ENSEMBLE LEARNING FOR VISUAL RECOGNITION

by

Mohammad Ali Bagheri

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
July 2016

To the memory of my father (1931-2002) and my mother (1935-2015).

Table of Contents

List of Tables

List of Figures

Abstract

Due to the widespread availability of affordable mobile cameras and popularity of social networking, a large variety of visual-based applications are emerging. These diverse applications have attracted many researchers and high-profile companies around the world, and led to the development of a wide range of approaches for visual content analysis. In the last two decades, a large number of methodologies have been proposed for enhanced classification of images and videos.However, the potential improvement in visual data classification through classifier fusion by ensemble-based methods has remained relatively unstudied.

In this dissertation, we present new ensemble classification models and propose the employment of such models for some challenging visual recognition tasks, including both image classification and human action recognition.

First, we present a unifying framework for multiple classifier systems, which unites most classification methods by an ensemble of classifiers. Following this perspective, we propose a new general approach to ensemble classification, named generic subclass ensemble. Then, we focus on the subclass approach, specifically on Error Correcting Output Codes (ECOC). We propose a subspace approach to ECOC by defining a 3D-ECOC matrix, where the third dimension corresponds to the feature space. Also, Genetic Algorithm is employed for a highly discriminative design of an application-dependent subspace ECOC.

Then, the ensemble approach is utilized for action recognition using depth data; and present three new methods that improve the classification of depth-based action videos. First, we address two action recognition problems using skeleton data; and propose the use of an ensemble framework based on the Dempster-Shafer fusion of classifiers. In the second method, a new class of SVM that is applicable to trajectory classification, such as action recognition, is developed by incorporating two efficient time-series distances measures into the kernel function. The third method is based on the well-known Bag of Visual Words (BoVW) framework: we propose a new coding algorithm by jointly encoding the set of local descriptors of each sample and considering the locality structure of descriptors. In this model, two classifiers are trained using both skeleton and depth feature sets, and then combined. Experimental comparison of the proposed methodologies shows the superiority of our methods compared to the state-of-the-art.

List of Abbreviation Used

ADMM ..... Alternating Direction Method of Multipliers

BoVW ..... Bag of Visual Words

ECOC ..... Error Correcting Output Codes

DMM ..... Depth Motion Maps

DT ..... Decision Tree

DTW ..... Dynamic Time Warping

DS ..... Dempster Shafer

DWT ..... Discrete Wavelet Transform

kNN ..... k-Nearest Neighbor

LCSS ..... Longest Common Subsequence

GA ..... Genetic Algorithm

OVA ..... One-versus-All

OVO ..... One-versus-One

RSM ..... Random Subspace Method

ppfSVM ..... pairwise proximity function SVM

PSD ..... Positive Semi Definite

SC ..... Sparse Coding

SPM ..... Spatial Pyramid Matching

SVM ..... Support Vector Machine

Acknowledgements

I am grateful to many people that made this work possible. First, I would like to thank my supervisor, Dr. Qigang Gao. Since my educational background was in the field of Industrial Engineering, I had very slow progress in the first two years of my PhD program. Also, due to some family reasons, I had to come back to my hometown for about two years in the last five years. During all these years, his patient guidance and life-giving advice always enlightened my path and supported me to solve bottlenecks. I truly owe all my research achievements to him. I am also indebted that he gave me all the freedom and resources to develop my skills as a researcher. More importantly, I learned a lot from him on how to become a good researcher and a good man for my family.

I am also thankful to my co-supervisor, Dr. Sergio Escalera, for his kind help in various aspects of my research. His strong passion for scientific research and broad knowledge in different areas really helped me throughout my thesis. I never forgot his encouraging and supportive advice.

I would also like to thank the great people on my committee, Dr. Milios, Dr. Brooks, Dr. Reilly, and Dr. Zelek. I am also grateful to Dr. Milios and Dr. Thomas Trappenberg, for their kind welcoming in their research groups. I enjoyed many conversations and discussions that have had a tremendous impact on my research.

I learned a lot from former and current members of the IPAMI Lab. I want to specially thank Gang Hu and Elham Etemad for their kind help and wonderful research comments they provided. A number of wonderful friends have also helped me throughout this effort: Ali, Maliheh, Mahdi, Zahra, Ali and Maryam were my first friends at Dalhousie and I never forget them, although I have not seen them for a long time. I also thank my new friends at Dalhousie: Maryam, Reza, Babak, Vahid, Mohammad Hossein, and Ehsan.

Finally, I would like to thank Fatemeh for her love and support and for being there for me.

Chapter 1

Introduction

1.1 Motivation

Due to the widespread availability of affordable mobile cameras and popularity of social networking, a large variety of visual-based applications is increasingly emerging. These diverse applications have attracted many researchers and high-profile companies all around the world, and led to the development of a wide range of approaches for visual content analysis.

In the last two decades, a large number of methodologies have been proposed for enhanced classification of images and videos and the performances of different image/video representation and recognition algorithms have been studied by a number of researchers. However, the potential improvement in visual data classification through classifier fusion by ensemble-based methods has relatively remained unstudied.

This dissertation presents new ensemble classification models and proposes the employment of such models for a challenging visual application: human action recognition. An ensemble classification model, also known as a multiple classification system, is made up of a committee of individual classifiers whose outputs are combined in some way to obtain a final classification decision. In this model, it is hoped that each base classifier will focus on different aspects of the data and will err under different situations [123, 186]. By combining a set of base classifiers, the combined efficiency of the ensemble of classifiers can compensate for a deficiency in another classifier.

