overline{h} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I_H(x, y) B(x, y) \qquad (11.39)$$

$$\overline{s} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I_S(x, y) B(x, y) \qquad (11.40)$$

$$\overline{v} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I_V(x,y) B(x,y) \tag{11.41}$$

The resulting feature vector is a three-dimensional vector generated by concatenating the mean values of the three colour channels. Additionally only the mean value of the hue channel is used as a one-dimensional feature.

Colour information is helpful to distinguish e.g. between green and red apple. Advantages of colour information are its scale and rotation invariance as well as its robustness to partial occlusion. Furthermore colour information can be effectively calculated.

11.8 Wavelets

Another complex feature type is calculated concatenating the mean coefficients of the two-dimensional wavelet decomposition. The wavelet transform is performed on grey-level images. If the original images are colour images first a transformation from colour to grey scale is performed.

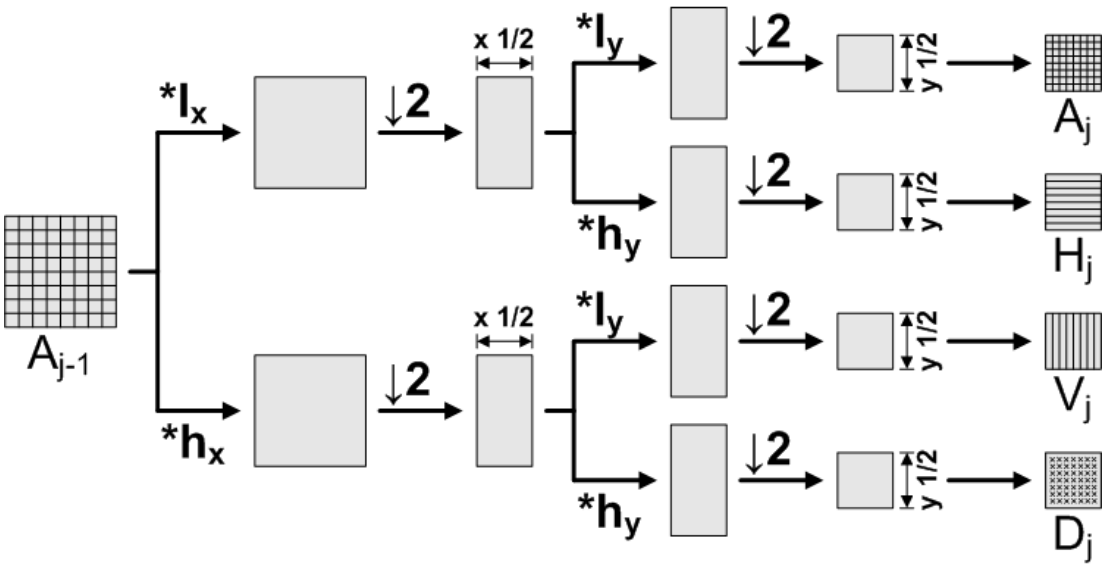

Figure 11.4: *Wavelet decomposition.*

A wavelet decomposition can be described by a sequence of separate convolutions and dyadic down-sampling. Figure 11.4 gives an overview of the different convolution and scaling operations performed in one decomposition step. Starting from the original grey-scale image $I(x,y)$ of size $M \times N$ these operations are

successively applied to decompose the image into different components. In each step j of the wavelet decomposition four equally sized output images of half of the width and half of the height of the input image are obtained. A complete decomposition requires $j_{max} = \log_2 \min(M, N)$ steps. The four images obtained in each step j show different properties. The approximation image A_j is a smoothed down scaled version of the input image A_{j-1}. It is the result of vertical and horizontal lowpass filtering of the input image and thus contains the low-frequency components. The approximation image A_j is the input image for the next step $j + 1$. The original image I is input image A_0 for level $j = 1$ The output image containing the horizontal details H_j is obtained by vertical highpass filtering and horizontal lowpass filtering the input image. It contains the high-frquency components in x-direction. When vertical lowpass filtering and horizontal highpass filtering the input image the resulting output image V_j contains the vertical details, i.e. the high-frequency components in y-direction. An image containing the diagonal details D_j is obtained by vertical and horizontal highpass filtering of the input image. It contains the high-frequency components in xy-direction.

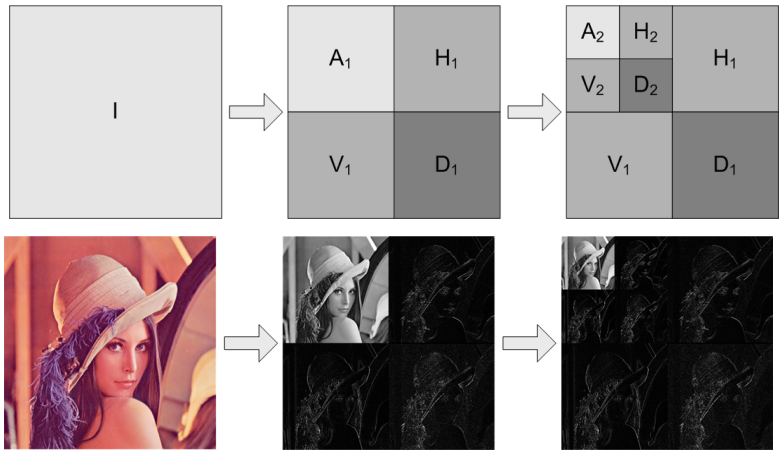

Figure 11.5: *Wavelet decomposition example.*

Figure 11.5 depicts the successive decomposition of the input image I into the four lower resolution output images A_j, H_j, V_j and D_j at a time. The decomposition is shown for level $j = 1$ and $j = 2$.

$$l = \frac{1}{\sqrt{2}} \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} \qquad h = \frac{1}{\sqrt{2}} \begin{array}{|c|c|} \hline -1 & 1 \\ \hline \end{array}$$

Figure 11.6: *Haar lowpass and highpass filter masks.*

The highpass and lowpass filters h and l used depend on the wavelet basis used. Example for common wavelet bases are the Haar wavelet and Daubechies wavelet. Figure 11.6 shows the highpass and lowpass filter mask based on the Haar wavelet.

The wavelet features are calculated on $m \times m$ potentially overlapping subimages.

11.9 Discussion

The different feature types described above vary in extraction effort and their suitability for the classification of three-dimensional objects. Figure 11.7 compares the different feature types and arranges them with regards to their complexity and their classification power on the data sets used in the context of this thesis.

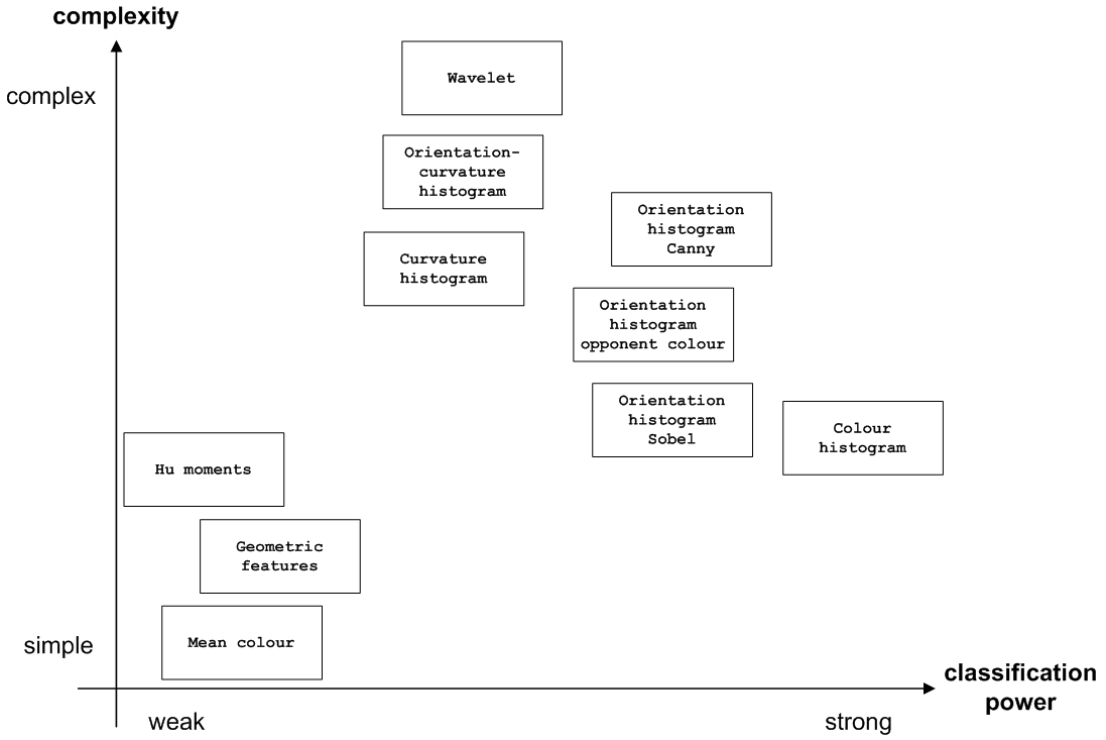

Figure 11.7: *Comparison of the different features types with regards to complexity and classification power evaluated on the data used within the scope of this thesis.*

The simple features show rather weak classification power, whereas the classification power of the complex features is generally rather good.

12 Statistical Evaluation

In this chapter the statistical experiments conducted to evaluate the proposed approach are described. The experiments examine the different aspects of the hierarchical neural network classifiers such as the classification performance of classifier hierarchies compared to non-hierarchical classifiers, the performance of the different strategies to combine the individual classifier outputs to a collective result, the capability of learning new objects during run time, the quality of the codes generated from the hierarchy activation and the suitability of different feature types for the classification of three-dimensional objects as well as their applicability in classifier hierarchies.

The experiments are conducted utilising 10-times 10-fold cross-validation. When comparing different approaches each approach has been evaluated using the same partition of the data in order to allow for paired significance testing. Both parametric and non-parametric tests were used as significance tests. The tests used are the corrected repeated k-fold cross validation t-test, the sign test, the Wilcoxon signed rank test and the maximum test.

The classification performance of the different classification approaches is expressed by means of mean error rates which are determined utilising 10-times 10-fold cross-validation experiments. The results are depicted by means of boxplots and error-bars.

12.1 Hierarchical Neural Networks

In order to benchmark the classification performance of hierarchical classifiers they are directly compared to non-hierarchical classifiers. In the experiments should be verified whether and in which cases hierarchical classifiers show improved classification performance compared to non-hierarchical classifiers. Therefore different classifiers types were used in their original non-hierarchical form as well as in a hierarchical arrangement. The classifier types (see chapter 2) used are

RBF networks, k-NN classifiers with $k = 3$ and $k = 5$ and fuzzy k-NN classifiers with $k = 3$ and $k = 5$. The experiments were conducted using different data sets. The data sets are from various domains, such as optical character recognition and three-dimensional object recognition. The data sets (see chapter 10) used are the fruits data set and the COIL-20 data set where orientation histograms with $m = 3$ and $b = 8$ were used as feature type (see chapter 11).

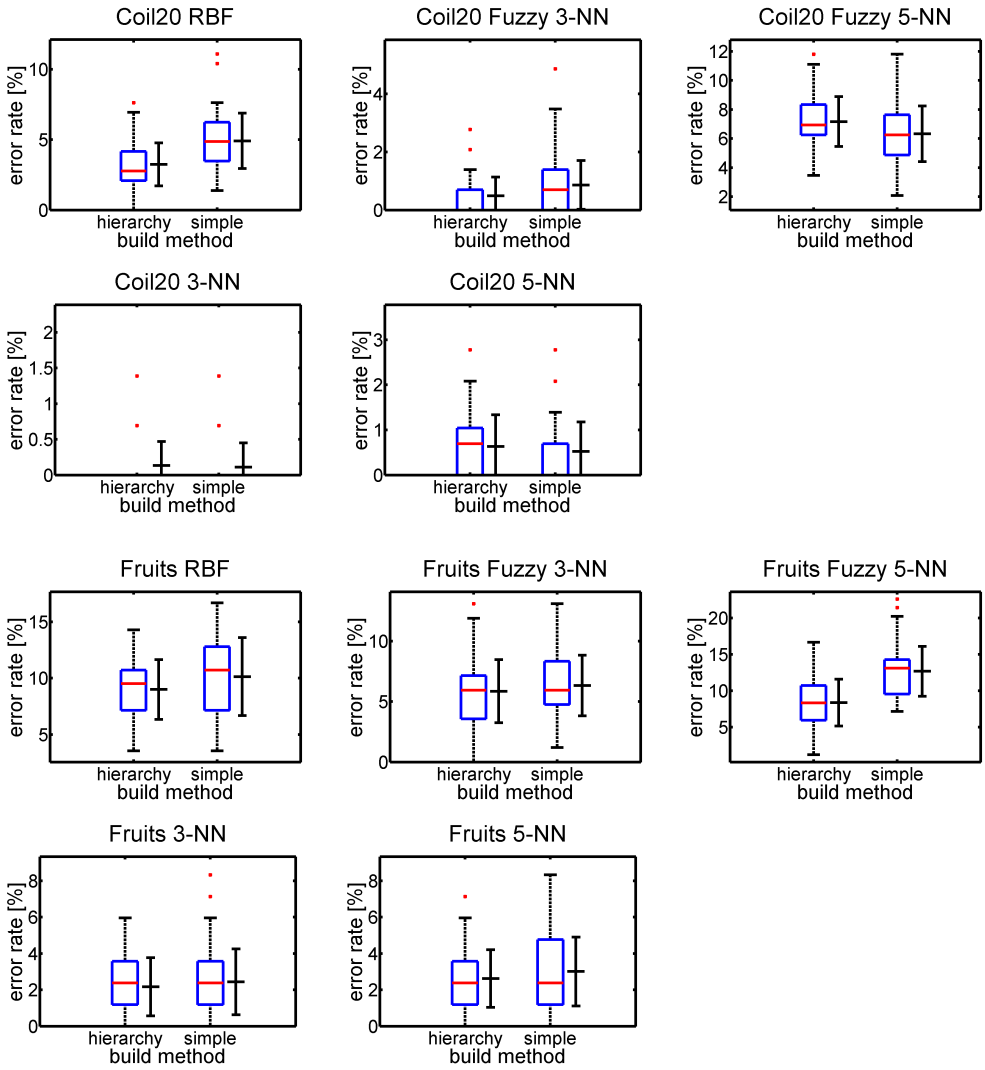

Figure 12.1: *Mean error rates for the two data sets (COIL-20, fruits) on the test data for hierarchical and non-hierarchical classifiers. As classifier RBF networks, fuzzy k-NN classifiers and k-NN classifiers were used. The box plots as well as the error bars indicate that hierarchical classifiers show lower error rates or at least the same error rates as the non-hierarchical approaches on both data sets.*

In order to show the universality of the presented approach different classifier types and data sets are used in the experiments. The experiments verify that

hierarchical neural network classifiers are applicable to various problem domains as well as to different types of classifiers.

Data	RBF		Fuzzy 3-NN		Fuzzy 5-NN		3-NN		5-NN	
	H	NH	H	NH	H	NH	H	NH	H	NH
COIL-20	**3.24**± **1.52**%	4.91± 1.97%	**0.49**± **0.65**%	0.86± 0.84%	7.17± 1.71%	**6.33**± **1.91**%	0.13± 0.34%	**0.11**± **0.34**%	0.63± 0.71%	**0.52**± **0.66**%
Fruits	**9.17**± **2.60**%	9.77± 3.24%	**5.86**± **2.61**%	6.33± 2.50%	**8.36**± **3.22**%	12.67± 3.43%	**2.17**± **1.60**%	2.44± 1.82%	**2.62**± **1.59**%	3.01± 1.90%

Table 12.1: *Mean error rates for the different data sets on the test data for classifier hierarchies (H) and corresponding non-hierarchical classifiers (NH) for the radial basis function network (RBF), k-nearest neighbour classifier (k-NN) and fuzzy k-nearest neighbour classifier (fuzzy k-NN). The average classification rates of the hierarchical approach are mostly higher than or almost equal to the classification rates of the non-hierarchical classifiers.*

The classification results are depicted in figure 12.1 by means of box-plots and error-bars. The mean error rates are listed in table 12.1. The results of the significance test are specified in table 12.2. In most cases the hierarchical classifier showed significantly better classification performance compared to the non-hierarchical classifier or no significant difference could be observed.

Data	Significance test	RBF	Fuzzy 3-NN	Fuzzy 5-NN	3-NN	5-NN
COIL-20	t-test	4.06e-2	1.27e-1	2.19e-1	7.96e-1	4.01e-1
	sign test	1.60e-10	2.03e-6	2.64e-3	7.27e-1	8.13e-3
	signed rank test	6.63e-8	2.38e-6	3.45e-5	8.44e-1	6.90e-3
	maximum test	1.91e-6	3.91e-3	1.00e+0	1.00e+0	6.25e-2
Fruits	t-test	5.69e-1	7.14e-1	8.50e-3	7.50e-1	6.58e-1
	sign test	6.62e-2	3.48e-1	5.78e-11	8.26e-1	1.35e-1
	signed rank test	1.00e-1	2.24e-1	1.16e-12	3.35e-1	2.24e-1
	maximum test	3.13e-2	1.00e+0	1.19e-7	5.00e-1	6.25e-2

Table 12.2: *Results of the significance tests for the different data sets on the test data comparing classifier hierarchies and corresponding non-hierarchical classifiers for the radial basis function network (RBF), k-nearest neighbour classifier (k-NN) and fuzzy k-nearest neighbour classifier (fuzzy k-NN). The table gives the calculated p-values. If a significant difference at the significance level $\alpha = 5\%$ could be observed the corresponding p-values are coloured in light grey. The significance tests indicate that no significant differences between the classification results of the two different methods can be observed for most data sets.*

12.2 Hierarchy Evaluation

The following experiments assess the performance of the different fusion strategies developed within the context of this work. The Demster-Shafer approach and the decision tree inspired approach are the main focus of this evaluation.

12.2.1 Comparison of the Different Fusion Strategies

The comparison of the different strategies for fusing the results of the classifiers within the hierarchy is performed using different data sets and different classifier types. For each data set and classifier type a hierarchy is generated and trained and to this hierarchy the different fusion strategies are applied. This ensures the direct comparability of the results. As classifiers RBF networks and fuzzy k-NN classifiers with $k = 3$ and $k = 5$ were deployed. The usage of the latter is motivated by the simplicity and the low training effort of this approach as well as by the fact that no parameters except k need to be optimised for this type of classifier. The experiments were conducted on the fruits data set and the COIL-20 data set using orientation histograms with $m = 3$ and $b = 8$ as feature type. The fusion strategies examined are the evidence theoretic approach, the decision-tree like approach, the voting scheme approach, the end nodes approach and the approach utilising similarity preserving codes.

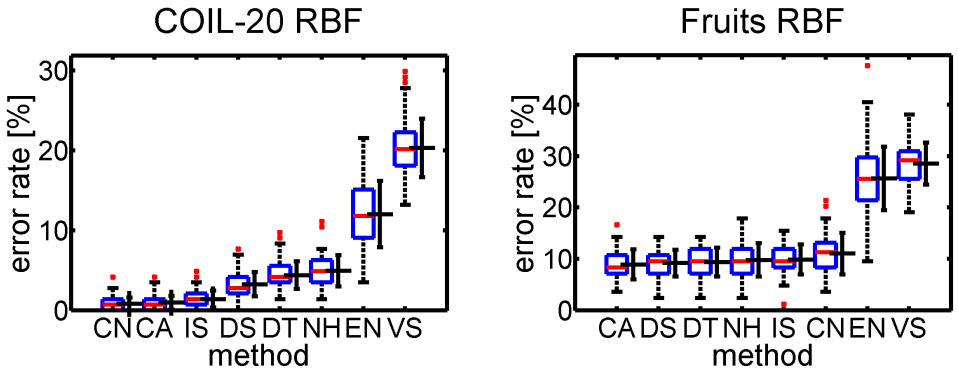

Figure 12.2: *Mean error rates for the COIL-20 data set and the fruits data set on the test data for the different retrieval strategies as well as for non-hierarchical classifiers. The strategies evaluated were the Dempster-Shafer method (DS), the decision tree method (DT), the voting scheme method (VS), the end node method (EN), the retrieval strategy utilising similarity preserving codes and associative memories (CA), the retrieval strategy utilising similarity preserving codes and non-hierarchical nearest-neighbour classifiers (CN), the inter-state decision template method (IS) and non-hierarchical classifiers (NH). As classifier radial basis function networks were used. The box plots as well as the error bars indicate that Dempster-Shafer method performs better than the other strategies.*

In the experiments should be ascertained which of the fusion strategies yields the best classification results and whether these results can be observed on all data sets and for all classifiers or whether certain strategies are suitable for specific classifiers and data sets whereas other classifiers yield better results on other data sets.

Data	DS	DT	CA	CN	IS	NH	EN	VS
COIL-20	3.24± 1.52%	4.38± 1.71%	0.93± 0.85%	**0.78**± **0.79**%	1.39± 1.04%	4.91± 1.97%	12.00± 4.14%	20.30± 3.65%
Fruits	9.17± 2.60%	9.35± 2.77%	**8.91**± **2.91**%	11.00± 4.05%	9.87± 2.90%	9.77± 3.24%	25.64± 6.21%	28.52± 4.08%

Table 12.3: *Mean error rates for the different data sets on the test data for the different retrieval strategies for the radial basis function network. The strategies evaluated were the Dempster-Shafer (DS) method, the decision tree method (DT), the voting scheme method (VS), the end node method (EN), the retrieval strategy utilising similarity preserving codes and associative memories (CA), the retrieval strategy utilising similarity preserving codes and non-hierarchical nearest-neighbour classifiers (CN), the interstate decision template method (IS) and the non-hierarchical classifiers (NH). The complex retrieval strategies show superior classification results in all experiments.*

The error rates for the different data sets are charted in figure 12.2 by means of box plots and error bars and are listed in table 12.3. The simple fusions strategies such as the voting scheme approach and the end node approach consistently showed weak performances. The more complex fusion strategies showed considerably stronger classification performances on the tested data sets. The decision tree-like approach yield good classification results in all experiments but was always outperformed by the evidence theoretic approach. The fusion strategy utilising similarity preserving codes and associative memories showed excellent performance that even exceeded the classification performance of the evidence theoretic approach on the data sets tested, but required exhaustive adjustment of the parameters to the given problem.

12.2.2 Comparison of the Evidence-Theoretic Fusion Strategy Against the Decision Tree Like Fusion Strategy

Another experiment was conducted evaluating the fusion strategy utilising the Dempster-Shafer theory of evidence and fusion strategy analogous to decision trees. The experiments were performed such that both fusion strategies were applied to the same trained classifier hierarchy in order to ensure maximal commensurability. Different data sets and classifiers were exploited in the experiments. As data sets the COIL-20 data set, the STATLOG digits data set and the letter-

recognition data set were utilised. The classifiers used were RBF networks and fuzzy k-NN classifiers with $k = 3$ and $k = 5$.

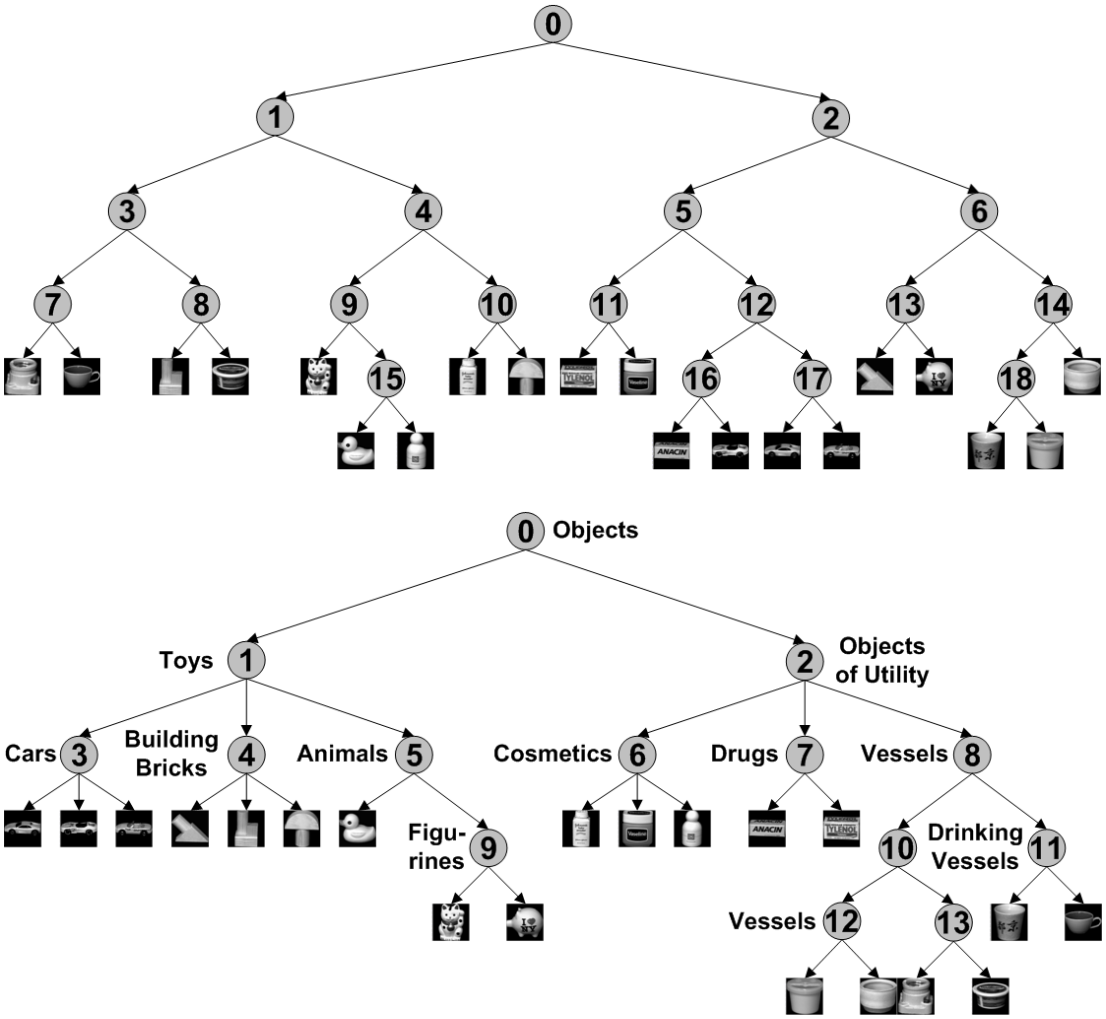

Figure 12.3: *Hierarchies for the classification of the COIL-20 objects. The upper hierarchy was automatically generated by unsupervised k-means clustering, the lower hierarchy was manually created grouping objects in a plausible way such that meaningful groups result.*

The approach was not only evaluated on automatically generated hierarchies but also on hierarchies that were manually created grouping classes in a plausible manner such that meaningful groups emerge and the classes within one group bear resemblance to each other in order to examine whether the proposed fusion strategies are also applicable to dictated hierarchies. Figure 12.3 depicts the two hierarchies for the COIL-20 data set generated in the described ways. The experiments clearly indicate the superiority of the evidence theoretic approach over the decision tree like approach.

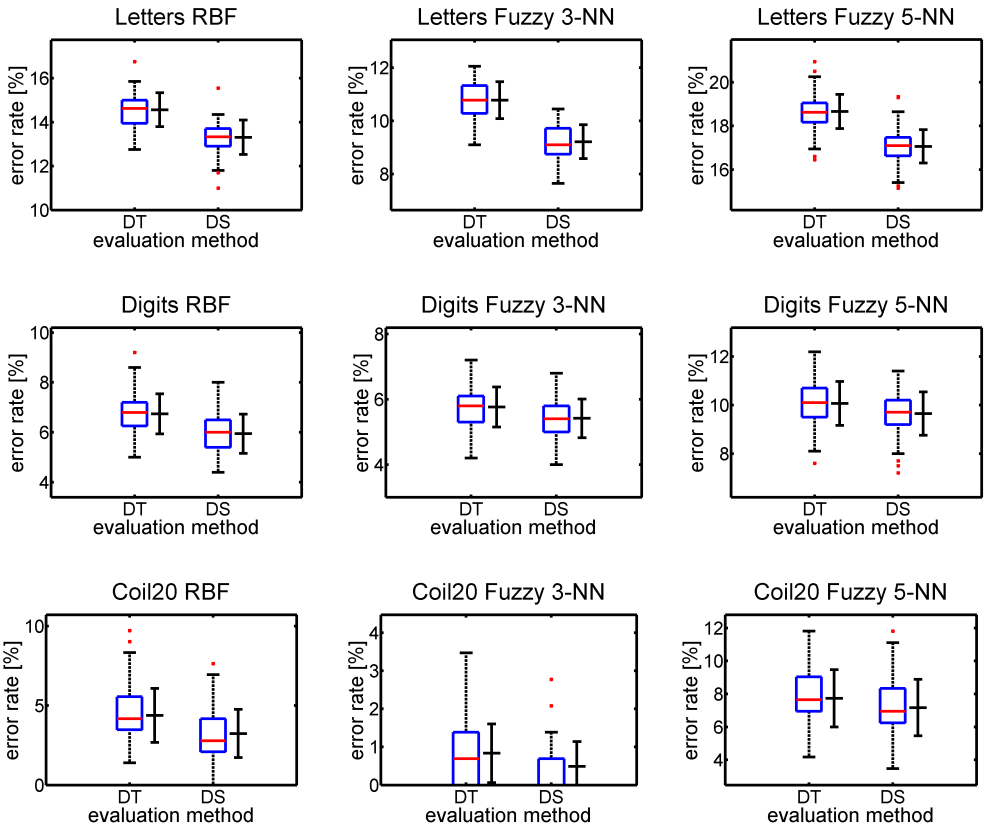

Figure 12.4: *Mean error rates for the three data sets (letters, digits, COIL-20) on the test data for the evidence based (DS) and the decision-tree-like (DT) approach on the automatically generated hierarchies. As classifier radial basis function networks were used. The box plots as well as the error bars indicate that Dempster-Shafer method performs better than the decision tree method on all three data sets.*

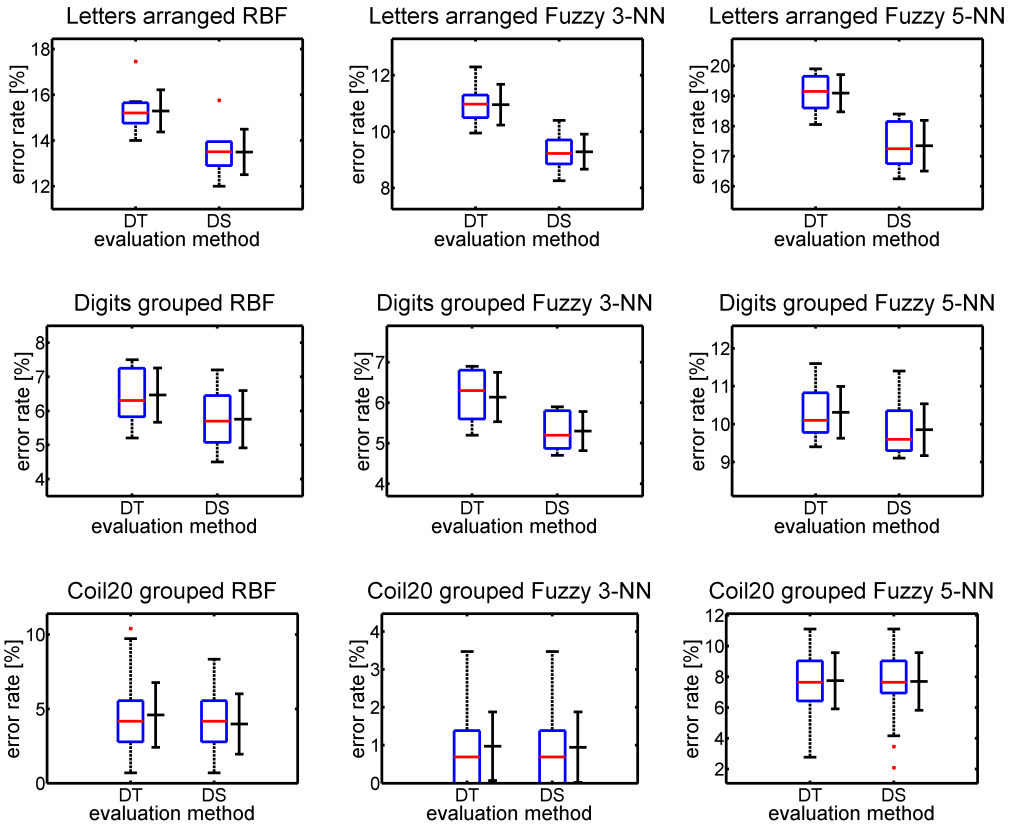

Figure 12.5: *Mean error rates for the three data sets (letters, digits, COIL-20) on the test data for the evidence based (DS) and the decision-tree-like (DT) approach on the manually generated hierarchies. As classifier radial basis function networks were used. The box plots as well as the error bars indicate that Dempster-Shafer method yields the same or even better classification rates than the decision tree method on all three data sets.*

Repeated k-fold cross validation experiments were conducted to assess the results of the different experiments statistically. The results for the automatically and manually generated hierarchies are visualised in figure 12.4 and 12.5 respectively by means of box plots and error bars. The tables 12.4 and 12.5 list the error rates for the different experiments performed on the automatically and manually generated hierarchies respectively. The results of the significance tests for the different experiments are listed in the tables 12.6 and 12.7. These tests imply that the classification results for the evidence theoretic approach are significantly better than the results for the decision-tree-like approach.

Data	RBF		Fuzzy 3-NN		Fuzzy 5-NN	
	DS	DT	DS	DT	DS	DT
Letters	**13.31** ± **0.78**%	14.56 ± 0.77%	**9.22** ± **0.64**%	10.78 ± 0.69%	**17.07** ± **0.76**%	18.66 ± 0.78%
Digits	**5.95** ± **0.79**%	6.74 ± 0.80%	**5.42** ± **0.60**%	5.76 ± 0.62%	**9.65** ± **0.90**%	10.07 ± 0.91%
COIL-20	**3.24** ± **1.52**%	4.38 ± 1.71%	**0.49** ± **0.65**%	0.83 ± 0.77%	**7.17** ± **1.71**%	7.74 ± 1.74%

Table 12.4: *Mean error rates for the different data sets on the test data for the Dempster-Shafer method (DS) and the decision tree method (DT) for the radial basis function network (RBF) and fuzzy k-nearest neighbour classifier (fuzzy k-NN) on automatically generated hierarchies. The evidence theoretic approach outperforms the decision tree approach in all experiments.*

Data	RBF		Fuzzy 3-NN		Fuzzy 5-NN	
	DS	DT	DS	DT	DS	DT
Letters	**13.49** ± **1.00**%	15.29 ± 0.93%	**9.29** ± **0.62**%	10.96 ± 0.72%	**17.35** ± **0.84**%	19.09 ± 0.61%
Digits	**5.75** ± **0.84**%	6.46 ± 0.80%	**5.30** ± **0.48**%	6.14 ± 0.61%	**9.85** ± **0.69**%	10.31 ± 0.68%
COIL-20	**3.99** ± **2.03**%	4.60 ± 2.18%	**0.95** ± **0.93**%	0.97 ± 0.90%	**7.70** ± **1.87**%	7.74 ± 1.83%

Table 12.5: *Mean error rates for the different data sets on the test data for the Dempster-Shafer method (DS) and the decision tree method (DT) for the radial basis function network (RBF) and fuzzy k-nearest neighbour classifier (k-NN) on manually generated hierarchies. The average error rates of the Dempster-Shafer approach are always lower than the error rates of the decision-tree method.*

A further experiment was conducted to emphasise the advantages of the evidence theoretic retrieval strategy. In this experiment the classification performance of a classification hierarchy where the classifier assigned to the root node shows a weak

Data	Significance test	RBF	Fuzzy 3-NN	Fuzzy 5-NN
Letters	t-test	4.68e-12	4.98e-29	5.21e-28
	sign test	4.46e-10	4.16e-23	4.16e-23
	signed rank test	3.50e-9	3.81e-18	3.86e-18
	maximum test	2.84e-14	1.58e-30	1.58e-30
Digits	t-test	3.84e-7	4.78e-6	1.11e-7
	sign test	1.40e-20	2.52e-29	6.31e-30
	signed rank test	1.28e-17	1.57e-17	7.61e-18
	maximum test	1.08e-19	2.52e-29	6.31e-30
COIL-20	t-test	2.58e-2	4.38e-2	9.59e-3
	sign test	5.80e-13	1.96e-11	6.94e-18
	signed rank test	3.95e-10	7.06e-8	2.44e-11
	maximum test	7.81e-3	1.53e-5	6.94e-18

Table 12.6: *Results of the significance tests for the different data sets on the test data comparing the Dempster-Shafer (DS) method and the decision tree method (DT) for the radial basis function network (RBF) and fuzzy k-nearest neighbour classifier (k-NN) on automatically generated hierarchies. The table gives the calculated p-values. If a significant difference at the significance level $\alpha = 5\%$ could be observed the corresponding p-values are coloured in light grey. The significance tests indicates that the evidence theoretic approach outperforms the decision tree approach significantly.*

Data	Significance test	RBF	Fuzzy 3-NN	Fuzzy 5-NN
Letters	t-test	3.78e-6	1.35e-7	8.97e-7
	sign test	1.95e-3	1.95e-3	1.95e-3
	signed rank test	1.95e-3	1.95e-3	1.95e-3
	maximum test	1.95e-3	1.95e-3	1.95e-3
Digits	t-test	4.27e-3	1.72e-5	1.41e-3
	sign test	2.38e-1	9.63e-2	2.38e-1
	signed rank test	7.32e-4	2.44e-4	4.88e-4
	maximum test	9.77e-4	2.44e-4	4.88e-4
COIL-20	t-test	2.12e-1	7.61e-1	8.55e-1
	sign test	3.32e-1	5.49e-1	8.56e-1
	signed rank test	5.18e-3	1.00e+0	4.71e-1
	maximum test	7.81e-3	1.00e+0	1.00e+0

Table 12.7: *Results of the significance tests for the different data sets on the test data comparing the Dempster-Shafer (DS) method and the decision tree method (DT) for the radial basis function network (RBF) and fuzzy k-nearest neighbour classifier (k-NN) on manually generated hierarchies. The table gives the calculated p-values. If a significant difference at the significance level $\alpha = 5\%$ could be observed the corresponding p-values are coloured in light grey. The significance tests indicate that the evidence theoretic approach shows significantly better classification performance or yields approximately the same classification results as the decision-tree approach.*

performance is looked at. Therefore two almost identical classifier hierarchies are compared where only the root node classifiers differ insofar that one classifier has a considerably lower classification performance.