In this dissertation, we first present a unifying framework for multiple classifier systems, which unites most classification methods by an ensemble of classifiers. Specifically, we link two research lines in machine learning: multiclass classification based on the class binarization techniques and the strategies of ensemble classification. With the proposed framework, the various ensemble classification strategies are broadly categorized into four main groups, which are subsample, subspace, subclass,

and learner manipulation approaches. Following this perspective, we propose a new general approach to ensemble classification, named *generic subclass ensemble*. In this method, each base classifier is trained with data belonging to a subset of classes, and thus discriminates among a subset of target categories. The ensemble classifiers are then fused using a combination rule.

Then, we focus on the subclass approach, specifically on Error Correcting Output Codes (ECOC). Two key factors affecting the performance of ECOC are independence and accuracy of binary classifiers. Here, we propose a subspace approach to the ECOC framework by simultaneously considering the class samples and features of a given multiclass problem. Also, a Genetic Algorithm-based technique is employed for the optimal design of an application-dependent subspace ECOC. The proposed method takes advantage of some basic concepts of ensemble classification, such as diversity of classifiers. By taking into account the problem domain, this method also benefits from the evolutionary approach for optimizing the three-dimensional code-matrix. The experimental results using a set of UCI benchmark datasets as well as two image recognition problems, including logo recognition and shape categorization, show advanced performance of the method in terms of classification accuracy.

The ensemble model is then utilized for a visual classification application. More specifically, the recent release of the Microsoft Kinect camera in late 2011 [81] and an emerging wide range of applications inspired us to employ the ensemble-based models for classification of actions using depth cameras.

We present three new methods that improve the recognition of depth-based action videos. First, we address two action recognition problems using skeleton joint information. We argue that there is a potential improvement in classification through classifier fusion by ensemble-based methods. The underlying rationale of the fusion approach is two folds. First, different learners employ varying structures of input descriptors/features to be trained. These varying structures cannot be attached and used by a single learner. Second, in an ensemble classification system, the combined efficiency of multiple classifiers can compensate for a deficiency in another classifier. Thus, combining the outputs of several learners can reduce the risk of an unfortunate selection of a poorly performing learner. This leads to having a more robust and general-applicable framework. Motivated by this, we aim to employ different learners

and efficiently fuse them. To this end, we propose the use of the Dempster-Shafer fusion method to effectively combine the outputs of different learners, taking into account the characteristic of a given test action and the behavior of ensemble learners in similar cases.

In the context of action recognition using skeleton data, trajectories of skeleton joints are visual salient points of human body, and their movements in 4D space reflect motion semantics. From the classification point of view, these trajectories may be considered as multi-dimensional time series. The traditional recognition technique in the literature is based on time series dis(similarity) measures, such as Dynamic Time Warping. For these general dis(similarity) measures, k-nearest neighbor algorithms are a natural choice. In general, the k-NN classification algorithms work reasonably well, but are known to be sensitive to noise and outliers. Since SVM often outperforms k-NN on many practical classification problems where a natural choice of positive semidefinite (PSD) kernels exists, it is desirable to extend the applicability of kernel SVMs. In our action classification problem, time series distance measures are generally non-PSD kernels and basic SVM formulations are not directly applicable. To include non-PSD kernels in SVMs, several ad-hoc strategies have been proposed. The straightforward strategy is to simply overlook the fact that the kernel should be non-PSD. In this case, the existence of a Reproducing Kernel Hilbert Space is not guaranteed [142] and it is no longer clear what is going to be optimized. Another strategy is based on *pairwise proximity function* SVM (ppfSVM) [60]. This strategy involves the construction of a set of inputs such that each sample is represented with its dis(similarity) to all other samples in the dataset. In this dissertation, we investigate the effectiveness of this strategy for human action classification when the pairwise similarities are based on time-series distances measures. More specifically, we demonstrate the effectiveness of two trajectory-based distances measures - including Longest Common Subsequence (LCSS) and Dynamic Time Warping (DTW) as well as their derivatives - as SVM kernel functions. Four SVMs are trained with four different types of kernels and finally fused at the classification stage.

Based on the well-known Bag of Visual Words (BoVW) models, we also propose a new feature coding algorithm by jointly encoding the set of local descriptors of each sample and considering the locality structure of descriptors. In order to utilize

the information of both depth and skeleton data, we employed two different feature extraction methods: skeleton position and Depth Motion Maps (DMM). These two sets of features are finally fused at the classification level. To efficiently implement our proposed method, we consider the Alternating Direction Method of Multipliers (ADMM) framework, which results in quadratic complexity in the problem size. The proposed method is then employed for our action recognition problem by depth cameras. In this model, two classifiers are trained using both skeleton and depth feature sets. These classifiers are then combined in order to reach a higher level of prediction.

1.2 Research Contributions

Our contributions in this work can be summarized as follows:

- We present a unifying framework for multiple classifier systems that unites most classification methods by an ensemble of classifiers. The proposed framework is unique in a sense that it links two research lines in machine learning: multiclass classification based on the class binarization techniques and the strategies of ensemble classification.

- We present a new general approach to ensemble classification, named Generic Subclass Ensemble. The proposed approach differs from existing methods that manipulate the target attribute, since in our approach individual classification problems are not restricted to two-class problems. In light of this approach, class binarization techniques are considered special cases of the generic subclass ensemble approach.

- We propose an evolutionary algorithm-based approach to the design of an application-dependent codematrix in the ECOC framework, named GA-SS-ECOC. The proposed method aims to decompose the original problem into a set of binary problems, considering the class samples and features simultaneously. We also present the applications of GA-SS-ECOC in two machine vision domains: image and action recognition.