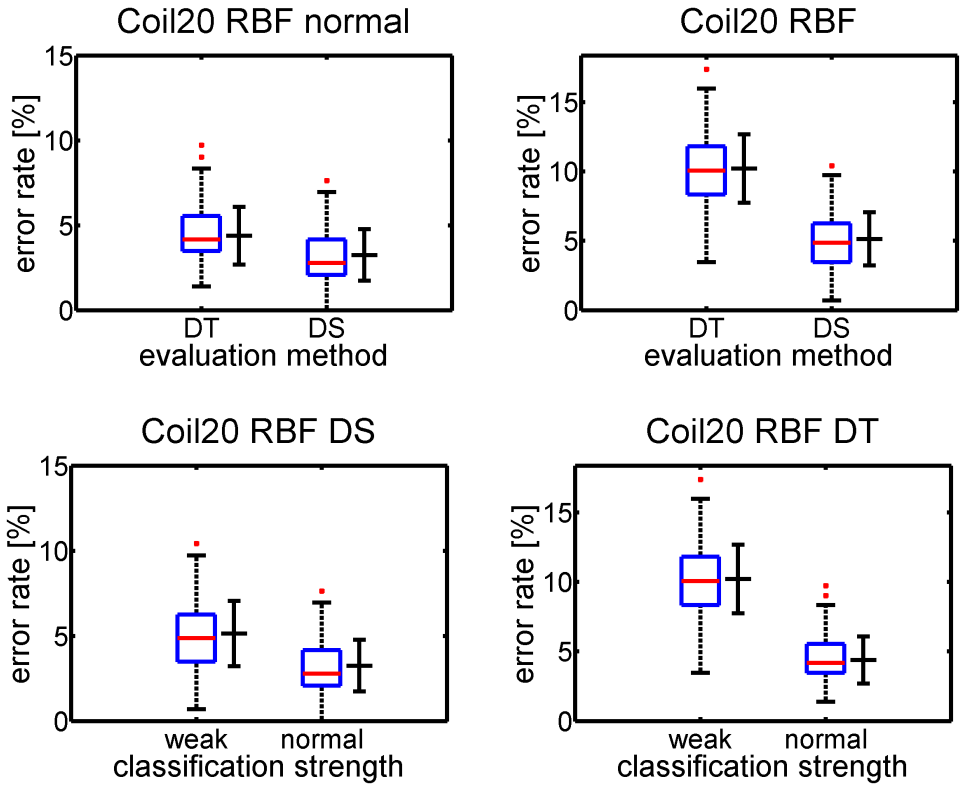

Figure 12.6: *Mean error rates for for the COIL-20 data sets on the test data comparing the Dempster-Shafer (DS) method and the decision tree method (DT) utilising RBF networks regarding their classification performance when having a weak classifier (weak) assigned to the root node of the hierarchy compared the a hierarchy with a normal classifier (normal) assigned to the root node.*

Data	Weak		Normal	
	DS	DT	DS	DT
COIL-20	$5.14 \pm 1.91\%$	$10.20 \pm 2.47\%$	$3.24 \pm 1.52\%$	$4.38 \pm 1.71\%$

Table 12.8: *Mean error rates on the test data comparing normally generated hierarchies (normal) and hierarchies with weak root nodes (weak) utilising RBF networks. The classification performance of the normal hierarchy is significantly higher than the classification performance of the weakened hierarchy for both retrieval strategies the evidence theoretic approach and the decision-tree approach. The evidence-theoretic approach however is much more robust against weak classifiers at high levels of the hierarchy.*

Figure 12.6 depicts the classification performance as error rates by means of box-

plots and error-bars. The mean classification errors as well as the results of the statistic significance tests are listed in the tables 12.8 and 12.9 respectively.

This experiment shows that when using the decision-tree approach the classification performance is highly degraded when the classifier at the root node shows a weak classification performance. The evidence theoretic approach is affected much less and clearly outperforms the decision tree approach.

Data	t-test	sign test	signed rank test	maximum test
COIL-20 normal DS vs DT	$2.58e-2$	$5.80e-13$	$3.95e-10$	$7.81e-3$
COIL-20 weak DS vs DT	$1.90e-8$	$3.16e-28$	$5.70e-18$	$1.26e-29$
COIL-20 DT normal vs weak	$1.39e-10$	$6.31e-30$	$8.07e-18$	$6.31e-30$
COIL-20 DS normal vs weak	$5.10e-4$	$5.78e-19$	$9.62e-16$	$1.06e-22$

Table 12.9: *Results of the significance tests for the COIL-20 data set on the test data comparing normally generated hierarchies and hierarchies with weak root nodes utilising RBF networks and the evidence theoretic retrieval strategy as well as the decision-tree retrieval strategy. The table gives the calculated p-values. If a significant difference at the significance level $\alpha = 5\%$ could be observed the corresponding p-values are coloured in light grey. The significance tests indicate that there are significant differences between the classification results when comparing the two different hierarchies as well as when comparing the two different retrieval strategies on both hierarchies.*

12.2.3 Evaluation of the Retrieval Strategy Utilising Similarity Preserving Sparse Codes

In order to evaluate the fusion strategy utilising similarity preserving sparse codes generated from the classifier hierarchies 10-times 10-fold cross-validation experiments have been performed on the fruits data set and on the COIL-20 data set. The classifier type used were RBF networks and orientation histograms with $m = 3$ and $b = 8$ were deployed as feature type.

The experiments should compare the classification results when utilising selected strategies for generating codes from the classifier hierarchy. The main focus of

the experiments was the comparison between codes generated from the activation of the hidden layer and codes generated from the activation of the output layer. As the different techniques for controlling the sparseness are equivalent they were not subject to the experiments.

Depending on the specific problems different learning strategies for the associative memories yield the best classification results. The dependencies of the input data is likely to be responsible for this behaviour as the input data are not independent, but so far the theory of associative memory only studied the case of independent input data. As this topic is out of scope of this thesis in the following the learning strategy that yield the best classification results has been chosen without further looking into the reasons for this behaviour. In the majority of cases the additive learning strategy that restrains the accumulation of the synaptic weights following the geometric series showed the best classification performance.

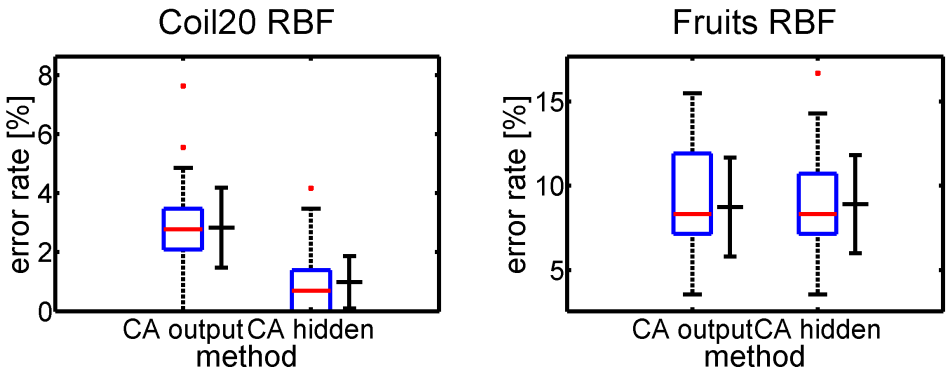

Figure 12.7: *Mean error rates for the COIL-20 data set and the fruits data set on the test data for the different inter-state decision template retrieval strategies. Two different variants of the similarity preserving sparse binary codes (CA) either use the activation of the hidden or the output layers of the neural classifiers within the hierarchy. As classifier radial basis function networks were used. The box plots as well as the error bars indicate that usage of the codes generated from the hidden layer yields considerably better results.*

The results of the experiments are depicted in figure 12.7 by means of box plots and error bars. The experiments show that the codes generated from the hidden activation yield a considerably higher classification performance or show at least the same classification performance. This is likely due to the fact that more information is contained in the activation of the hidden layer and that due to the higher number of neurons in the hidden layer the corresponding code is much more sparse.

12.2.4 Evaluation of the Inter-State Decision Template Approach

Within the context of this work two different types of inter-state decision templates as fusion strategies were developed. They differ only in the kind of neural activation they use. One variant utilises the activation of the neurons in the hidden layers of the neural classifiers and the other variant utilises the activation of the output layers.

The two variants were evaluated on two data sets, namely the fruits data set and the COIL-20 data set, by conducting 10-times 10-fold cross-validation experiments. As feature types orientation histograms (see chapter 11) with $m = 3$ and $b = 8$ were used.

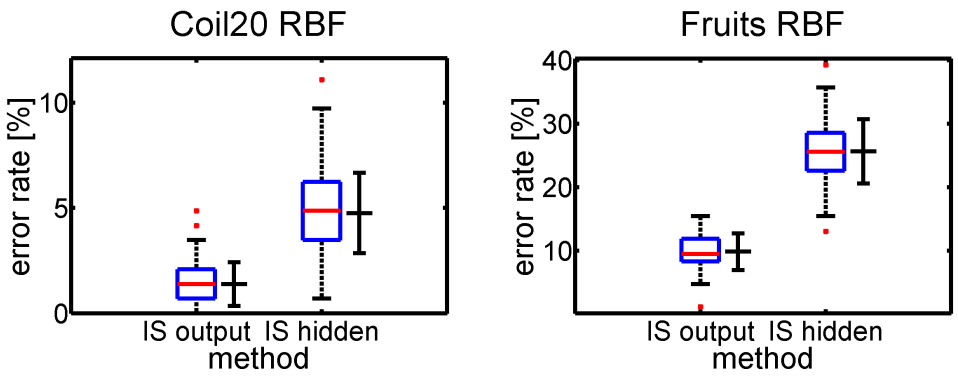

Figure 12.8: *Mean error rates for the COIL-20 data set and the fruits data set on the test data for the different inter-state decision template retrieval strategies. The two different variants of the inter-state decision templates (IS) either use the activation of the hidden or the output layers of the neural classifiers within the hierarchy. As classifier radial basis function networks were used. The box plots as well as the error bars indicate that usage of the activation of the output layer yields considerably better results.*

Figure 12.8 charts the classification results of the two variants by means of box plots and error bars. The variant utilising the activation of the output layers clearly outperforms the variant utilising the activation of the hidden layers on the tested data sets. The former variant even yields very good classification results.

12.3 Adaptive Incremental Learning of Novel Classes

By means of classification experiments the suitability of hierarchical neural networks for extension was examined. It could be verified that new classes which are only represented by a few samples can be learnt sufficiently well in moderate time and whether it is possible to learn new classes without negatively affecting the classification performance of already learnt classes. It is also looked at the positions were the newly learnt classes are inserted into the hierarchy. Are new classes added at arbitrary positions or can accumulations be observed? Another question to be evaluated is whether it is possible to incrementally learn the hierarchy from scratch while achieving sufficient classification quality. The differences between incrementally generated and hierarchies generated at a stroke are also looked at.

To evaluate these question the following experiments were conducted with RBF classifiers on the COIL-20 data set using orientation histograms with $m = 3$ and $b = 8$ as feature type.

12.3.1 Extension of Existing Hierarchies by Adaptive Incremental Learning

The first experiment added an unknown class to an already trained hierarchy. For this experiments 10 classes of the 20 classes of the COIL20 data set were chosen to represent the familiar objects. The remaining 10 classes formed a pool of potentially new objects. The first 10 classes were used to generate and train hierarchies in a 10-times 10-fold cross-validation experiment. To train the hierarchy sophisticated learning algorithms were used. Here the three-phase learning for RBF networks was used. The hierarchy structure was held constant for reasons of comparability while the hierarchy training was subject to cross-validation. This means the same hierarchy was trained 10×10 times resulting in 100 trained hierarchies. These trained hierarchies for the classification of the first 10 classes formed the basis for the incremental learning experiments. To each of these hierarchies one of the remaining 10 classes is added utilising the proposed incremental learning approach resulting in $10 \times 10 \times 10$ hierarchies each classifying 11 classes. The number of samples for the added classes is considerably lower than the number of the classes used to generate and train the hierarchy in the first instance. For the incremental learning 10 samples of the unknown class were used compared to 64 to 65 samples (depending on the specific cross-validation run) of the other classes. In the test data set all classes are represented with the same number of samples.

In these experiments the position within the hierarchy to which the respective new class was added is looked at. In order to validate whether the previously learnt classes were negatively affected, the classification accuracies of the individual classes in the hierarchies trained with 10 and with 11 classes are compared.

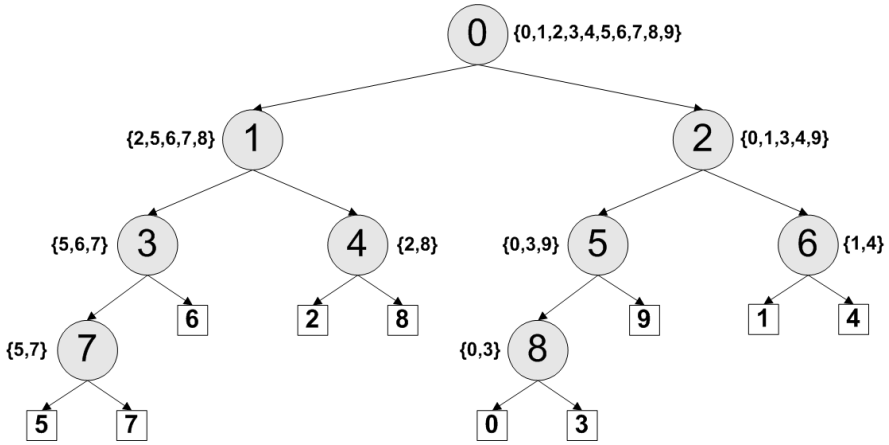

Figure 12.9: *Classifier hierarchy generated for the classification of 10 classes of the COIL20 data set using orientation histograms as feature type. Each node within the hierarchy represents a neural network which is used as a classifier. The end nodes represent classes. To each node a feature type and a set of classes is assigned. The corresponding neural network uses the assigned feature type to discriminate between the assigned classes. The highlighted path shows the nodes activated during the classification of a sample that is classified as class 8.*

The overall classification results of this approach were compared to the results achieved with a alternative approach for incrementally learning novel classes during run-time. This approach requires a complete rebuild of the hierarchy. The hierarchy is generated and trained using the samples of the old classes and the new class coinstantaneously. Thus each hierarchy is trained with 11 classes also resulting in $10 \times 10 \times 10$ hierarchies. For the cross-validation experiments the same partition was used as for the experiment evaluating the incremental learning approach. As the time is a crucial factor only fast learning algorithms for the training of the individual classifiers such as the two-phase learning scheme can be applied.

Despite the usage of a fast training procedure this approach shows a higher computation time than the incremental learning approach as the hierarchy needs to be generated and all classifiers have to be trained. The incremental learning approach leaves the majority of the classifiers unchanged and only a few classifiers need to be adjusted. The advantage of the rebuilding approach is that all classes can be considered when building the hierarchy structure and thus an optimal structure can be determined, but the higher computation time as well as the simple training algorithm are disadvantageous. In contrast the incremental

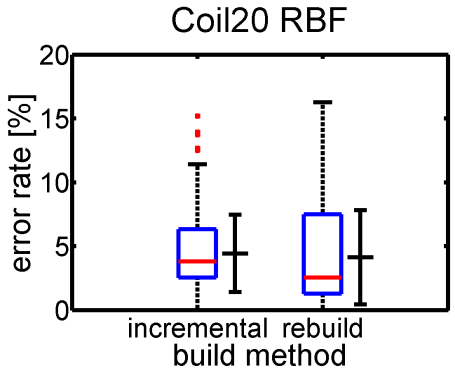

Figure 12.10: *Error rates for the incremental learning of novel classes.*

learning approach shows the advantages that a large part of the classifiers, that are trained using a sophisticated training scheme such as three-phase learning, remains unchanged and that only a few classifiers have to be retrained resulting in faster computation time. A drawback is the fact that contingently not the optimal hierarchy structure is identified as no restructuring of the hierarchy is allowed for.

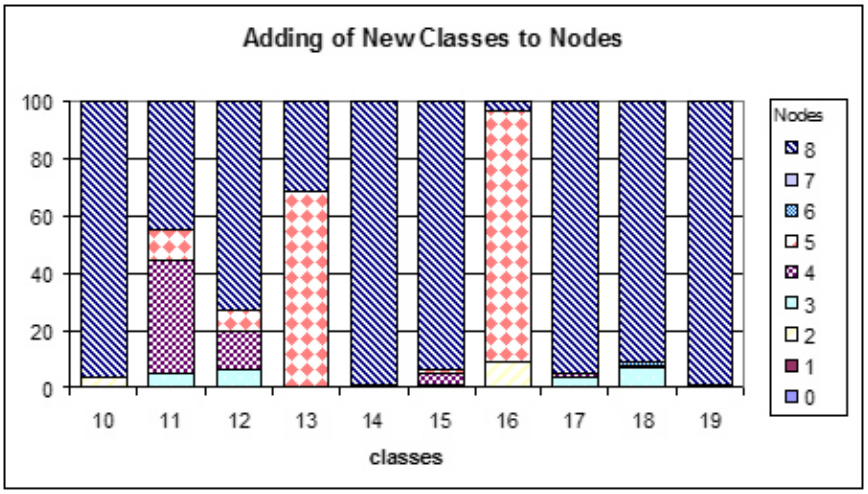

Figure 12.11: *Positions of the added novel classes. For each of the* 10×10 *cross-validation experiments conducted for each of the new classes 10 to 19 a leaf was added to different nodes of the classification hierarchy. For each new class is shown in percent to which node the corresponding leaf was added.*

Both approaches showed essentially the same classification quality. Despite being less extensive the incremental learning approach achieved the same results. Figure 12.10 visualises these results as box plots. Table 12.10 gives the mean error rates for both approaches utilising the decision-tree and the evidence theoretic retrieval strategy. The results of the statistical significance tests are listed in table 12.11.

The results of the significance tests indicate that there is no significant difference between the classification results of the two approaches.

Data	Incremental	Rebuild
COIL-20	$4.43 \pm 3.02\%$	$4.13 \pm 3.68\%$

Table 12.10: *Mean error rates on the test data comparing the incremental learning approach and the rebuild approach for adding one class utilising RBF networks. Both approaches yield approximately the same error rates.*

The confusion matrix for the incremental learning experiments displayed in figure 12.12 shows that although being represented by a significantly lower number of samples the classification rates of the new classes is equal to the classification rates of the primarily learnt classes.

Data	t-test	sign test	signed rank test	maximum test
COIL-20	0.8689	$3.11e-6$	0.2795	0.0625

Table 12.11: *Results of the significance tests on the test data for comparing the incremental learning approach and the rebuild approach for adding one class utilising RBF networks and the evidence theoretic retrieval strategy. The table gives the calculated p-values. If a significant difference at the significance level $\alpha = 5\%$ could be observed the corresponding p-values are coloured in light grey. The significance tests indicate that no significant differences between the classification results of the two different approaches can be observed. Both approaches show essentially the same performance.*

Having a look at the positions within the hierarchy were the new classes were added, it could be found that in the most instances a class was mainly assigned to one node. If this is not the case then at least the class was added to nodes lying on the same path of the hierarchy as can be observed for class 13. The only exception to this is class 11, which is distributed over different pathes of the hierarchy. This might result from deficient resemblance between class 11 and the 10 classes used to train the hierarchy. Figure 12.11 illustrates for each new class how often it was added to which node of the hierarchy shown in figure 12.9.

These results show that the new classes are not added arbitrarily to the hierarchy but for each class preferred positions emerge. The noticeable frequency of node 8 is likely to be a characteristic of the COIL20 data set.

12.3.2 Incrementally Building Classifier Hierarchies

In a second experiment the question regarding the ability to incrementally build hierarchies from scratch was addressed. For this experiment each of the 20 classes

	0	1	2	3	4	5	6	7	8	9	new
0	87,42	0	0	0,278	0	0	0	0	0	0,014	12,29
1	0	94,56	0	0,111	0	0	0	0	0	0	5,333
2	0	0	96,83	0,083	0	0	0	0	0	0	3,083
3	0	0,014	0	97,06	0	0	0	0	0	0	2,931
4	0	0	0	0	100	0	0	0	0	0	0
5	0	0,306	0	0,097	0	97,19	0	0	0	0	2,403
6	0	0	0	0,347	0	0	98,99	0	0	0	0,667
7	0	0	0	0	0	0	0	99,86	0	0	0,139
8	0	0	0	0	0	0	0	0	95,17	0	4,833
9	0	0	0	0,083	0	0	0	0	0	91,93	7,986
new	0	0,917	0	0,667	0	0	0	0	0,667	1,375	96,38

Figure 12.12: *Confusion matrix for the experiments utilising incremental learning.*

was consecutively added to the hierarchy using the incremental learning approach of adding new nodes. The classification accuracies of the resulting hierarchies was compared to the classification results of hierarchies generated and trained with all 20 classes at once and using a fast learning algorithm. As the hierarchy structure generated by means of the incremental learning approach depends on the order the classes added, the experiments were conducted with several different orders of the classes. Figure 12.13 charts the mean error rates by means of box plots and error bars. Table 12.12 lists the mean error rates and table 12.13 gives the results of the significance tests.

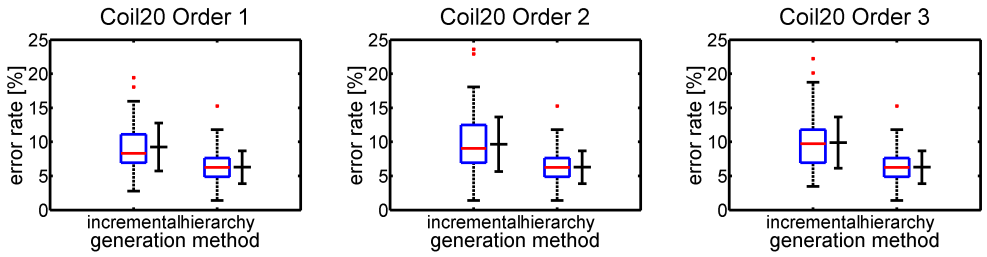

Figure 12.13: *Mean error rates for the incremental learning of novel classes from scratch.*

The rebuild approach shows better performances in all experiments, but the incremental approach still shows good classification results. The superiority of the rebuild approach is likely to be due to the fact that all information is available at once which facilitates the generation of a hierarchy optimally adapted to the data at hand. The hierarchy generated by the incremental approach is influenced by the order of the added classes and can only adapt the hierarchy to the sofar learnt data. Nevertheless it is possible to incrementally build classifier hierarchies with sufficient classification performance. When applying this approach in the robotic

field this is satisfactory to begin with. Nevertheless it is possible to retrain or even rebuilt the hierarchy in a longer intermission.

Data	Incremental		Rebuild	
	DS	DT	DS	DT
Order 1	$9.24 \pm 3.53\%$	$12.81 \pm 4.87\%$	$\mathbf{6.28 \pm 2.41\%}$	$\mathbf{7.80 \pm 2.58\%}$
Order 2	$9.65 \pm 4.00\%$	$11.11 \pm 3.63\%$	$\mathbf{6.28 \pm 2.41\%}$	$\mathbf{7.80 \pm 2.58\%}$
Order 3	$9.89 \pm 3.76\%$	$11.10 \pm 3.47\%$	$\mathbf{6.28 \pm 2.41\%}$	$\mathbf{7.80 \pm 2.58\%}$
Order 4	$10.04 \pm 3.84\%$	$12.78 \pm 4.31\%$	$\mathbf{6.28 \pm 2.41\%}$	$\mathbf{7.80 \pm 2.58\%}$

Table 12.12: *Mean error rates for the different class orders on the test data comparing the incremental learning approach and the rebuild approach utilising RBF networks. The average error rates of the rebuild approach are lower than the error rates of the incremental learning approach and do not depend on the order of the class items.*

Data	t-test	sign test	signed rank test	maximum test
Order 1	0.0248	$5.15e-9$	$2.08e-10$	$7.28e-12$
Order 2	0.0415	$1.18e-9$	$1.01e-9$	$2.38e-7$
Order 3	0.0079	$4.03e-12$	$2.95e-12$	$7.45e-9$
Order 4	0.0124	$7.38e-12$	$1.69e-12$	$3.05e-5$

Table 12.13: *Results of the significance tests for the different class orders on the test data comparing the incremental learning approach and the rebuild approach utilising RBF networks and the evidence theoretic retrieval strategy. The table gives the calculated p-values. If a significant difference at the significance level $\alpha = 5\%$ could be observed the corresponding p-values are coloured in light grey. The significance tests indicate that significant differences between the classification results of the two different approaches can be observed for all orders. The rebuild approach outperforms the incremental approach with respect to classification performance.*

12.4 Features for 3D-Object Recognition

The following section examines the suitability of different feature types (see chapter 11) for 3D-object recognition. The features were extracted from different data sets and were evaluated utilising non-hierarchical nearest-neighbour classifiers. The data sets used are the fruits data set, the COIL-20 data set and the COIL-100 data set (see chapter 10).

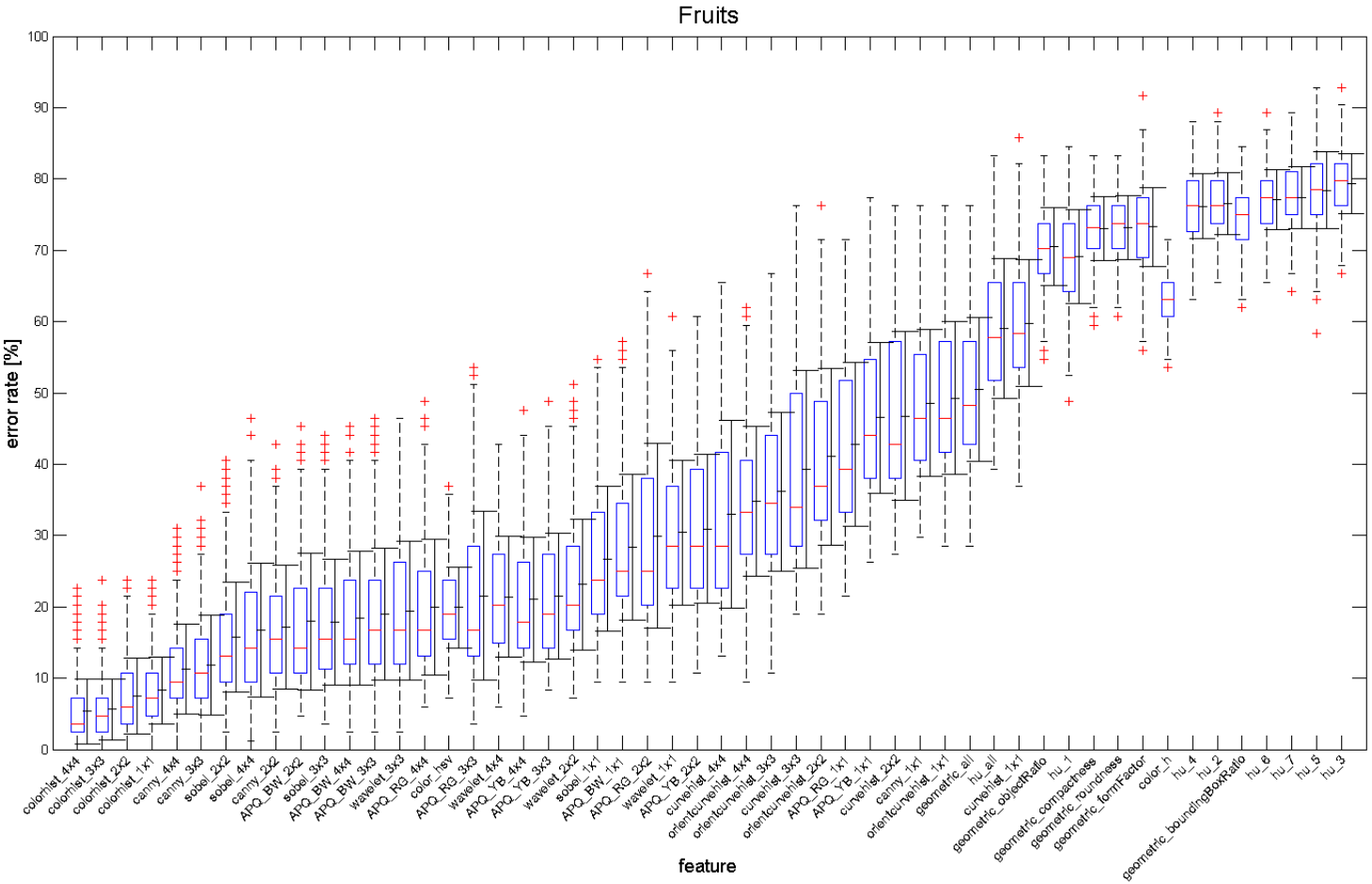

Figure 12.14: *Classification results for different features extracted from images of the fruits data set. The features were extracted from the previously identified regions of interest.*

Before extracting the features from the fruit data set the objects were localised within the image using a simple colour-based attention control mechanism [38]. From the so identified regions of interest which contained the fruits the features were extracted. On the COIL-20 data set also regions of interest were determined for the feature extraction. For the COIL-100 data set the features were extracted from the complete image, i.e. the regions of interest are identical to the image, as the objects almost fill the complete image.

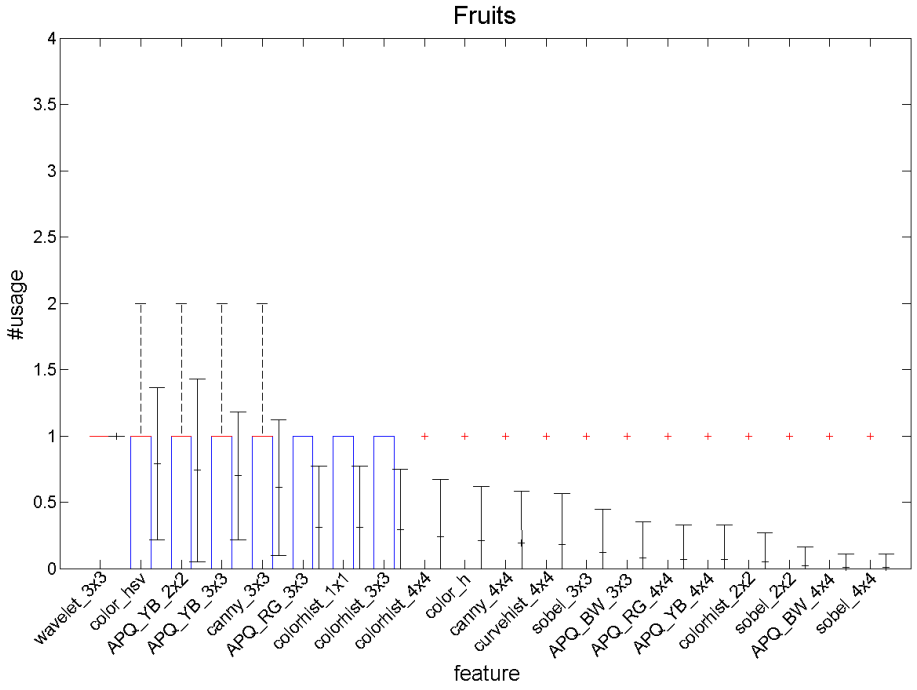

Figure 12.15: *Usage of the different features extracted from images of the fruits data set within classifier hierarchies. The features were extracted from the previously identified regions of interest.*

Not all feature types could be extracted from all data sets as e.g. feature types based on colour information are not very meaningful for grey scale images, for other features it is necessary to first localise the object of interest as the features are calculated from a binary image defining the object pixels. If applicable the features were extracted utilising a division of the regions of interest into overlapping subimages. Divisions into 1×1, 2×2, 3×3 and 4×4 subimages were used.

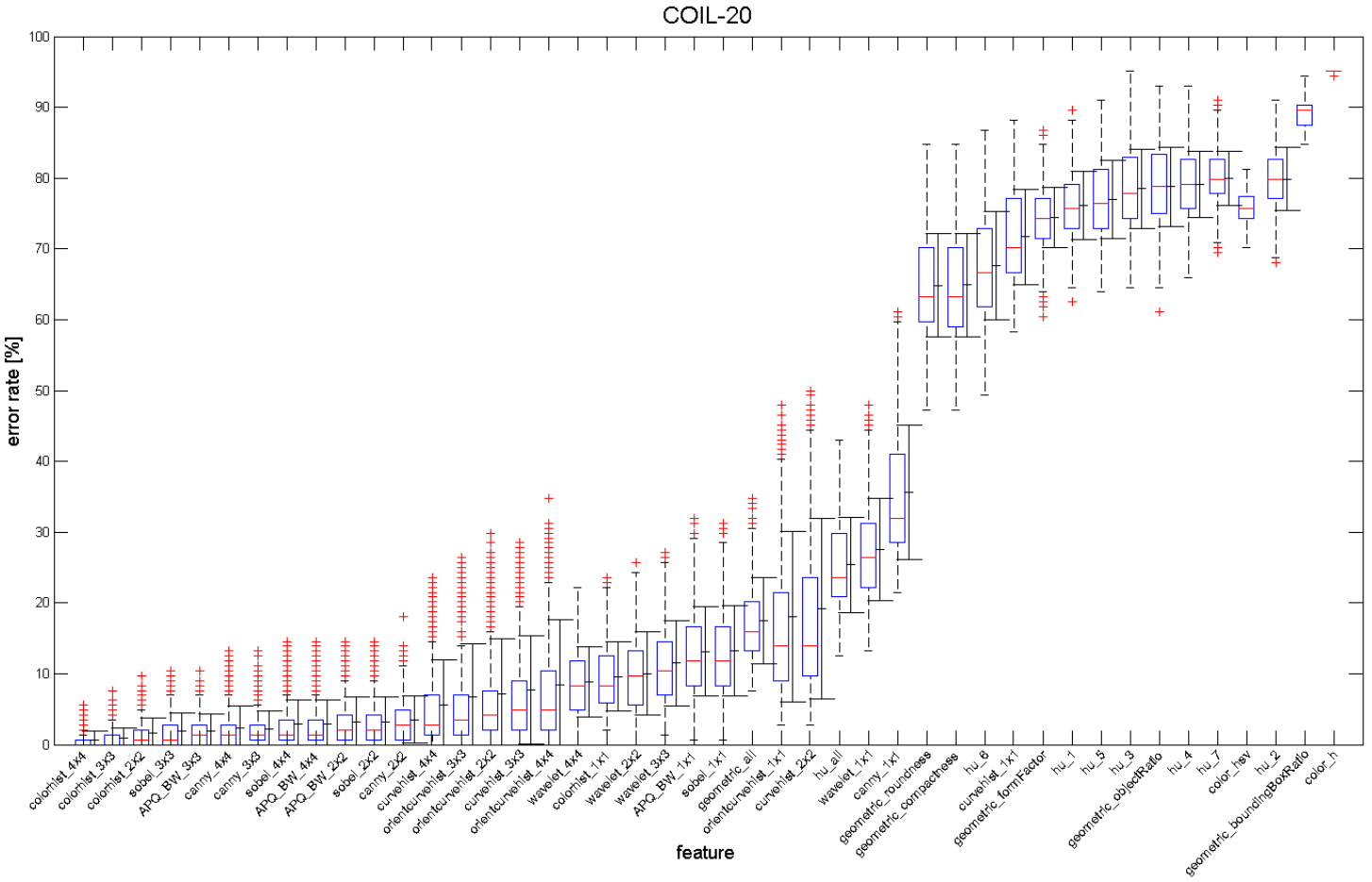

Figure 12.16: *Classification results for different features extracted from images of the COIL-20 data set. The features were extracted from the previously identified regions of interest.*

The different features were evaluated on the three data sets using k-nearest neighbour classifiers with $k = 1, 3, 5$ as well as fuzzy k-nearest neighbour classifiers with $k = 3, 5$. For the evaluation 10-times 10-fold cross-validation experiments were conducted with each classifier for each feature type on each data set. For each data set the results of the individual classifiers are used to rank the different feature types according to their respective classification accuracy. The ranks of the different data sets averaged over the individual classifier performances are then compared to each other in order to assess the different feature types and their suitability for the classification of three-dimensional objects.