- We present an ensemble classification framework to address the action recognition problem. We designed a model that consists of a set of classifiers, each one

trained over different feature sets. The individual classifier outputs are then efficiently combined by means of the Dempster-Shafer fusion method, taking benefit from diversity of base classifiers trained on different sources of information.

We also introduce two fast action representation techniques, using only the skeleton joints' position during the time. The advantages of our methods are that: 1) They will generate a fixed size feature vector for an action, that may vary in time based on the action and subject that performs the action. Thus, they can be used as an input for any type of classifier, 2) The proposed representation techniques are relatively very fast. Thus, they are computationally very efficient for real time applications.

- A new class of Support Vector Machines that is applicable to trajectory classification, such as action recognition, is developed by incorporating two efficient time-series distance measures into the kernel function. Dynamic Time Warping and Longest Common Subsequence distance measures along with their derivatives are employed as the SVM kernel. In addition, the pairwise proximity learning strategy is utilized in order to make use of non-positive semi-definite kernels in the SVM formulation.

- A new encoding algorithm is proposed for human action recognition. The algorithm jointly encodes the set of local descriptors of each sample by considering the locality structure of descriptors. The proposed method takes advantage of locality coding such as its stability and robustness to noise in descriptors, as well as the strengths of the group coding strategy by taking into account the potential relation among descriptors of a sample. We also proposed the utilization of the Alternating Direction Method of Multipliers (ADMM) framework, which results in quadratic complexity in the problem size.

1.3 A Discussion on Deep Neural Networks for Action Classification

A deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers of units between the input and output layers, which can model complex

non-linear relationships. DNNs have dramatically improved the state-of-the-art in visual object recognition, object detection, speech recognition and many other domains. Deep learning discovers intricate structure in large data sets to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Two main categories of deep learning methods are convolutional neural networks (CNNs) and recurrent neural networks (RNNs). CNNs have brought about breakthroughs in processing images, video, speech and audio, whereas RNNs are a better choice for tasks that involve sequential inputs, such as speech and language. However, the important thing with deep learning is *"it needs a lot of data to train on, so a computer learns to do a task like recognizing an object in an image or identifying that there is a cancer cell or recognizing which word you're saying when you're speaking by looking at millions of examples, and one reason why neural nets didn't catch on earlier is that we didn't have that much data in the 90s"* [47].

In the field of action recognition by depth data, there is not any large publicly available dataset. This reason was the main bottleneck to approach this strategy for our problem. In addition, deep neural networks are generally more computationally intensive than other standard classification procedures, like bag of features + SVM.

1.4 Outline of the Thesis

The rest of this dissertation is organized as follows. In Chapter 2, we review the existing ensemble methods and present a unifying framework for multiple classifier systems. The proposed Generic Subclass Ensemble is then explained in Section 3. In Chapter 4, the GA-SS-ECOC is presented and the the results of experimental comparison over a set of benchmark datasets are provided.

In the next three chapters, the ensemble classification approach is employed for a visual and interesting application: human action recognition using depth data. We present three new methods that improve the recognition of depth-based action videos. First, in Chapter 5, we propose the use of an ensemble framework based on the Dempster-Shafer fusion of multiple classifiers. In Chapter 6, an ensemble system consisting of four SVMs with time-series distance kernels is employed for action classification. In chapter 7, a new encoding algorithm based on the BoVW

framework is proposed. In this model, two classifiers are trained using both skeleton and depth feature sets. These classifiers are then combined in order to reach a higher level of prediction.

Finally, in Chapter 8, we summarize our contributions, and discuss the future research directions.

Chapter 2

A Unifying Framework for Ensemble Classification

2.1 Introduction

The efficiency of pattern classification by a single classifier has been recently challenged by multiple classifier systems [88, 123, 127, 136, 186]. A multiple classifier system is a classification system made up of an ensemble of individual classifiers whose outputs are combined in some way to obtain a final classification decision. In an ensemble classification system, it is hoped that each base classifier will focus on different aspects of the data and will err under different situations [107, 123, 186]. By combining a set of base classifiers, the combined efficiency of the ensemble of classifiers can compensate for a deficiency in one classifier. However, the ensemble approach depends on the assumption that single classifiers' errors are uncorrelated, which is known as classifier *diversity* in the background literature [170]. The intuition is that if each classifier makes different errors, then the total errors can be reduced by an appropriate combination of these classifiers. In fact, the ensemble classification philosophy is derived from the fact that humans seek several opinions before making any critical decision. We weight the individuals' opinion and combine them to reach a final decision [123].

The design process of a multiple classifier system generally involves two main stages: the collection of an ensemble of classifiers and the design of the combination rule [62, 123]. The first stage is to construct an ensemble of accurate and diverse classifiers and the second phase aims to find a proper combination rule for the ensemble. Kuncheva summarized three primary approaches to creating an ensemble of classifiers [88]. These approaches can be considered as different ways to achieve diversity across base classifiers. The straightforward approach is to use different learning algorithms or variations of the parameters of the base learners. The second approach, which has been getting more attention in the ensemble literature, is to use different training sets to train base classifiers. Such sets are often obtained from the original training set by

resampling techniques. The third approach is to train the individual classifiers with datasets that consist of different feature subsets [69, 115]. The Random subspace method (RSM) proposed by Hu is the one early algorithm that builds an ensemble of diverse classifiers by randomly choosing feature subsets for each base learner [69].