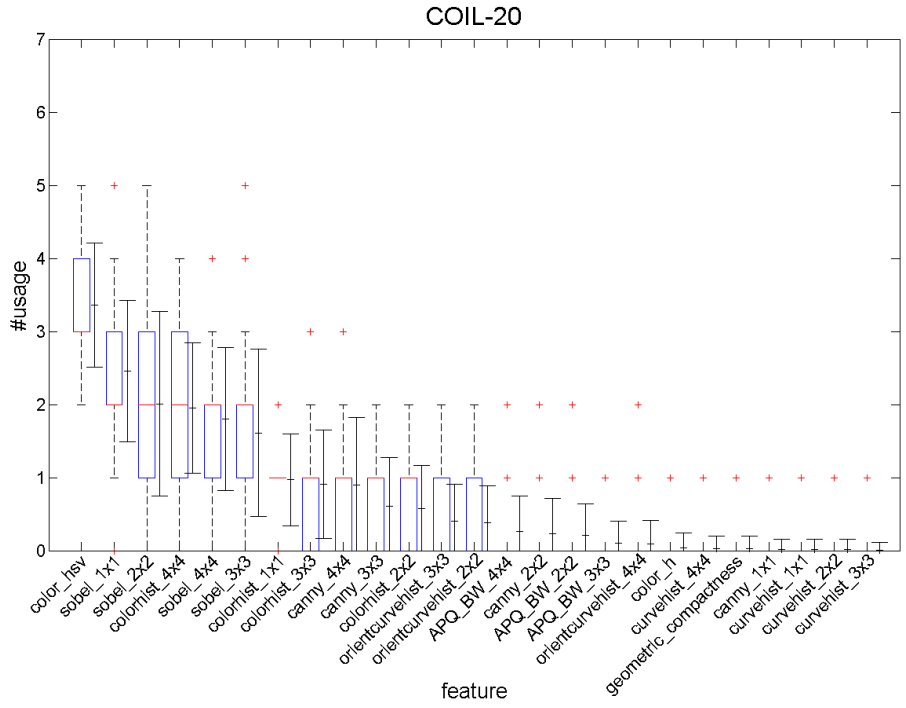

Figure 12.17: *Usage of the different features extracted from images of the COIL-20 data set within classifier hierarchies. The features were extracted from the previously identified regions of interest.*

The figures 12.14, 12.16 and 12.18 show the classification results for the individual feature types on the different data sets averaged over the different classifiers. The feature types are sorted according to their average classification performance and are ranked accordingly. For the detailed information on the error rates and ranks see appendix A.

Although the error rates for the different classifiers vary per feature type, the ranks, i.e. the relative order of the feature types with respect to the achieved clas-

sification performance, remain mainly constant per feature type. Similar rankings of the feature types can be observed across the different data sets.

The classification results indicate that the complex, high-dimensional feature types are suitable for the recognition of three-dimensional objects as represented in the test data sets since these feature types show continuously high classification performances in all experiments conducted. The colour histograms followed by the different types of orientation histograms showed the best classification results. Feature types also showing good classification results are wavelets, orientation-curvature histograms and curvature histograms. The classification results achieved with simple, low-dimensional feature types are considerably lower compared to the other feature types. These feature types are presumably not complex enough in order to facilitate the discrimination of the nontrivial objects the used data sets comprise.

The figures 12.15 and 12.17 show the frequency of usage of the different feature types within automatically generated hierarchies.

The feature types used were primarily complex feature types showing at least good classification results when tested separately. The simple feature types that are characterised by low classification performance were generally not used. The only exception is the mean colour information which was selected frequently and the geometric feature type compactness which was selected exceptionally. This result substantiate the appropriateness of the feature selection mechanism included in the hierarchy generation.

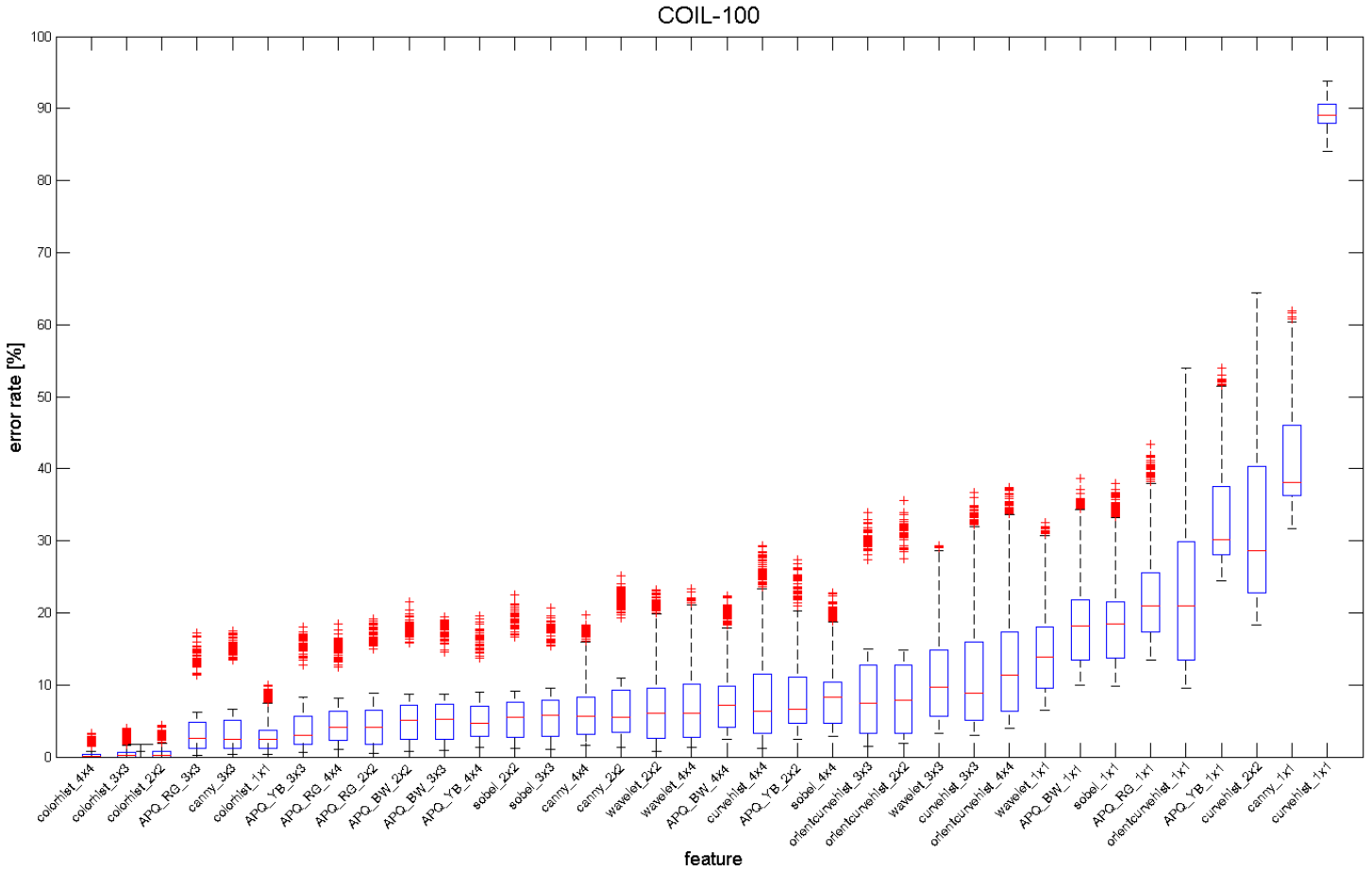

Figure 12.18: *Classification results for different features extracted from images of the COIL-100 data set. The features were extracted from the complete images.*

12.5 Discussion

Within the scope of this thesis several fusion strategies that yield considerable classification performance were developed. The most promising strategies were the evidence-theoretic approach, the fusion strategy analogous to decision trees, the inter-state decision templates and the approach utilising similarity preserving codes where the decision tree like retrieval strategy yields slightly lower classification results but is less expensive with regards to computation time. The results achieved by means of the simple voting strategy and the approach taking only the end nodes into account were considerably lower than the results obtained by the other fusion strategies.

When directly comparing the two approaches the evaluation of the classifier hierarchy by means of Dempster-Shafer evidence theory yields improved or at least the same classification results compared to the simple decision-tree-like evaluation method on all data sets used and with all classifier types deployed. Hierarchies automatically generated show more stable results than manually generated hierarchies, but the manually hierarchies also show good results. It could also be shown that the evidence-theoretic approach is more robust against weak classifiers at higher levels of the hierarchy.

It could be shown that the incremental learning of novel classes proved functional. If only a small number of new classes are added the performance of the incremental learning approach can be compared with the rebuild approach. If the complete hierarchy is built incrementally the rebuild approach outperforms the incremental approach as the availability of the complete data during the construction phase facilitates a more reasonable generation of the hierarchy.

The more complex feature types proved more suitable for the recognition of three-dimensional objects than the simple feature types. The colour histograms consistently showed the best classification results. The different types of orientation histograms also yield respectable results. These observations are also underpinned by the fact that during the hierarchy generation primarily the strong feature types are chosen and the weak feature types are scarcely selected.

Part IV: Discussion

This part summarises the main findings of this thesis, compares the proposed method to related approaches and discusses the developed approach.

13 Summary

Hierarchical neural network classifiers form the basis of the research presented in this work. Different aspects of hierarchical classifiers were evaluated. These aspects include the hierarchy generation, the selection of suitable features, the hierarchy training, the fusion of information within the hierarchy for retrieving the final classification result, the detection of outliers, the incremental extension of classifier hierarchies and the generation of similarity preserving sparse binary codes from the hierarchy. Another aspect investigated was the recognition of three-dimensional objects by means of hierarchical classifiers and the suitability of different feature types for this purpose.

A method for generating classifier hierarchies and selecting appropriate features utilising unsupervised clustering has been proposed within the scope of this work. The generation strategy yields balanced hierarchies that show good classification performance and exhibit a reasonable choice of features. The concurrent usage of several feature types within a classifier hierarchy is also a notable characteristic of the developed approach.

A main focus of the work at hand is the information fusion utilising the Dempster-Shafer theory of evidence. This strategy yields excellent classification results and continuously shows a higher classification performance than the simple strategies such as the decision tree like method, the voting scheme strategy and the approach only considering the end nodes. Moreover the availability of estimates for the class memberships of the individual classes and even of the possible sets of classes is a valuable by-product.

Two other very promising fusion strategies were developed within the scope of this thesis, namely the inter-state decision template method and an approach utilising similarity preserving codes generated from the classifier hierarchies. Both methods show a considerable classification performance.

Not only within this context, but also in connection with the integration of the proposed approach into a comprehensive cortical model consisting of associative memories the generation of similarity preserving sparse binary codes from hier-

archical neural network classifiers plays an important role. This thesis proposes several strategies to generate such codes which can be applied according to the specific requirements of the task at hand.

Another important aspect of hierarchical neural network classifiers examined in the context of this work is the adaptive incremental learning of novel classes. Two slightly different approaches for the incremental extension of classifier hierarchies have been proposed and successfully deployed. The approaches facilitate the learning of new objects during run-time in moderated time yielding appropriate classification results.

The general applicability of the approach could be shown by deploying different types of classifiers as well as evaluating the approach on data sets from different domains. Furthermore the approach has successfully been implemented on a mobile service robot. The hierarchies can be adjusted to a given classification task either by manually generating the hierarchy accordingly or by adapting the validation function underlying the hierarchy generation process.

As this thesis is concerned with the recognition of three-dimensional objects feature types suitable for this task are another noteworthy aspect of this work. Various feature types for representing three-dimensional objects have been examined in detail. The most promising feature types were colour histograms and orientation histograms.

14 Main Contributions

In this chapter the main contributions and findings of this thesis are specified following loosely the working hypotheses (see section 1.3) which formed the starting point of this work.

The contributions were no publications emerged from are either supplemental investigations or recent studies.

C1: Application of divide-and-conquer strategy to classifier fusion incorporating multiple feature types

The thesis introduces a new method for generating suitable classifier hierarchies employing unsupervised clustering methods. This method performs a coarse to fine output space decomposition with coinstantaneous feature selection [24]. The hierarchies generated utilise multiple feature types. The coinstantaneous usage of a large number of feature types separates the proposed approach from other methods. Moreover a strategy for the robust identification of unknown classes has been developed. Another notable characteristic is the universal applicability of the proposed approach with regards as well to the objects to be classified as well as the classifiers and feature types to be used. The classifier hierarchies show a number of benefits such as the availability of intermediate results, easy extensibility and enhanced controllability of complex classification tasks by subdividing the problem.

C2: Different Methods for the Evaluation of Hierarchical Neural Network Classifiers

The approach developed provides several strategies for the retrieval of the combined classification result out of the individual classifier results within the hierarchy. These strategies were developed, implemented and evaluated. The following

methods for fusing the classifier results were developed:

- Retrieval analogous to decision trees [24] [26] [27]
- Combining classifier results via Dempster-Shafer evidence theory [26] [27]
- Evaluating results of end node classifiers
- Using a simple voting scheme
- Classifying sparse binary codes generated from the classifier hierarchy by means of associative memories
- Inter-state decision templates

The evidence theoretic approach, the inter-state decision template method, the fusion strategy utilising sprase binary codes adn the decision tree like approach continuously showed good classification results whereas the classification results of the voting scheme and the end node evaluations were considerably weaker.

The usage of the evidence theory framework for fusing information within classifier hierarchies is a central point of this thesis and is an innovation compared to other approaches which only use the belief theory to fuse singe classifiers or classifiers that provide hierarchical output.

C3: Applying Dempster-Shafer Evidence Theory to Hierarchical Neural Network Classifiers

In the context of this thesis several theories for handling uncertainty have been examined, compared and assessed regarding their suitability when applied to hierarchical neural network classifiers. The belief theory was chosen above the other approaches as it offers the possibility to assign belief to hypotheses that are not part of the universe of discourse, i.e. it allows for dealing with unknown objects. Moreover within this framework it can be distinguished between uncertainty and ignorance and belief does not need to be particularised if no information is available for doing so.

The experiments performed within the scope of this work show that the Dempster-Shafer theory of evidence is suitable for the application to hierarchical neural classifier hierarchies. It is used to fuse the information provided by the individual classifiers to a common result. This fusion method does not only provide the resulting class but also an estimate of the class memberships of the samples to be classified. The classification performance of this evidence theoretic approach outperformed in all tested cases the performance of the straightforward approach of retrieving the classification results analogue to decision trees [26] [27].

C4: Adaptive Incremental Learning of Novel Classes

The thesis proposes two strategies for incrementally extending the classifier hierarchies. The new classes can either be added as new leaves or as new nodes. The latter approach allows for expanding the hierarchy structure, i.e. the hierarchy can evolve in depth, and thus facilitates the incremental generation of hierarchies. Furthermore a strategy for the incremental adaptive training of RBF networks has been developed. These strategies facilitate the learning of novel classes during run-time in adequate time and with sufficient classification quality without considerably negatively impacting the classification quality of the previously learnt classes [25]. Moreover hierarchies can completely be learnt incrementally.

The ability of learning new classes during run-time is an important capability for autonomous mobile robots employed in real-world environments. The functionality of the incremental learning strategy could not only be proved in statistical experiments but also by successfully implementing the proposed approach on a robot.

C5: Generation of Distributed Similarity Preserving Sparse Codes

The thesis introduces diverse concepts for generating distributed visual similarity preserving sparse codes from hierarchical neural network classifiers. These codes are either generated from the activation of the hidden layer of the individual classifiers within the hierarchy or from their output activation. Also different methods for controlling the sparseness of the code vectors have been employed. Moreover information provided by the classifiers such as classification quality or the overall classification result have been incorporated.

The so generated codes could successfully be employed for classification with associative memories or non-hierarchical nearest-neighbour classifiers.

C6: Evaluation of Different Features for 3-D Object Recognition

Within the scope of this thesis several feature types for the recognition of three-dimensional objects were either employed or developed and their suitability for the recognition of three-dimensional objects from two-dimensional camera images was investigated. The following feature types were used:

- Orientation histograms utilising the Sobel edge detector
- Orientation histograms utilising the Canny edge detector
- Colour-based orientation histograms on the different channels of the opponent-colour system

- Curvature histograms
- Orientation-curvature histograms
- Colour histograms
- Wavelet coefficients
- Mean colour values (HSV)
- Geometric features
- Invariant Hu moments

For a preliminary evaluation of the different feature types non-hierarchical nearest-neighbour classifiers were used. On the data sets used where the features are extracted from a previously identified region of interest eliminating as much of the background as possible the high-dimensional histogram-based feature types such as colour histograms and the different orientation histogram feature types showed the best performance, whereas the low-dimensional features such as geometric features, invariant hu moments and mean colour values continuously showed weak performance.

When utilised within classifier hierarchies feature types that show good classification results are selected preferentially. The usage of a variety of diverse feature types distinguishes the proposed approach from other object recognition approaches.

C7: Implementation and Integration of the Developed Approach on a Robot

In order to show the feasibility and the operability of the proposed approach it has been integrated into a comprehensive cortical model and has been implemented on an autonomous mobile robot [23] [41]. The robot's functionality was inter alia presented on a NeuroBotic workshop [22].

15 Comparison of Hierarchical Classification Approach With Related Approaches

This chapter relates the approach developed within the context of this thesis to related research fields.

15.1 Related Work

The thesis at hand is related to varying degrees to the fields of pattern recognition, classifier fusion and uncertainty theory. In the following a choice of approaches of these fields is described.

In current literature there is much evidence for the usefulness of hierarchical approaches as well as the incorporation of uncertain knowledge in object recognition.

15.1.1 Hierarchical Classification Approaches

Object recognition is a well known problem in current literature. There are many approaches to classify objects from images. There are various approaches for three-dimensional object recognition e.g. [8] [49]. A biologically motivated approach emphasising the categorisation aspect is described in [68] [42] [76]. A hierarchical image representation that is scale and translation invariant is obtained by hierarchically building complex cells from simple cells. The so derived representation is then used to perform diverse object recognition tasks such as identification and categorisation.

Multiple classifier systems for 3D-object recognition are described in [1] [40]. These architectures are serial multiple classifier systems exploiting hierarchical output coding. It has been shown that improved recognition performance could be achieved by decomposing the decision process into several stages utilising coarse

to fine classification. These approaches rank among model-based object recognition [66]. The approach presented in this paper is not model-based, but it classifies objects from single views. Poggio et. al. used a radial basis function network architecture for view-invariant object recognition [65]. An appearance-based approach to object recognition using colour, shape and texture histograms and a feed-forward neural network is presented in [54] [55]. Fairly expensive features are used in this approach. Objects are not localised before classification. Nayar and Murase describe in [59] [60] another appearance-based object recognition approach which comprehends image segmentation, feature extraction and appearance matching utilising splines and radial basis function networks for object classification. Eigenspace representations are used as features.

As the decomposition of problems into simpler sub-problems features advantages such as effectiveness and efficiency in learning and interpretability modular learning has attracted much interest recently. There are various ways of dividing a problem into less complex sub-problems. One possible way is a partitioning of the output space. In [46] [47] a hierarchical decomposition of a multi-class problem into several two-class problems is performed utilising Fisher discriminant analysis in combination with a deterministic annealing process. The grouping of the classes is based on the class distributions resulting in a binary tree architecture. Simple Bayesian classifiers are used to solve the sub-problems. The approach is applied to the problem of categorising landcover using hyperspectral data. Instead of Bayesian classifiers support vector machines are used in [67]. The approach has been evaluated on several pattern recognition problems. An alternative method for the decomposition of the output space is applied in [14]. A max-cut algorithm is successively applied in order to find those class partitions that have a maximal distance. As classifiers support vector machines are used. Another approach for building a hierarchical binary tree classifier architecture is proposed in [15] where a self-organising map is trained in the kernel space where classification by the deployed support vector machines takes place. On the basis of the trained self-organising map the class grouping is determined by identifying the grouping that maximises the inter-group distance while minimising the intra-group variance. In this architecture no disjoint partitioning of the classes is forced, but overlaps are allowed and are shown to improve the performance.

In [39] [74] [77] examples of object recognition approaches developed and investigated within our department are presented. The hierarchical approaches showed encouraging results. The approaches differ in particular with respect to the way the classifier hierarchy is generated and the kind of features and classifiers used within the hierarchy.

15.1.2 Classification Approaches Utilising Dempster-Shafer Evidence Theory

Dempster-Shafer evidence theory has been applied to classifier fusion in numerous applications for pattern recognition.

Dempster-Shafer theory was used for multiple classifier fusion in [51]. This approach uses prototype-based classifiers and calculates belief functions from distance measures of different classifiers which are then combined utilising Dempster-Shafer evidence theory. As distance measures the inter-class-distances and intra-class-distances were used. The approach was evaluated in the field of online script recognition.

In [87] classification rates, misclassification rates and rejection rates were used to derive basic probability assignments. Dempster's combination rule is applied to combine the evidences. This approach considers an extra class representing unknown classes or ignorance and it assigns belief to singleton hypotheses, their complement and to the universal proposition Ω. In contrast to the method developed in this thesis the classifiers used in [87] only have class labels as output and do not produce information that can be interpreted as class memberships or other measurements. The approach was applied to the problem of recognising handwritten numerals and scored well compared to other approaches.

A technique closely related to decision templates [48] is used to calculate degrees of belief in [70]. The distances between the classifier outputs for the sample to be classified and the mean classifier outputs calculated on the training samples are transformed into basic probability assignments. The so calculated evidences are then combined using the orthogonal sum. Several experiments in the field of digits and character recognition have been conducted to test this method and it was also part of the experimental comparison of the decision templates approach for classifier fusion to other well-established methods in [48] where it compared favourably well.

In [2] this approach has been varied by using reference outputs adapted to the training data so that the overall mean square error is minimised instead of simply using the mean classifier outputs.

Dempster-Shafer evidence theory is used to combine the normalised outputs of multiple classifiers and to reject samples in case of highly conflicting information in [82].

If at all, these approaches only exploit the possibility to allocate evidence to non-atomic hypotheses by assigning masses to atomic hypotheses θ_i and to their not necessarily atomic complement $\overline{\theta_i}$ or to the frame of discernment Ω. The approach presented in this work utilises the possibility to also assign masses to sets of hypotheses as the classifier hierarchy naturally provides classification

results for sets of classes.

In [58] expert knowledge about the domain of application, namely the detection of anti-personnel mines, is used to calculate basic probability assignments not only for atomic hypotheses but also for composite hypotheses. Hence this approach is rather specific and less general than the proposed approach.

An approach based on [87] for the fusion of multi-level decisions utilising the transferable belief model is proposed in [56]. In this method a classifier ensemble is used where the individual classifiers each provide decisions in a hierarchical decision space, i.e. their decisions can either be a single class, a set of classes or a rejection. The hierarchy spanning the decision space is either given or can be build on the basis of the confusion matrix. Like in [87] the classifiers used are abstract level classifiers providing only class labels as classification result. The basic probability assignments are therefore calculated from a confusion matrix determined on a training data set. This approach a classifier ensemble instead of a classifier hierarchy and does not incorporate the strength of the classifier responses that provide information on the degree of class membership.

15.1.3 Incremental Learning Approaches

An example for an incremental learning approaches are **A**daptive **R**esonance **T**heory (ART) networks. ART networks [35] [13] allow for online learning of evolving data sets. If a presented sample is similar enough to an learnt prototype this prototype is adjusted to the sample, otherwise a new prototype is defined by the sample. In the ARAM model [81] new classes can be learnt while preserving previously learnt classes and so the stability-plasticity dilemma is regarded. However, ART networks are non-hierarchical networks and they consider new samples one at a time, whereas in this work more than one sample is allowed for and the resulting classifiers develop a hierarchical structure.

15.2 Classification of Work

The proposed approach provides general framework for pattern recognition. The research presented in this thesis focused on the recognition of three-dimensional objects as one possible domain of application. For the object recognition a view-based approach was chosen. The majority of the feature types used were histogram based.

The presented approach is characterised inter alia by the usage of multiple feature types, the general applicability of the approach and the ability to incrementally learn new classes. Hence the the proposed approach in particular features versatility.

The application of the Dempster-Shafer evidence theory to hierarchical neural network classifiers is another main subject matter. The evidence theory could successfully be used for fusing the information provided by the individual classifiers within the hierarchy. Hereby hierarchically structured information provided by the single classifiers is combined resulting not only in evidence for the single classes but also for the possible sets of classes.

16 Conclusions

The presented approach of hierarchical neural network classifiers was thoroughly evaluated and substantially examined under miscellaneous aspects. The approach proved functional and showed encouraging results and various advantages.

The approach offers a substantial universality: The developed classifier hierarchies can be applied to diverse pattern recognition problems of variable size. Within the hierarchies miscellaneous classifier types can be utilised and the usage of various feature types is immanent. Thus classifier types and feature types can be chosen according to the classification problem at hand. The hierarchy generation can also easily be adapted to a given classification problem by varying the size of the simple classification problems that the complex classification problem is decomposed into or by adjusting the valuating function that forms the basis of the hierarchy generation process. There is also the possibility of a customised manual definition of the hierarchy structure.

For classifiers only applicable to binary classification problems such as support vector machines it provides means to solve multi-class problems by splitting the original classification problem into several binary classification problems. The availability of intermediate results is especially beneficial when the classification problem is deployed in order to fulfill certain task that only require coarse information.

Hierarchical neural network classifiers compare favourably well to simple classifiers. With regards to classification results they yield the same or even improved performance. Within the context of this work several strategies for fusing the individual classifier results in a classifier hierarchy were developed. The complex fusion strategies such as the decision tree like approach, the evidence theoretic approach and the approach utilising codes generated from the classifier hierarchy consistently showed good classification performance.

Hierarchical neural network classifiers proved most suitable for the application of the belief theory. The potentially uncertain information provided by the classifier hierarchies could be processed adequately by the concepts offered by the belief

theory resulting in a fusion strategy that yields constantly good classification performance.

The generation of similarity preserving sparse binary codes is facilitated by hierarchical neural network classifiers. This allows for easy integration of the proposed object recognition approach into comprehensive cortical models that are based on associative memories.

The easy extendability of the classifier hierarchies makes them particulary suitable for the application in environments where unfamiliar objects are likely to be encountered as it is possible to incrementally learn new objects during run-time.

These advantages and capabilities make hierarchical neural network classifiers a universal and promising approach that is applicable for various problem areas and offers diverse possibilities for further enhancements.

Bibliography

Bibliography

[1] Alireza R. Ahmadyfard and Josef Kittler. A multiple classifier system approach to affine invariant object recognition. In James L. Crowley, Justus H. Piater, Markus Vincze, and Lucas Paletta, editors, *Computer Vision Systems, Third International Conference, ICVS*, volume 2626 of *Lecture Note in Computer Science LNCS*, pages 438–447, Graz, Austria, 2003. Springer.

[2] Ahmed Al-Ani. A new technique for combining multiple classifiers using the dempster-shafer theory of evidence. *Journal of Artificial Intelligence Research*, 17:333–361, 2002.

[3] B. G. Batchelor and B. R. Wilkins. Method for location of clusters of patterns to initialize a learning machine. *Electronics Letters*, 5(20):481–483, 1969.

[4] Richard E. Bellman. *Adaptive Control Processes.* Princeton University Press, Princeton, NJ., 1961.

[5] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 2000.

[6] Isabelle Bloch. Information combination operators for data fusion: A comperative review with classification. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*, 26(1):52–67, 1996.

[7] Remco R. Bouckaert and Frank Eibe. Evaluating the replicability of significance tests for comparing learning algorithms. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *Advances in Knowledge Discovery and Data Mining, Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2004, Sydney, Australia, May 26-28, 2004*, volume 3056 of *Lecture Notes in Artificial Intelligence LNAI*, pages 3–12. Springer, 2004.

[8] R. Brooks. Model-based three-dimensional interpretations of two-dimensional images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:140–149, 1983.

[9] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[10] Antony Browne and Ron Sun. Connectionist inference models. *Neural Networks*, 14(10):1331–1355, 2001.

[11] G. Buchsbaum and A. Gottschalk. Trichromacy, opponent colours coding and optimum colour information transmission in the retina. *Porceedings of the Royal Socienty of London B Biological Science*, 220(1218):89–113, 1983.

[12] John Francis Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 8(6):679–698, 1986.

[13] Gail A. Carpenter and Stephen Grossberg. Adaptive resonance theory. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 87–90. MIT Press, Cambridge, 2nd edition, 2002.

[14] Yangchi Chen, Melba M. Crawford, and Joydeep Ghosh. Integrating support vector machines in a hierarchical output space decomposition framework. In *IEEE International Geoscience and Remote Sensing Symposium*, volume II, pages 949 – 952, 2004.

[15] Sungmoon Cheong, Sang Hoon Oh, and Soo-Young Lee. Support vector machines with binary tree architecture for multi-class classification. *Neural Information Processing - Letters and Reviews*, 2(3):47–51, 2004.

[16] David M. Coppola, Harriett R. Purves, Allison N. McCoy, and Dale Purves. The distribution of oriented contours in the real world. *Proceedings of the National Academy of Sciences USA*, 95(7):4002–4006, 1998.

[17] József Dombi. A general class of fuzzy operators, the de morgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy Sets and Systems*, 8:149–163, 1982.

[18] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2nd edition, 2001.

[19] Sharon Duvdevani-Bar and Shimon Edelman. Visual recognition and categorization on the basis of similarities to multiple class prototypes. *International Journal of Computer Vision*, 33(3):201–228, 1999.

[20] Shimon Edelman and Sharon Duvdevani-Bar. A model of visual recognition and categorization. *Philosophical Transactions of the Royal Society London (B): Biological Sciences*, 352(1358):1191–1202, 1997.

[21] Peter W. Eklund and Anh Hoang. A comparative study of public domain supervised classifer performance on the uci database. *Australian Journal of Intelligent Systems*, 9(1):1–47, 2006.

[22] Rebecca Fay, Ulrich Kaufmann, Andreas Knoblauch, Heiner Markert, and Gïther Palm. Integrating Object Recognition, Visual Attention, Language and Action Processing on a Robot Using a Neurobiologically Plaussible Associative Architecture. In *KI Workshop on NeuroBotics, Ulm, September 2004*, 2004.

[23] Rebecca Fay, Ulrich Kaufmann, Andreas Knoblauch, Heiner MArkert, and Gnther Palm. Combining visual attention, object recognition and associative information processing in a neurobotic system. In Stefan Wermter, Günther Palm, and Mark Elshaw, editors, *Biomimetic Neural Learning for Intelligent Robots. Intelligent Systems, Cognitive Robotics, and Neuroscience.*, volume 3575 of *Lecture Notes in Computer Science LNAI*, pages 118–143. Springer, Berlin, Heidelberg, 2005.

[24] Rebecca Fay, Ulrich Kaufmann, Friedelm Schwenker, and Günther Palm. Learning object recognition in an neurobotic system. *3rd IWK Workshop SOAVE2004 - SelfOrganization of AdaptiVE behavior, Illmenau, Germany*, pages 198–209, 2004.

[25] Rebecca Fay, Friedhelm Schwenker, and Günther Palm. Incremental learning in hierarchical neural networks for object recognition. In Joaquim Filipe, Juan Andrade-Cetto, and Jean-Louis Ferrier, editors, *Proceedings of the Second International Confernece on Informatics in Control, Automation and Robotics ICINCO 2005, Barcelona, Spain, September 14-17, 2005, 4 Volumes / CD*, volume III, pages 298–303. INSTICC Press, 2005.

[26] Rebecca Fay, Friedhelm Schwenker, and Günther Palm. Evidence Based Reasoning in Classifier Hierarchies. In *Second International Workshop on Neural-Symbolic Learning and Reasoning NeSy 2006.*, 2006.

[27] Rebecca Fay, Friedhelm Schwenker, Christian Thiel, and Günther Palm. Hierarchical Neural Networks Utilising Dempster-Shafer Evidence Theory. In Friedhelm Schwenker, editor, *2nd IAPR TC3 International Workshop on Artificial Neural Networks in Pattern Recognition ANNPR 2006.*., volume 4087 of *Lecture Notes in Artifical Intelligence LNAI*, pages 198–209. Springer, Berlin, Heidelberg, 2006.

[28] Wolfgang Förstner. A framework for low-level feature extraction. In Jan-Olof Eklundh, editor, *Computer Vision - ECCV 1994. Proceedings of the Third European Conference on Computer Vision. Volume II. Stockholm, Sweden, May 2-6, 1994.*, volume 801 of *Lecture Notes in Computer Science LNCS*, pages 383–394, Berlin, Heidelberg, 1994. Springer.

[29] David J. Freedman, Maximilian Riesenhuber, Tomaso Poggio, and Earl K. Miller. Categorical representation of visual stimuli in the primate prefrontal cortex. *Science*, 291(5502):312–316, 2001.

[30] Wiliam T. Freeman and Michael Roth. Orientation histograms for hand gesture recognition. In *IEEE International Workshop on Automatic Face- and Gesture-Recognition*, pages 296–301, Zürich, Switzerland, 1995.

[31] Peter W. Frey and David J. Slate. Letter recognition using holland-style adaptive classifiers. *Machine Learning*, 6(2):161–182, 1991.

[32] Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Stanford University, 1996.

[33] Luis Garmendia. The evolution of the concept of fuzzy measure. In Da Ruan, Guoqing Chen, Etienne E. Kerre, and Geert Wets, editors, *Intelligent Data Mining. Techniques and Applications*, volume 5 of *Studies in Computational Intelligence*, pages 185–200. Springer, Berlin, Heidelberg, 2005.

[34] Rafael C. Gonzales and R. E. Woods. *Digital Image Processing.* Addison-Wesley, 2nd edition, 1992.

[35] Stephen Grossberg. Adaptive resonance theory. Technical Report TR-2000-024, Center for Adaptive Systems and Department of Cognitive and Neural Science, Boston University, 2000.

[36] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference, Manchester, 1988.*, pages 147–151, 1988.

[37] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8:179–187, 1962.

[38] Ulrich Kaufmann, Rebecca Fay, Heiner Markert, and Günther Palm. Neural networks for visual object recognition based on selective attention. In *SenseMaker Workshop on Life Like Perception Systems*, 2005.

[39] Hans A. Kestler, Stefan Sablatnög, Steffen Simon, Stefan Enderle, Axel Baune, Gerhrad K. Kraetzschmar, Friedhelm Schwenker, and Günther Palm. Concurrent object identification and localization for a mobile robot. *Künstliche Intelligenz*, pages 23–29, 2000.

[40] Josef Kittler, Alireza R. Ahmadyfard, and David Windridge. Serial multiple classifier systems expoiting a coarse to fine output coding. In Terry Windeatt and Fabio Roli, editors, *Multiple Classifier Systems, 4th International Workshop, MCS*, volume 2709 of *Lecture Notes in Computer Science LNCS*, pages 106–114, Guilford, UK, 2003. Springer.

[41] Andreas Knoblauch, Rebecca Fay, Ulrich Kaufmann, Heiner Markert, and Günter Palm. Associating words to visually recognized objects. In S. Coradeschi and A. Saffiotti, editors, *Anchoring symbols to sensor data. Papers from the AAAI Workshop. Technical Report WS-04-03*, pages 10–16. AAAI Press, Menlo Park, California, 2004.

[42] Ulf Knoblich, Maximilian Riesenhuber, David J. Freedman, Earl K. Miller, and Tomaso Poggio. Visual categorization: How the monkey brain does it. In Heinrich H. Bülthoff, Seong-Whan Lee, Tomaso A. Poggio, and Christian Wallraven, editors, *Biologically Motivated Computer Vision: Second International Workshop, BMCV 2002, Tübingen, Germany, November 22-24, 2002*, volume 2525 of *Lecture Notes in Computer Science LNCS*, pages 273–281, Berlin, Heidelberg, 2002. Springer.

[43] T. Kohonen. *Self-organizing maps.* Springer, Berlin, 1995.

[44] Teuvo Kohonen, Jussi Hynninen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. Lvq pak: The learning vector quantization program package. Technical Report A30, Helsinki University of Technology, 1996.

[45] Ulrich H.-G. Kressel. The impact of the learning-set size in handwritten-digit recognition. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Proceedings of the International Confernece on Artificial Neural Networks, ICANN 1991, Helsinki, Finland*, pages 1685–1689, Amsterdam, North-Holland, 1991. Elsevier Science Publishers B.V.

[46] Shailesh Kumar, Joydeep Ghosh, and Melba M. Crawford. A hierarchical multiclassifier system for hyperspectral data analysis. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science LNCS*, pages 270–279. Springer, 2000.

[47] Shailesh Kumar, Joydeep Ghosh, and Melba M. Crawford. Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *International Journal on Pattern Analysis and Applications*, 5(2):210–220, 2002.

[48] Ludmila I. Kuncheva, James C. Bezdek, and Robert P. W. Duin. Decision templates for multiple classifier fusion: An experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001.

[49] D. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[50] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Berkeley: University of California Press, 1967.