Another effective approach, which has not been paid much attention in the ensemble literature, is to *"manipulate the target attribute"*, in which individual classifiers are built with different and usually simpler representations of the target classes [135, 136]. This approach was initially developed to solve the dilemma of extending binary classification algorithms to multiclass problems [130] and usually referred to as *"class binarization"* [54] in the multiclass classification literature. Existing methods based on this approach decompose the original multiclass problem into a series of smaller two-class problems. In this way, two-class problems can be solved by binary classifiers and their results can then be combined so as to provide a solution to the original problem. The procedure to decompose the multiclass problem into a set of binary problems is usually defined within the framework of Error Correcting Output Codes (ECOC) [37, 172].

As mentioned above, this approach has been ignored by many researchers in the ensemble classification literature. As an example, in a recent book by Zhou [186], the author indicates that: *"in a strict sense, they cannot be recognised as ensemble combination methods"* [186, p. 89]. More interestingly, some authors in the multiclass classification field segregate class binarization techniques from ensemble classification methods. As indicated in [54]: *"We must bear in mind that the fusion of binary classifiers is different from other similar tasks, such as the fusion of classifiers in an ensemble"*. There are likely two main reasons for this oversight. The first reason is that this approach was introduced to tackle the dilemma of extending binary classification algorithms to multiclass problems, and is therefore considered as a multiclass decomposition technique. Accordingly, this approach is only applicable to multiclass problems. The second reason is that standard ensemble methods differ from the class binarization approach in that in the former, each classifier attempts to solve the same problem, whereas in the class binarization approach each classifier solves a different sub-problem.

Here, we argue that class binarization techniques, including the ECOC framework,

can be seen as an ensemble classification system. There are two main reasons behind this argument. First, the philosophy of class binarization techniques is in accordance with the ensemble philosophy: multiple learners are generated to learn different aspects of the data and then fused to reach a final decision. Second, the cornerstone of ensemble methods and class binarization techniques are the same. In both methods, combining the outputs of individual classifiers would be effective if they make an error in different situations, usually called *diversity* in the ensemble literature and *independency* in the multiclass classification literature. The intuition is that if each classifier makes different errors, then the errors can still be effectively detected and possibly corrected.

Based on this holistic view, we present a unifying framework for multiple classifier systems that unites most classification methods using an ensemble of classifiers, including existing class binarization techniques such as ECOC. According to this proposed framework, we provide a brief survey of ensemble creation approaches as well as the principal techniques proposed to combine them.

2.2 The Unifying Ensemble Classification Framework

Here, we propose a conceptually unified framework for ensemble classification methods. In the framework, ensemble strategies are broadly categorized into four general approaches. Among them, three approaches generate ensemble classifiers by manipulating data, which we have named subsample, subspace, and subclass approaches. The fourth approach generates ensemble classifiers by manipulating learners, usually by using different learning algorithms or variations of the parameters of base learners.

The proposed framework is depicted in Figure 2.1, where approaches based on data manipulation techniques are shown in the inner circle. Three main data manipulation approaches are indicated by the three sectors in a solid blue color. Combinations of these three approaches are indicated by the crosshatched blue sectors between them. These approaches are surrounded by the learner manipulation approach, which implies that, using data manipulation approaches, ensemble diversity can be improved by utilizing learner manipulation techniques. For example, in the subspace approach, instead of training a set of identical base learners with different feature subsets, different classification algorithms can be employed to be trained, as in [96].

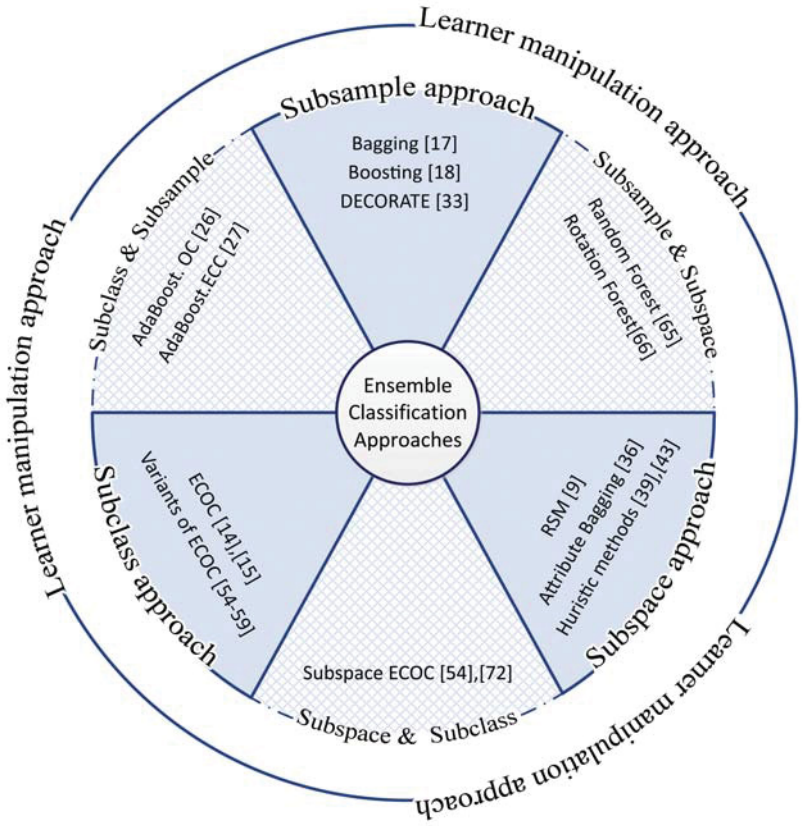


Figure 2.1: The proposed unifying ensemble classification framework.

In the following, we briefly present an overview of the primary ensemble creation approaches as well as the main combination methods based on these approaches.

2.3 Subsample Approach

The subsample approach, which has been getting more attention in the related literature, is to use different training samples to build base classifiers. Each sample subset is usually drawn from the whole training set and has the same features as the original data. These sample subsets may be disjointed or overlapping. Each subset is used to train an individual classifier, and the classifiers are finally combined in some way. This approach has been referred to, by some researchers, as the *"data level"* approach [88] or *"sample manipulating"* [136].

The main strategy to generate training subsets is based on the resampling techniques, where sample subsets are drawn randomly, usually with replacement, from the entire training data. Bagging [23] and boosting [49] are two of the most well-known

ensemble classification techniques based on this strategy. Bagging, an abbreviation for bootstrap aggregating, is one of the oldest and simplest ensemble methods, which shows surprisingly good performance in different applications [12]. In bagging, different sample subsets are randomly chosen, with replacement, from the all training samples. Since sampling with replacement is performed, some of the training samples may appear more than once in each subset and some may not be included at all. Individual classifiers are trained on each subset and then combined by the simple majority voting of their decisions. For a given test sample, the class chosen by most classifiers is the ensemble decision. Different algorithms have been proposed based on Breiman's bagging method. Among them, wagging [13] and pasting small votes [21] are the most well-known.

Boosting is similar to bagging in that individual classifiers are built by resampling training data and then are combined by majority voting. In the boosting strategy, however, classifiers are generated sequentially and samples for each classifier are selected based on the performance of previously built classifiers. In AdaBoost [49], a weight is assigned to each sample according to its classification difficulty. In the beginning, all weights are equal, but, in each iteration the samples' weights are updated such that the relative weights of samples misclassified by previous classifiers are effectively increased. Therefore, the algorithm increasingly focuses on difficult samples. This procedure generates a set of classifiers that complement one another. Many variations of AdaBoost have been proposed in the literature. Among them, Real AdaBoost [141], Gentle AdaBoost [50], Float Boost [94], and Reweight Boost [131] are some of the most important algorithms that are designed for binary classification problems and AdaBoost.M1 and AdaBoost.M2 [49], AdaBoost.OC [140], AdaBoost.ECC [61], AdaBoost.M1W [5], and AdaBoost. SAMME [187] are versions that address multiclass classification problems. In [51], a new method is presented by integrating the instance selection technique and boosting approach, where the instance selection technique search the subset of the training set by minimizing the training error to generate the next classifier. In [51], a new method is presented by integrating the instance selection technique and boosting approach, where the instance selection technique search the subset of the training set by minimizing the training error to generate the next classifier. Readers may refer to [48] for more details on

boosting strategies.

Another strategy to obtain subsets of data is based on the partitioning of training samples [118]. As a simple strategy, cross-validated committees, the training samples can be randomly partitioned into k-equal-sized disjoint subsets. Then, k classifiers are trained, in which the ith classifier uses all subsets, except the ith subset. Rokach [137] proposed an algorithm that partitions the input space into mutually exclusive subsets using the K-means clustering technique and a classifier is trained on each subset.

Most of the research above uses the original training samples to generate sample subsets. When the number of training samples is small, this limits the amount of ensemble diversity that these methods can obtain. Melville and Mooney [106] proposed a method, named DECORATE, which directly constructs diverse classifiers using additional artificially created training samples. Using C4.5 base classifiers, their results show that DECORATE achieves higher performance than Bagging and Random Forests.

2.4 Subspace Approach

The subspace approach, also called ensemble feature selection [115], trains the individual classifiers with datasets that consist of different feature subsets. The selected feature subsets can be either disjoint or overlapping. While traditional feature selection algorithms seek to find an optimal subset of features, the goal of ensemble feature selection is to find different feature subsets to generate accurate and diverse classifiers. The Random Subspace Method (RSM) proposed by Ho [69] is the one early algorithm that builds an ensemble of classifiers by randomly choosing feature subsets for each base learner. The experimental results show that while most classification approaches suffer from the curse of dimensionality, the subspace approach can benefit from high dimensionality phenomena. Ho showed that the subspace approach is expected to work better in classification problems which have many redundant features [69].

Different techniques based on this approach have been proposed in the related literature. Kuncheva categorizes techniques for choosing feature subsets into three main groups [88]:

Natural grouping: In some classification problems, especially in many real-world

applications, the original features are initially grouped, mainly because features are extracted from different sources. This strategy has been employed in many real-world applications [6, 11]. As an example, in [6], a logo recognition system using an ensemble of three classifiers is proposed. Individual classifiers are trained using different feature sets obtained from three shape description techniques and then fused by the Demspter-Shafer combination method.

Random selection: In this category, feature subsets are selected by a random or a pseudo-random strategy. Ho's RSM is the one early algorithm that randomly chooses about half of the available features for each classifier [69]. Bryll et al. presented a method, named Attribute Bagging, which first chooses an appropriate size of the feature subset using the wrapper feature selection approach and then randomly selects feature subsets [25].

Nonrandom selection: Most ensemble feature selection methods fall in this category. Representative works include [58, 72, 89, 115, 153, 155]. These methods aim to simultaneously improve the accuracy and diversity of base classifiers by choosing optimal features for each classifier that enhance the performance of the entire ensemble system. Various heuristic search techniques, such as genetic algorithms (GA), particle swarm optimization (PSO), and ant colony optimization (ACO) have been employed for feature subset selection [58, 89, 115, 134, 139].