[51] Eberhard Mandler and Jürgen Schürmann. Combining the classification results of independent classifiers based on the dempaster/shafer theory of evidence. In *Pattern Recognition and Artificial Intelligence PRAI*, pages 381–393, 1988.

[52] Gerd Mayer. *Objekterkennung in hochdynamischen Systemen.* PhD thesis, University of Ulm, 2006.

[53] Kenneth McGarry, Stefan Wermter, and John MacIntyre. Hybrid neural systems: From simple coupling to fully integrated neural networks. *Neural Computing Surveys*, 2:62–93, 1999.

[54] Bartlett W. Mel. Seemore: A view-based approach to 3-d object recognition using multiple visual cues. In David S. Touretzky, Michael Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Infrormation Processing Systems NIPS 1996*, volume 8, pages 865–871. MIT Press, 1995.

[55] Bartlett W. Mel. Seemore: Combining color, shape, and texture histogramming in a neurally inspired approach to visual object recognition. *Neural Computation*, 9:777–804, 1997.

[56] David Mercier, Genevieve Cron, Thierry Denoeux, and Mylene Masson. Fusion of multi-level decision systems using the transferable belief model. In *The Eighth International Conference on Information Fusion FUSION 2005, July 25-29, Philadelphia, USA*, volume 2, 2005.

[57] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[58] Nada Milisavljevic and Isabelle Bloch. Sensor fusion in anti-personnel mine detection using a two-level belief function model. *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews*, 33(2):269–283, 2003.

[59] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[60] Shree K. Nayar, Sameer A. Nene, and Hiroshi Murase. Real-time 100 object recognition system. In *ARPA Image Understanding Workshop IUW 96*, pages 1223–1227, 1996.

[61] Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia Object Image Library (COIL-100). Technical Report Technical Report CUCS-006-96, Department of Computer Science, Columbia University, February 1996.

[62] Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia Object Image Library (COIL-20). Technical Report Technical Report CUCS-005-96, Department of Computer Science, Columbia University, February 1996.

[63] Günther Palm and Gerhard K. Kraetzschmar. Sfb 527: Integration symbolischer und subsymbolischer informationsverarbeitng in adaptiven sensomotorischen systemen. In Matthias Jarke, Klaus Pasedach, and Klaus Pohl, editors, *Informatik '97 - Informatik als Innovationsmotor, 27. Jahrestagung der Gesellschaft für Informatik*, Aachen, 1997. Springer.

[64] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51:406–413, 1955.

[65] Tomasio Poggio and Shimon Edelman. A network that learns to recognize three-dimensional objects. *Letters to Nature*, 343:263–266, 1990.

[66] Arthur R. Pope. Model-based object recognition - a survey of recent research. Technical Report TR-94-04, University of British Columbia, Vancouver, BC, Canada, 1994.

[67] Suju Rajan and Joydeep Ghosh. An empirical comparison of hierarchical vs. two-level approaches to multiclass problems. In Fabio Roli, Josef Kittler, and Terry Windeatt, editors, *Multiple Classifier Systems*, volume 3077 of *Lecture Notes in Computer Science LNCS*, pages 283–292. Springer, 2004.

[68] Maximilian Riesenhuber. and Tomaso Poggio. Models of object recognition. *Nature Neuroscience*, 3:1199–1204, 2000.

[69] Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, January 1996.

[70] Galina L. Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777–781, 1994.

[71] Galina L. Rogova and Vincent Nimier. Reliability in information fusion: Literature survey. In Per Svensson and Johan Schubert, editors, *Proceedings of the Seventh International Conference on Information Fusion*, volume II, pages 1158–1165, Mountain View, CA, Jun 2004. International Society of Information Fusion.

[72] John C. Russ. *The Image Processing Handbook*. CRC Press, 3rd edition, 1998.

[73] B. Schweizer and A. Skalar. Associative functions and statistical triangle inequalities. *Publicationes Mathematicae Debrecen*, 8:169–186, 1961.

[74] Friedhelm Schwenker and Hans A. Kestler. 3-d visual object classification with hierarchical radial basis function network. In Robert J. Howlett and Lakhmi C. Jain, editors, *Radial Bais Function Networks*, volume 2, pages 269–293. Physica-Verlag, Heidelberg, New York, 2001.

[75] Friedhelm Schwenker, Hans A. Kestler, and Günther Palm. Three learning phases for radial-basis-function networks. *Neural Networks*, 14:439–458, 2001.

[76] Thomas Serre, Maximilian Riesenhuber, Jennifer Louie, and Tomaso Poggio. On the role of object-specific features for real world object recognition in biological vision. In Heinrich H. Bülthoff, Seong-Whan Lee, Tomaso A. Poggio, and Christian Wallraven, editors, *Biologically Motivated Computer Vision: Second International Workshop, BMCV 2002, Tübingen, Germany, November 22-24, 2002*, volume 2525 of *Lecture Notes in Computer Science LNCS*, pages 387–397, Berlin, Heidelberg, 2002. Springer.

[77] Steffen Simon, Friedhelm Schwenker, Hans A. Kestler, Gerhrad K. Kraetzschmar, and Günther Palm. Hierarchical object classification for autonomous mobile robots. In *International Conference on Artificial Neural Networks (ICANN)*, pages 831–836, 2002.

[78] Philippe Smets. The combination of evidence in the transferable belief model. *IEEE Transactions on Pattern Analysis and Machine Learning*, 12(5):447–458, 1990.

[79] Philippe Smets and Robert Kennes. The transferable belief model. *Artificial Intelligence*, 66(2):191–234, 1994.

[80] Alvy Ray Smith. Color gamut transform pairs. *Computer Graphics*, 12(3):12–19, 1978.

[81] Ah-Hwee Tan. Adaptive resonance associative map. *Neural Networks*, 8(3):437–446, 1995.

[82] Christian Thiel, Friedhelm Schwenker, and Günther Palm. Using dempster-shafer theory in mcf systems to reject samples. In Nikunj C. Oza, Robi Polikar, Josef Kittler, and Fabio Roli, editors, *Proceedings of the 6th International Workshop on Multiple Classifier Systems MCS 2005*, volume 3541 of *Lecture Notes in Computer Science LNCS*, pages 118–127. Springer, 2005.

[83] Julius T. Tou and Rafael C. Gonzales. *Pattern Recognition Principles*. Addison-Wesley, 1979.

[84] Edward Walter. Einige einfache nichtparametrische überall wirksame tests zur prüfung der zweistichprobenhypothese mit paarigen beobachtungen. *Metrika*, 1(4):81–88, 1958.

[85] S. Weber. A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorms. *Fuzzy Sets and Systems*, 11:115–134, 1983.

[86] D.J. Willshaw, O.P. Buneman, and H.C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222:960–962, 1969.

[87] Lei Xu, Adam Krzyzak, and Ching Y. Suen. Methods of combining multiple classifiers and their application to handwriting recognition. *IEEE Transaction on Systems, Man and Cybernetics*, 22(3):418–435, 1992.

[88] Ronald R. Yager. On a general class of fuzzy connectives. *Fuzzy Sets and Systems*, 4:235–242, 1980.

[89] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[90] Zijian Zheng. A benchmark for classifier learning. Technical Report 474, University of Sydney, 1993.

Appendix

A Detailed Results of the Statistical Evaluation

A.1 Features for 3D-Object Recognition

The tables A.1 and A.2 list which feature types were used on which data set.

Feature	Fruits	COIL-20	COIL-100
colorh	+	-	-
colorhsv	+	-	-
colorhist1x1	+	+	+
colorhist2x2	+	+	+
colorhist3x3	+	+	+
colorhist4x4	+	+	+
APQYB1x1	+	-	+
APQYB2x2	+	-	+
APQYB3x3	+	-	+
APQYB4x4	+	-	+
APQRG1x1	+	-	+
APQRG2x2	+	-	+
APQRG3x3	+	-	+
APQRG4x4	+	-	+
APQBW1x1	+	+	+
APQBW2x2	+	+	+
APQBW3x3	+	+	+
APQBW4x4	+	+	+

Table A.1: *Feature types extracted from different data sets (continued on the next page). The feature types APQYB, APQRG, APQBW are the orientation histograms on the yellow-blue, red-green and black-white opponent colour channels respectively.*

Feature	Fruits	COIL-20	COIL-100
geometricroundness	+	+	-
geometriccompactness	+	+	-
geometricformFactor	+	+	-
geometricboundingBoxRatio	+	+	-
geometricobjectRatio	+	+	-
geometricall	+	+	-
hu1	+	+	-
hu2	+	+	-
hu3	+	+	-
hu4	+	+	-
hu5	+	+	-
hu6	+	+	-
hu7	+	+	-
huall	+	+	-
orientcurvehist1x1	+	+	+
orientcurvehist2x2	+	+	+
orientcurvehist3x3	+	+	+
orientcurvehist4x4	+	+	+
curvehist1x1	+	+	+
curvehist2x2	+	+	+
curvehist3x3	+	+	+
curvehist4x4	+	+	+
canny1x1	+	+	+
canny2x2	+	+	+
canny3x3	+	+	+
canny4x4	+	+	+
sobel1x1	+	+	+
sobel2x2	+	+	+
sobel3x3	+	+	+
sobel4x4	+	+	+
wavelet1x1	+	+	+
wavelet2x2	+	+	+
wavelet3x3	+	+	+
wavelet4x4	+	+	+

Table A.2: *Feature types extracted from different data sets.*

The mean classification errors and the averaged ranks for the different feature types and classifiers on the different data sets are listed in the tables A.3, A.4, A.5, A.6, A.7, A.8, A.9 and A.10.

Feature	1-NN		3-NN		5-NN		Fuzzy 3-NN		Fuzzy 5-NN		Mean	
	error	rank	error	rank	error	rank	error	rank	error	rank	error	rank
APQBW1x1	21.82± 4.24%	25	21.45± 4.53%	24	21.94± 4.16%	24	31.79± 5.01%	23	44.93± 4.97%	23	28.39± 10.23%	24
APQBW2x2	10.27± 3.32%	12	12.61± 3.27%	11	12.62± 3.38%	8	20.26± 4.44%	11	33.98± 5.36%	14	17.95± 9.59%	11
APQBW3x3	10.82± 3.23%	13	13.39± 3.92%	14	15.69± 3.65%	18	20.50± 3.88%	12	34.42± 4.35%	15	18.96± 9.19%	14
APQBW4x4	9.32 ± 2.88%	9	12.58± 3.17%	10	15.19± 3.49%	15	21.08± 4.06%	14	33.89± 4.64%	12	18.41± 9.41%	12
APQRG1x1	36.49± 5.35%	32	33.88± 5.43%	31	34.38± 4.50%	31	48.71± 4.89%	32	60.43± 4.54%	30	42.78± 11.48%	31
APQRG2x2	22.12± 4.11%	26	21.08± 4.42%	23	19.74± 3.68%	21	35.04± 4.67%	25	51.74± 4.73%	27	29.94± 12.96%	24
APQRG3x3	12.51± 3.74%	15	13.75± 3.71%	16	14.00± 3.59%	9	25.86± 4.42%	20	41.68± 4.78%	21	21.56± 11.89%	16
APQRG4x4	13.17± 3.47%	17	13.69± 3.45%	15	14.81± 3.63%	14	22.27± 4.41%	15	35.93± 5.23%	19	19.97± 9.55%	16
APQYB1x1	41.82± 5.64%	35	39.87± 5.00%	34	36.88± 4.69%	33	52.08± 4.82%	34	62.05± 4.99%	33	46.54± 10.56%	34
APQYB2x2	24.48± 4.44%	29	24.17± 3.93%	25	23.08± 4.46%	25	35.80± 4.69%	26	47.12± 5.29%	25	30.93± 10.39%	26
APQYB3x3	16.48± 3.95%	20	15.00± 3.56%	17	15.25± 3.42%	16	25.58± 4.73%	19	34.96± 4.82%	17	21.45± 8.83%	18
APQYB4x4	15.06± 3.49%	19	15.12± 3.58%	18	15.61± 3.75%	17	24.64± 4.05%	17	34.89± 4.63%	16	21.06± 8.74%	17
canny1x1	43.18± 5.35%	36	41.93± 4.73%	35	40.85± 4.58%	35	52.04± 4.63%	33	64.88± 4.74%	35	48.57± 10.26%	35
canny2x2	9.05 ± 3.11%	8	11.74± 3.31%	9	14.54± 3.18%	11	19.26± 4.12%	8	31.20± 4.79%	9	17.16± 8.65%	9
canny3x3	5.02 ± 2.65%	5	8.74 ± 3.10%	6	10.11± 2.98%	6	11.82± 3.30%	6	23.37± 4.18%	6	11.81± 7.01%	6
canny4x4	5.20 ± 2.48%	6	8.20 ± 2.90%	5	9.98 ± 2.56%	5	11.25± 3.21%	5	21.50± 3.82%	5	11.23± 6.30%	5
colorh	65.73± 2.79%	40	61.95± 3.15%	40	61.44± 3.20%	40	-	52	-	52	-	45
colorhsv	17.24± 3.38%	21	16.85± 3.44%	20	16.51± 3.22%	19	20.90± 3.78%	13	27.94± 4.21%	7	19.89± 5.64%	16
colorhist1x1	4.85 ± 2.35%	4	5.48 ± 2.63%	4	6.52 ± 2.80%	4	9.36 ± 2.85%	4	15.23± 3.42%	3	8.29 ± 4.73%	4
colorhist2x2	3.07 ± 1.78%	3	4.44 ± 2.06%	3	5.18 ± 2.27%	3	8.89 ± 2.68%	3	15.83± 3.84%	4	7.48 ± 5.29%	3
colorhist3x3	2.96 ± 1.78%	2	3.08 ± 1.77%	2	3.43 ± 2.00%	2	6.45 ± 2.51%	2	12.25± 3.36%	1	5.64 ± 4.26%	2
colorhist4x4	2.31 ± 1.49%	1	2.44 ± 1.82%	1	3.01 ± 1.90%	1	6.33 ± 2.50%	1	12.67± 3.43%	2	5.35 ± 4.58%	1

Table A.3: *Mean error rates and ranks for the different feature types on the fruits data set (continued on the next page).*

Feature	1-NN		3-NN		5-NN		Fuzzy 3-NN		Fuzzy 5-NN		Mean	
	error	rank	error	rank	error	rank	error	rank	error	rank	error	rank
curvehist1x1	56.79± 4.49%	39	55.21± 5.15%	39	51.70± 5.48%	39	62.61± 5.33%	38	72.58± 4.98%	38	59.78± 8.90%	39
curvehist2x2	38.62± 4.86%	33	37.77± 5.11%	33	39.17± 5.15%	34	53.70± 5.19%	36	64.58± 5.10%	34	46.77± 11.83%	34
curvehist3x3	28.75± 4.40%	31	29.83± 4.83%	28	29.80± 4.36%	28	46.42± 5.09%	31	61.68± 5.52%	32	39.30± 13.87%	30
curvehist4x4	21.45± 4.05%	24	24.81± 4.25%	27	25.51± 4.07%	26	38.37± 5.06%	28	54.90± 4.55%	29	33.01± 13.14%	27
geometricall	46.76± 4.67%	37	43.94± 4.74%	37	41.77± 4.62%	36	53.55± 4.92%	35	66.54± 4.69%	37	50.51± 10.11%	36
geometric-bounding-BoxRatio	74.08± 4.47%	46	74.55± 4.21%	46	74.89± 4.07%	47	-	52	-	52	-	48
geometric-compactness	72.33± 4.85%	43	73.42± 3.52%	45	72.25± 5.01%	44	72.73± 4.32%	41	74.63± 4.23%	40	73.07± 4.49%	43
geometric-formFactor	72.52± 5.25%	45	70.69± 4.85%	43	70.21± 4.70%	43	74.82± 4.33%	44	78.06± 4.38%	45	73.26± 5.52%	44
geometric-objectRatio	70.77± 4.57%	42	68.45± 4.36%	42	66.37± 4.93%	42	71.99± 4.91%	40	74.77± 4.36%	42	70.47± 5.44%	42
geometric-roundness	72.39± 4.91%	44	73.33± 3.49%	44	72.33± 4.99%	45	72.98± 4.39%	43	74.75± 4.19%	41	73.16± 4.50%	43
hu1	68.96± 4.88%	41	65.35± 4.67%	41	62.55± 4.68%	41	72.80± 4.43%	42	75.86± 3.86%	43	69.10± 6.60%	42
hu2	75.43± 4.16%	48	75.99± 4.60%	48	77.18± 4.67%	52	76.74± 4.01%	46	77.31± 3.97%	44	76.53± 4.33%	48
hu3	79.95± 3.79%	52	78.36± 3.97%	52	76.49± 3.51%	50	80.10± 4.26%	49	81.75± 3.66%	49	79.33± 4.23%	50
hu4	75.25± 4.71%	47	75.04± 3.99%	47	75.04± 4.22%	48	77.08± 4.63%	47	78.43± 4.38%	46	76.17± 4.59%	47
hu5	79.27± 4.64%	51	76.64± 4.46%	50	73.57± 4.53%	46	80.12± 4.41%	50	82.38± 4.37%	50	78.40± 5.40%	49
hu6	77.43± 4.18%	50	76.20± 4.02%	49	76.51± 4.08%	51	76.74± 4.15%	45	78.43± 4.26%	47	77.06± 4.20%	48
hu7	76.46± 4.23%	49	77.44± 4.16%	51	75.45± 3.68%	49	77.94± 4.27%	48	79.85± 4.10%	48	77.43± 4.34%	49
huall	53.98± 5.41%	38	53.48± 5.20%	38	51.24± 5.74%	38	62.93± 5.00%	39	73.73± 4.48%	39	59.07± 9.82%	38
orientcurve-hist1x1	40.46± 5.07%	34	43.87± 5.00%	36	42.10± 4.54%	37	53.88± 5.09%	37	66.01± 4.27%	36	49.26± 10.73%	36
orientcurve-hist2x2	28.71± 4.14%	30	34.57± 4.21%	32	35.30± 4.72%	32	45.19± 4.97%	30	61.43± 5.37%	31	41.04± 12.42%	31
orientcurve-hist3x3	23.85± 5.03%	28	30.44± 4.74%	29	33.33± 5.09%	30	39.90± 4.93%	29	53.17± 5.34%	28	36.14± 11.15%	29
orientcurve-hist4x4	22.65± 4.14%	27	30.61± 4.45%	30	32.46± 4.08%	29	37.19± 5.07%	27	51.08± 5.19%	26	34.80± 10.47%	28

Table A.4: *Mean error rates and ranks for the different feature types on the fruits data set (continued on the next page).*