2.5 Subclass Approach

The subclass approach is to train classifiers with samples belonging to different target classes [135]. Current methods based on this approach mainly deal with multiclass classification problems by decomposing the original problem into a series of smaller binary classification problems. The most well-known techniques based on this approach are one-versus-all (OVA) [4], one-versus-one (OVO) [66, 5], and Error Correcting Output Codes (ECOC) [37, 172, 119]. In one-versus-all, the multiclass problem is decomposed into several binary problems in the following way: for each class a binary classifier is trained to discriminate among the patterns of the class and the patterns of the remaining classes. In the one-versus-one technique, one classifier is trained to separate each possible pair of classes. In both approaches, the final classification prediction is usually obtained by means of a voting or committee

procedure. The ECOC framework decomposes a multiclass problem into a series of different binary problems [2, 37]. In this framework, each classifier is trained on a two meta-class problem, where each meta-class consists of some combinations of the original classes. Accordingly, classical OVA and OVO frameworks can be considered as special cases of the ECOC framework.

The ECOC method can be broken down into two stages: encoding and decoding. The aim of the encoding stage is to design a discrete decomposition matrix (codematrix) for the given problem. According to the codematrix, a set of binary classifiers are trained. Each row of the codematrix, named *codeword*, is a sequence of bits representing each class, where each bit identifies the membership of the class to a classifier [44]. In the decoding stage, the final classification decision is obtained based on the outputs of binary classifiers. Given an unlabeled test sample, each binary classifier casts a vote for one of the two meta-classes used in its training. The output vector is compared to each class codeword of the matrix and the test sample is assigned to the class whose codeword is closest to the output vector, according to a distance measure. Because of the ability of the ECOC framework to correct the bias and variance errors of the base classifiers [85, 188], it has been successfully applied to a wide range of applications [1, 117, 82]. In recent years, many authors tried to improve the efficiency of the ECOC framework, mainly by taking into account the characteristics of the problem at hand [9, 45, 67, 124, 183, 184].

2.6 Learner Manipulation Approach

One straightforward approach to create diverse classifiers is to manipulate the base learners used for creating the ensemble. Several strategies have been employed to achieve diversity based on this approach. One strategy is to use different classification algorithms for the base learners. This approach is mainly utilized in applied problems. The other common strategy is varying the parameters of the base classifiers; e.g. different initial weights [84] or different topologies [114] of a series of neural network classifiers. In [29], the authors employed an ensemble of nine classifiers with three different learning algorithms, namely decision tree, back-propagation network, and support vector machine. They applied the scatter search-based method to search for the optimal parameter settings for each classifier, and then combined the classifiers

by the majority voting rule. In [63], a neural network ensemble was designed, in which individual networks were trained with different objective functions. Islam et al. proposed an algorithm which simultaneously determines the ensemble size along with the number of hidden nodes [75].

2.7 Combination Based Approaches

Combination of subsample and subspace approaches

Several methods have been proposed in the ensemble classifier literature to simultaneously manipulate samples and the feature space. Breiman's Random Forest [22] integrates the merits of both bagging and random subspace in a way that is specific to using decision trees as the base learner. In this method, the bagging procedure is utilized to generate a training subset for each individual tree. However, features used to split a node in the tree are randomly selected. Rodriguez et al. [132] proposed Rotation Forest, which first randomly splits feature sets into K subsets and applies principal component analysis (PCA) on each subset of a bootstrap replicate. Therefore, K axis rotations take place to form the new features for base classifiers. More recently, a method based on the combination of Rotation Forest and the AdaBoost, named RotBoost [181], has been proposed. Zhou and Yu proposed a method for building an ensemble of local learners, such as nearest-neighbor classifiers, by multimodal perturbation on features, samples and base classifiers' parameters [182]. Tao et. al [150] proposed a method based on the combination of bagging-based SVM and random subspace SVM in order to improve the relevance feedback in content based image retrieval.

In a series of works, some similar algorithms that combine the philosophy of boosting with the basis of the random subspace method were proposed [52, 53]. In [52], a method is presented that chooses a subset of features for the current classifier by considering samples that are previously misclassified. In this method, each classifier uses all training samples for learning, but uses a different nonlinear projection of the data in order to better classify difficult samples. In [53], the authors proposed a very similar method that combined RSM and boosting, named Not so Random Subspace Method (NsRSM). In this method, instead of choosing random features, they select features that optimize the weighted classification error given by the boosting

algorithm, and then the new classifier is trained with the selected features.

Combination of subsample and subclass approaches

As mentioned earlier, the most representative subclass-based method is that of ECOC. Some previous studies have proposed the use of bagging and boosting within the ECOC framework, mainly by selecting a sampling of data for each dichotomizer in order to increase the diversity of binary problems. In this sense, Schapire proposed a new technique, named Adaboost.OC, by combining the boosting algorithm with output codes [140]. After that, Guruswami and Sahai, proposed a variant of Adaboost.OC that uses different weighting of the votes of the weak learners, named Adaboost.ECC [61].

Combination of subspace and subclass approaches

Based on integrating the idea of the subspace approach and the ECOC framework, we have proposed a new method, named subspace ECOC [7, 9]. The central idea of this method is using feature space in the design process of the ECOC matrix. That is, each binary classifier (dichotomizer) is trained with a different feature subset, leading to better classification accuracy and diversity among binary classifiers. More recently, the proposed method is extended by optimizing the whole ensemble classification system using the GA algorithm by considering the characteristics of the problem at hand [9].

Combination of data manipulation approaches and learner manipulation

There are relatively few theoretical studies that integrate the strategy of data manipulation approaches with the learners' manipulation approach. The main idea behind these works is to improve the diversity of ensemble classifiers by the simultaneous use of different base learners. In [96], an ensemble of six heterogeneous classifier is generated, in which each classifier is trained with different feature sets and different classification algorithms. Zhou and Yu proposed a method for building ensembles of local learners, such as nearest-neighbor classifiers, by multimodal perturbation on features, samples and base classifiers' parameters [182]. More specifically, their method employs data perturbation with bootstrap sampling, feature perturbation with attribute filtering and subspace selection, and learning perturbation with randomly configured distance metrics. More recently, bagging models composed of heterogeneous learners is proposed in [32].

2.8 Summary

In this chapter, first, we presented a unifying framework for multiple classifier systems that conceptually unites a large variety of ensemble classification methods, including existing class binarization techniques such as ECOC. In the framework, various ensemble methods are broadly categorized into four general approaches. Among them, three approaches generate ensemble classifiers by manipulating data, which we have named subsample, subspace, and subclass approaches. The fourth approach, learner manipulation, is usually based on using different learning algorithms or variations of the parameters of base learners. According to this proposed framework, we provided a brief survey of ensemble creation methods as well as the principal techniques proposed to combine them.

Chapter 3

Generic Subclass Ensemble Classification

3.1 Introduction

As mentioned before, one effective approach in the ensemble literature is to "manipulate the target attribute", in which individual classifiers are built with different and usually simpler representations of the target classes [135, 136]. This approach was initially developed to solve the dilemma of extending binary classification algorithms to multiclass problems and usually referred to as *"class binarization"* [54] in the multiclass classification literature. Existing methods based on this approach decompose the original multiclass problem into a series of smaller two-class problems. In this way, two-class problems can be solved by binary classifiers and their results can then be combined so as to provide a solution to the original problem.

The main drawback of the class binarization approach is its limitation to two-class problems. In this chapter, we propose a new general approach to ensemble classification, named *generic subclass ensemble*, in which each base classifier is trained with data belonging to a subset of classes, and thus discriminates among a subset of target categories. The number of categories in each subset may vary between two and the number of classes in the problem. The ensemble classifiers are then fused using a combination rule. The proposed approach differs from existing methods that manipulate the target attribute, since in our approach individual classification problems are not restricted to two-class problems. In light of this approach, class binarization techniques are considered special cases of the generic subclass ensemble approach. We also perform a series of experiments to evaluate the efficiency of the subclass approach on a set of benchmark datasets.

Table 3.1: An example of a class decomposition matrix in the generic subclass approach.

	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}
ω_1	1	0	1	0	0	1	0	0	1	0
ω_2	1	0	0	1	0	2	1	2	1	0
ω_3	0	1	2	0	0	3	0	1	2	1
ω_4	2	2	3	0	1	0	2	0	1	1
ω_5	0	3	4	2	2	0	0	1	4	0
ω_6	0	3	2	0	1	0	2	1	5	2

3.2 The Generic Subclass Approach

In this section, we first introduce the generic subclass approach and explain its training and testing phases. Then, we propose three methods based on this approach.

3.2.1 Training phase

In the training phase, different sub-problems are generated and a base classifier is trained on each sub-problem using samples belonging to a subset of the original set of classes. For each sub-problem, the subset of classes is divided into two or more meta-classes, where each meta-class consists of some combinations of the original classes. Accordingly, each classifier discriminates among classes that have been seen in its training. Here, similar to the ECOC framework, the process of decomposing the multiclass problem into a set of smaller binary /multiclass problems is represented by a matrix. This matrix is interpreted as a set of L learning problems, one for each column. Table 3.1 shows an example of a class decomposition matrix for a six-class problem that uses 10 classifiers. In this matrix, each column is associated with a subclass classifier, h_j, and each row is a unique codeword that is associated with an individual target class. For example, the second classifier, h_2, discriminates among samples of four classes: $\omega_3, \omega_4, \omega_5$ and ω_6. These classes are split into three meta-classes: $\{\omega_3\}, \{\omega_4\}$, and $\{\omega_5, \omega_6\}$. Similar to the sparse ECOC matrix [2], the zero value means that a given class is not considered in the training phase of a particular classifier.

3.2.2 Testing phase

When testing an unlabeled pattern, x^*, each classifier casts a vote for one of the classes used in its training, creating an L long output code vector. This output vector

is compared to each class codeword in the matrix, and the class whose codeword has the closest distance to the output vector is chosen as the predicted class. Similar to the ECOC method, the process of merging the outputs of individual classifiers is called decoding [42]. Here, we propose a simple decoding technique:

$$y = \arg\min \frac{\sum_{i=1}^{L} w_i * sign(M(r,i)) * (M(r,i) <> h_i(x))}{sum(M(r,i) <> 0)},$$

where $sign(z)$ is +1 if $z > 0$, -1 if $z < 0$ and 0 otherwise and $a <> b$ is 1 if $a \neq b$ and 0 otherwise. Here, w_i represents each classifier' weight, which is set to the accuracy of classifier on the training or validation samples. $M(r,.)$ designates the codeword r in the matrix, $h_i(x)$ represents the output of classifier i, and $y \in \{1, \ldots, N_c\}$ is the predicted class.

3.3 Three Methods Based on the Generic Subclass Approach

Utilizing the subclass approach will pose an important challenge: how to decompose the original multiclass problem into smaller problems? For a given problem of c classes, the number of valid partitions is $B_n - 1$, where B_n is the total number of partitions of a set with n members, named the Bell number.

Here, we propose three decomposition techniques: a problem-independent technique based on the exhaustive decomposition and two problem-dependent techniques based on the partitioning of the class space.

3.3.1 Exhaustive decomposition

One straightforward technique for class decomposition is to consider all possible partitions of classes, except the one that puts all classes in one partition. For problems with a large number of classes, however, the number of partitions and the number of selected classes for each sub-problem become very large. In order to limit the computational complexity, we employ two strategies to reduce the length of the code-matrix, i.e. the number of classifiers. The first strategy is to have only one class in each meta-class. Second, we limit the number of classes to be taken for each classifier. Based on preliminary sets of experiments, choosing from two up to four classes for each sub-problem leads to high classification results. As an example, Table 3.2

Table 3.2: An example of an exhaustive coding matrix for a six class problem.