Feature	1-NN		3-NN		5-NN		Fuzzy 3-NN		Fuzzy 5-NN		Mean	
	error	rank	error	rank	error	rank	error	rank	error	rank	error	rank
sobel1x1	20.08± 4.28%	22	19.37± 3.65%	22	20.73± 3.66%	23	30.21± 4.82%	22	43.26± 5.17%	22	26.73± 10.15%	22
sobel2x2	9.80 ± 3.14%	11	11.52± 3.21%	8	11.85± 3.04%	7	17.17± 3.64%	7	28.54± 4.78%	8	15.77± 7.74%	8
sobel3x3	9.63 ± 2.90%	10	12.93± 3.33%	13	14.74± 3.60%	13	19.63± 4.10%	10	32.37± 5.04%	11	17.86± 8.83%	11
sobel4x4	6.70 ± 2.73%	7	11.26± 3.05%	7	14.07± 3.07%	10	19.58± 3.99%	9	32.00± 4.51%	10	16.72± 9.40%	9
wavelet1x1	21.18± 4.31%	23	24.31± 4.57%	26	27.24± 4.52%	27	32.86± 4.88%	24	46.64± 5.45%	24	30.45± 10.15%	25
wavelet2x2	14.29± 3.53%	18	17.90± 3.54%	21	19.77± 3.84%	22	26.38± 4.73%	21	37.21± 5.66%	20	23.11± 9.16%	20
wavelet3x3	11.39± 3.26%	14	12.77± 3.80%	12	14.71± 3.83%	12	23.31± 4.37%	16	35.04± 4.68%	18	19.45± 9.70%	14
wavelet4x4	12.77± 3.44%	16	16.02± 4.19%	19	19.55± 3.83%	20	24.74± 4.38%	18	33.95± 4.33%	13	21.41± 8.45%	17

Table A.5: *Mean error rates and ranks for the different feature types on the fruits data set.*

Feature	1-NN		3-NN		5-NN		Fuzzy 3-NN		Fuzzy 5-NN		Mean	
	error	rank	error	rank	error	rank	error	rank	error	rank	error	rank
APQBW1x1	6.47± 2.10%	23	9.56± 2.25%	22	12.17± 3.06%	22	14.09± 2.72%	23	23.51± 2.97%	21	13.16± 6.36%	22
APQBW2x2	0.44± 0.55%	7	1.03± 0.91%	11	2.47± 1.40%	11	2.85± 1.40%	10	9.36± 2.30%	11	3.23± 3.50%	10
APQBW3x3	0.25± 0.42%	5	0.56± 0.67%	3	1.40± 0.93%	6	1.52± 0.88%	5	6.08± 1.77%	4	1.96± 2.36%	5
APQBW4x4	0.58± 0.61%	10	0.92± 0.85%	8	1.62± 1.04%	9	2.60± 1.23%	9	9.00± 2.03%	9	2.94± 3.35%	9
canny1x1	28.87± 3.24%	29	29.30± 3.12%	29	29.60± 3.32%	29	38.31± 3.50%	29	51.78± 3.86%	29	35.57± 9.47%	29
canny2x2	0.67± 0.63%	12	1.40± 1.04%	12	2.62± 1.29%	12	3.58± 1.28%	12	9.15± 2.44%	10	3.49± 3.34%	12
canny3x3	0.47± 0.61%	9	0.76± 0.72%	6	1.24± 0.91%	4	1.80± 1.07%	6	6.59± 2.02%	6	2.17± 2.54%	6
canny4x4	0.33± 0.48%	6	0.79± 0.74%	7	1.08± 1.01%	3	1.97± 1.16%	7	7.82± 2.06%	7	2.40± 3.02%	6
colorh	95.00± 0.28%	44	95.00± 0.28%	44	95.00± 0.28%	44	-	42	-	42	-	43
colorhsv	77.63± 1.59%	38	75.66± 1.82%	37	73.97± 1.97%	40	-	42	-	42	-	40
colorhist1x1	4.67± 1.64%	21	6.31± 1.74%	19	9.05± 2.04%	18	10.83± 2.19%	20	17.22± 2.64%	14	9.62± 4.83%	18
colorhist2x2	0.10± 0.27%	3	0.68± 0.73%	5	1.42± 0.97%	7	0.77± 0.80%	3	5.38± 1.55%	3	1.67± 2.13%	4
colorhist3x3	0.01± 0.10%	2	0.22± 0.35%	2	0.60± 0.64%	2	0.24± 0.45%	1	3.37± 1.49%	2	0.89± 1.47%	2
colorhist4x4	0.00± 0.00%	1	0.05± 0.20%	1	0.33± 0.48%	1	0.36± 0.48%	2	2.72± 1.23%	1	0.69± 1.20%	1
curvehist1x1	68.57± 3.75%	33	66.97± 3.11%	33	66.13± 3.06%	33	75.58± 3.54%	33	81.15± 3.25%	34	71.68± 6.69%	33
curvehist2x2	7.45± 1.95%	25	10.69± 2.30%	24	13.85± 2.74%	24	21.75± 3.24%	26	42.13± 3.69%	28	19.17± 12.75%	25
curvehist3x3	1.56± 1.04%	17	2.83± 1.30%	17	4.51± 1.69%	15	7.61± 2.02%	16	22.10± 2.81%	20	7.72± 7.71%	17
curvehist4x4	0.77± 0.62%	14	1.64± 1.09%	13	2.82± 1.37%	13	5.63± 1.68%	13	17.46± 2.59%	15	5.66± 6.33%	14

Table A.6: *Mean error rates and ranks for the different feature types on the COIL-20 data set (continued on the next page).*

Feature	1-NN		3-NN		5-NN		Fuzzy 3-NN		Fuzzy 5-NN		Mean	
	error	rank	error	rank	error	rank	error	rank	error	rank	error	rank
geometricall	13.29± 2.33%	26	13.73± 2.47%	26	13.96± 2.65%	26	18.94± 2.82%	24	27.47± 3.28%	24	17.48± 6.05%	25
geometric-bounding-BoxRatio	90.12± 1.76%	43	88.77± 1.84%	43	88.72± 1.65%	43	-	42	-	42	-	43
geometric-compactness	62.11± 3.00%	30	61.00± 3.40%	31	57.57± 3.18%	30	67.58± 3.65%	31	76.19± 3.60%	31	64.89± 7.33%	31
geometric-formFactor	74.08± 2.51%	34	72.38± 2.89%	34	69.98± 3.40%	34	76.71± 2.71%	34	78.90± 2.86%	32	74.41± 4.26%	34
geometric-objectRatio	77.78± 3.30%	39	77.81± 3.21%	40	71.59± 3.30%	37	80.91± 2.99%	39	85.93± 2.60%	40	78.80± 5.60%	39
geometric-roundness	62.24± 2.98%	31	60.95± 3.37%	30	57.57± 3.18%	30	67.48± 3.70%	30	76.08± 3.60%	30	64.86± 7.28%	30
hu1	75.31± 3.10%	36	74.74± 2.82%	35	70.92± 3.01%	35	77.45± 3.13%	35	82.33± 3.07%	35	76.15± 4.81%	35
hu2	79.13± 2.92%	42	79.24± 2.75%	41	74.56± 2.99%	41	81.47± 2.82%	41	85.03± 2.58%	39	79.89± 4.42%	41
hu3	76.99± 3.34%	37	76.35± 2.88%	38	72.15± 2.73%	38	80.90± 3.21%	38	86.15± 2.76%	41	78.51± 5.59%	38
hu4	78.83± 2.97%	41	76.88± 2.91%	39	73.96± 3.29%	39	81.15± 2.93%	40	84.67± 2.94%	38	79.10± 4.73%	39
hu5	75.02± 3.32%	35	75.11± 2.87%	36	71.18± 3.20%	36	79.26± 3.41%	36	84.37± 2.93%	37	76.99± 5.48%	36
hu6	67.03± 3.31%	32	62.85± 2.93%	32	58.69± 3.22%	32	70.72± 3.30%	32	78.99± 3.29%	33	67.66± 7.66%	32
hu7	78.10± 2.98%	40	80.25± 2.80%	42	76.66± 2.66%	42	80.70± 3.01%	37	84.35± 2.56%	36	80.01± 3.83%	39
huall	19.49± 2.89%	28	21.15± 3.24%	27	23.54± 3.11%	27	26.43± 3.21%	27	36.38± 3.16%	25	25.40± 6.73%	27
orientcurve-hist1x1	6.56 ± 1.99%	24	10.72± 2.52%	25	13.94± 2.86%	25	19.34± 3.37%	25	39.90± 3.44%	27	18.09± 12.04%	25
orientcurve-hist2x2	1.03 ± 0.85%	16	2.80 ± 1.39%	16	4.56 ± 1.87%	16	5.92 ± 2.09%	15	21.64± 3.31%	19	7.19 ± 7.70%	16
orientcurve-hist3x3	0.67 ± 0.60%	13	2.03 ± 1.19%	14	3.83 ± 1.73%	14	5.75 ± 1.76%	14	21.06± 2.61%	17	6.67 ± 7.60%	14
orientcurve-hist4x4	0.86 ± 0.71%	15	2.35 ± 1.23%	15	4.91 ± 1.73%	17	8.92 ± 2.34%	17	25.31± 3.29%	23	8.47 ± 9.09%	17

Table A.7: *Mean error rates and ranks for the different feature types on the COIL-20 data set (continued on the next page).*

Feature	1-NN		3-NN		5-NN		Fuzzy 3-NN		Fuzzy 5-NN		Mean	
	error	rank	error	rank	error	rank	error	rank	error	rank	error	rank
sobel1x1	6.45± 2.09%	22	9.69± 2.22%	23	12.31± 3.11%	23	14.04± 2.87%	22	23.70± 3.19%	22	13.24± 6.44%	22
sobel2x2	0.44± 0.55%	7	0.97± 0.87%	10	2.42± 1.43%	10	2.90± 1.45%	11	9.36± 2.27%	12	3.22± 3.51%	10
sobel3x3	0.24± 0.41%	4	0.57± 0.71%	4	1.36± 0.93%	5	1.46± 0.92%	4	6.25± 1.75%	5	1.98± 2.42%	4
sobel4x4	0.58± 0.61%	10	0.93± 0.84%	9	1.60± 1.04%	8	2.52± 1.20%	8	8.93± 2.07%	8	2.91± 3.33%	9
wavelet1x1	19.41± 2.64%	27	24.08± 2.87%	28	26.87± 3.09%	28	28.44± 3.11%	28	39.14± 3.60%	26	27.59± 7.23%	27
wavelet2x2	2.81± 1.31%	19	6.73± 1.95%	20	11.12± 2.18%	20	10.15± 2.12%	18	19.33± 2.60%	16	10.03± 5.87%	19
wavelet3x3	4.38± 1.56%	20	7.95± 2.10%	21	12.10± 2.37%	21	11.69± 2.31%	21	21.24± 2.81%	18	11.47± 6.07%	20
wavelet4x4	2.61± 1.21%	18	5.67± 1.78%	18	9.45± 1.94%	19	10.23± 2.04%	19	16.10± 2.62%	13	8.81± 4.96%	17

Table A.8: *Mean error rates and ranks for the different feature types on the COIL-20 data set.*

Feature	1-NN		3-NN		5-NN		Fuzzy 3-NN		Fuzzy 5-NN		Mean	
	error	rank	error	rank	error	rank	error	rank	error	rank	error	rank
APQBW1x1	12.63± 1.04%	29	16.49± 1.17%	29	20.31± 1.49%	30	21.04± 1.50%	30	34.13± 1.60%	29	19.59± 7.44%	29
APQBW2x2	2.09± 0.52%	9	4.00± 0.70%	10	6.25± 0.86%	11	7.02± 0.73%	11	17.86± 1.18%	14	6.58± 5.47%	11
APQBW3x3	2.12± 0.51%	11	4.18± 0.65%	12	6.41± 0.79%	12	6.88± 0.94%	10	17.41± 1.09%	12	6.55± 5.28%	11
APQBW4x4	3.79± 0.57%	22	6.16± 0.82%	21	8.14± 0.90%	19	9.59± 1.03%	18	19.80± 1.17%	16	8.58± 5.54%	19
APQRG1x1	16.47± 1.17%	32	19.85± 1.17%	32	22.26± 1.26%	31	25.77± 1.23%	31	39.24± 1.54%	31	23.33± 7.92%	31
APQRG2x2	1.48± 0.44%	7	3.24± 0.64%	8	5.39± 0.75%	9	6.53± 0.66%	9	17.02± 1.13%	10	5.86± 5.38%	9
APQRG3x3	1.02± 0.34%	4	2.14± 0.49%	5	3.60± 0.74%	5	4.82± 0.71%	5	14.04± 1.27%	5	4.44± 4.57%	5
APQRG4x4	2.10± 0.47%	10	3.51± 0.60%	9	5.03± 0.68%	8	6.39± 0.81%	8	14.92± 1.24%	6	5.67± 4.48%	8
APQYB1x1	27.35± 1.25%	34	29.55± 1.33%	34	30.69± 1.35%	34	37.76± 1.47%	33	50.58± 1.39%	33	33.88± 8.37%	34
APQYB2x2	4.34± 0.84%	24	5.83± 0.84%	20	7.65± 1.01%	18	10.99± 1.10%	21	23.84± 1.47%	21	9.50± 6.90%	21
APQYB3x3	1.57± 0.42%	8	2.60± 0.56%	7	3.95± 0.65%	7	5.86± 0.94%	7	15.68± 1.08%	8	5.17± 4.94%	7
APQYB4x4	2.55± 0.55%	16	4.06± 0.72%	11	5.69± 0.83%	10	7.15± 0.87%	12	16.39± 1.40%	9	6.37± 4.84%	12
canny1x1	35.64± 1.69%	35	37.60± 1.30%	35	37.98± 1.40%	35	46.04± 1.73%	35	58.86± 1.48%	34	42.00± 8.50%	35
canny2x2	3.08± 0.63%	21	4.62± 0.70%	15	6.88± 1.02%	14	9.27± 0.94%	16	21.97± 1.22%	20	8.18± 6.63%	17
canny3x3	1.03± 0.31%	5	1.87± 0.51%	4	3.19± 0.65%	4	5.20± 0.67%	6	15.29± 1.06%	7	4.62± 5.05%	5
canny4x4	2.86± 0.52%	17	4.86± 0.65%	17	6.53± 0.70%	13	8.45± 0.74%	15	17.12± 1.04%	11	7.14± 4.96%	15
colorhist1x1	1.07± 0.37%	6	2.25± 0.50%	6	3.65± 0.63%	6	3.13± 0.62%	4	8.00± 0.94%	4	3.16± 2.43%	5
colorhist2x2	0.07± 0.10%	3	0.26± 0.21%	3	0.68± 0.37%	3	0.52± 0.24%	3	2.88± 0.64%	3	0.74± 1.03%	3
colorhist3x3	0.01± 0.04%	2	0.12± 0.14%	2	0.38± 0.26%	2	0.35± 0.23%	2	2.64± 0.55%	2	0.70± 1.02%	2
colorhist4x4	0.01± 0.04%	1	0.10± 0.12%	1	0.34± 0.21%	1	0.31± 0.20%	1	2.33± 0.46%	1	0.51± 0.85%	1

Table A.9: *Mean error rates and ranks for the different feature types on the COIL-100 data set (continued on the next page).*

Feature	1-NN		3-NN		5-NN		Fuzzy 3-NN		Fuzzy 5-NN		Mean	
	error	rank	error	rank	error	rank	error	rank	error	rank	error	rank
curvehist1x1	88.63± 1.13%	36	88.60± 1.06%	36	87.18± 1.25%	36	90.56± 0.94%	36	92.22± 0.96%	36	89.29± 1.95%	36
curvehist2x2	21.88± 1.45%	33	27.54± 1.54%	33	29.52± 1.41%	33	40.48± 1.42%	34	61.11± 1.60%	35	33.57± 13.81%	34
curvehist3x3	4.71 ± 0.74%	25	7.91 ± 0.85%	25	10.39± 0.90%	25	16.05± 1.34%	26	32.96± 1.41%	27	12.68± 9.86%	26
curvehist4x4	2.90 ± 0.60%	18	5.49 ± 0.79%	19	7.57 ± 0.81%	17	11.59± 1.20%	22	25.92± 1.52%	22	9.25 ± 8.00%	20
orientcurve-hist1x1	12.74± 1.09%	30	19.31± 1.56%	31	22.86± 1.20%	32	30.03± 1.33%	32	50.43± 1.64%	32	24.44± 13.04%	31
orientcurve-hist2x2	3.07 ± 0.58%	20	6.71 ± 0.74%	23	10.08± 0.84%	24	12.86± 1.02%	23	31.28± 1.56%	26	11.01± 9.69%	23
orientcurve-hist3x3	2.99 ± 0.64%	19	6.52 ± 0.78%	22	9.49 ± 1.01%	23	12.89± 1.10%	24	30.48± 1.44%	25	10.73± 9.47%	23
orientcurve-hist4x4	5.88 ± 0.88%	27	10.30± 0.90%	27	13.25± 0.99%	26	17.38± 1.25%	27	33.95± 1.81%	28	14.26± 9.67%	27
sobel1x1	12.77± 1.11%	31	16.84± 1.31%	30	20.10± 1.40%	29	20.86± 1.38%	29	34.67± 1.55%	30	19.71± 7.53%	30
sobel2x2	2.32 ± 0.52%	12	4.38 ± 0.63%	13	6.91 ± 0.77%	15	7.32 ± 0.76%	13	19.07± 1.19%	15	7.08 ± 5.79%	14
sobel3x3	2.49 ± 0.53%	14	4.48 ± 0.75%	14	7.00 ± 0.82%	16	7.48 ± 0.84%	14	17.78± 1.07%	13	6.98 ± 5.29%	14
sobel4x4	4.32 ± 0.58%	23	7.15 ± 0.85%	24	9.41 ± 0.85%	22	9.89 ± 1.05%	19	19.89± 1.15%	17	9.20 ± 5.35%	21
wavelet1x1	8.94 ± 0.95%	28	12.70± 1.03%	28	15.99± 1.14%	28	18.14± 1.05%	28	29.65± 1.45%	24	15.58± 7.17%	27
wavelet2x2	2.33 ± 0.58%	13	4.83 ± 0.81%	16	8.88 ± 0.92%	21	9.30 ± 0.92%	17	20.84± 1.31%	19	7.96 ± 6.41%	17
wavelet3x3	5.22 ± 0.72%	26	8.66 ± 0.96%	26	13.74± 1.26%	27	14.54± 1.09%	25	26.47± 1.56%	23	12.16± 7.41%	25
wavelet4x4	2.55 ± 0.54%	15	4.88 ± 0.63%	18	8.67 ± 0.81%	20	10.00± 0.98%	20	20.31± 1.28%	18	8.04 ± 6.18%	18

Table A.10: *Mean error rates and ranks for the different feature types on the COIL-100 data set.*