	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}	h_{16}	h_{17}	h_{18}	h_{19}	h_{20}
ω_1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
ω_2	2	2	2	2	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
ω_3	3	0	0	0	2	2	2	0	0	0	2	2	2	0	0	0	1	1	1	0
ω_4	0	3	0	0	3	0	0	2	2	0	3	0	0	2	2	0	2	2	0	1
ω_5	0	0	3	0	0	3	0	3	0	2	0	3	0	3	0	2	3	0	2	2
ω_6	0	0	0	3	0	0	3	0	3	3	0	0	3	0	3	3	0	3	3	3

Table 3.3: An example of an equidistant coding matrix for a six class problem.

	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}
ω_1	1	1	1	1	1	0	0	0	0	0
ω_2	2	2	0	0	0	1	1	1	0	0
ω_3	3	0	2	0	0	2	0	0	1	1
ω_4	0	3	0	2	0	0	2	0	2	2
ω_5	0	0	0	3	2	3	0	2	3	0
ω_6	0	0	3	0	3	0	3	3	0	3

shows an exhaustive decomposition matrix for a six-class problem that considers all permutations of three out of six classes.

Still, for problems with a large number of classes, the number of possible permutations of three or four classes out of the number of classes increases dramatically. As a result, a large number of classifiers is required in the training and testing phases. In these cases, we propose a new version of exhaustive coding, named equidistant coding, where the distances between all pairs of class codewords are equal [1]. In this version, a subset of all permutations is chosen such that the sum of the number of each pair of classes is identical. As an example, Table 3.3 shows an equidistant coding for a six-class problem, in which the number of each pair of classes in all sub problems is 2. Thus, to discriminate between each pair of classes, two classifiers exist in the ensemble system.

3.3.2 Decomposition based on a hierarchical clustering of class space

In this strategy, the class decomposition process is performed based on a clustering of the class space that maximizes class discrimination in the patterns. Using a clustering technique, classes within each cluster are generally more similar which make them more difficult to be discriminated by a classifier. In this work, generating class subsets are guided through the hierarchical clustering of classes. The hierarchical clustering

[1] distance is defined as the number of corresponding bits that differ.

algorithm groups classes by creating a cluster tree or dendrogram. The dendrogram is not a single set of clusters, but rather a multilevel hierarchy, where clusters at one level are merged in clusters at the next level. Each node of the cluster tree defines a partition of the classes. The partition at each node should be highly separable in terms of discrimination.

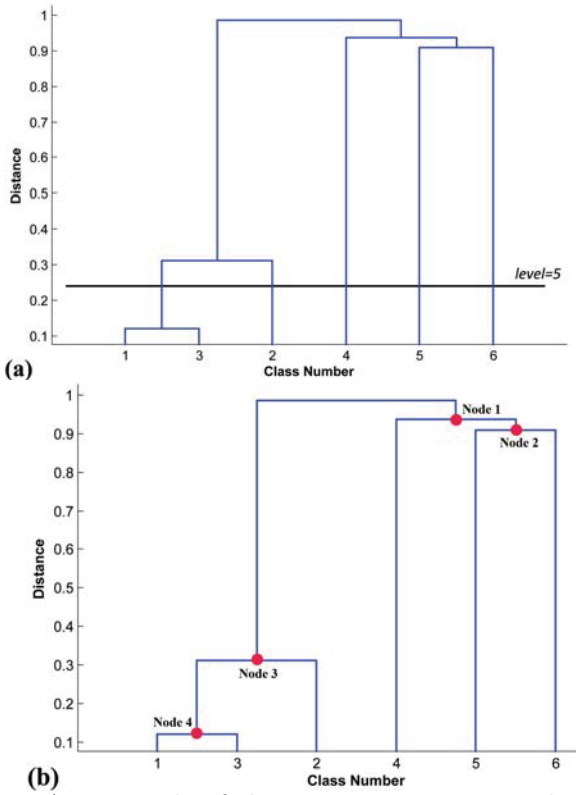
Using the generated cluster tree, the procedure of class partitioning, i.e. generating the class decomposition matrix, will be performed and the decomposition matrix will be generated. The procedure is composed of two main modules. Each module creates one decomposition matrix, and the final matrix is made by concatenating these two matrices.

In the first module, class partitioning is performed at different clustering levels ($1 < level < 6$). At each level, all classes are grouped into different partitions (clusters). An example of a clustering tree for a six class problem as well as the corresponding decomposition matrix is shown in Figure 3.1.a. For instance, consider the horizontal line at the fifth level. This line intersects five lines of the dendrogram. These five lines partition the classes into five clusters: $\Psi_5^1 = \{c_1, c_3\}, \Psi_5^2 = \{c_2\}, \Psi_5^3 = \{c_4\}, \Psi_5^4 = \{c_5\}$, and $\Psi_5^5 = \{c_6\}$.

The second module partitions classes under each internal node into two clusters. As an example, Figure 3.1.b shows four internal nodes for a six-class problem. For example, the first node partitions classes below the right-hand side, namely c_5 and c_6, belonging to one cluster, while the class below the left-hand line, namely c_4, belongs to the other cluster. This partitioning is represented in the first column of the matrix.

3.3.3 Two methods based on hierarchical class partitioning

The clustering procedure involves defining a dissimilarity measure between objects in order to optimize the within- and between-cluster structure. Here, we employed two measures that aim to satisfy the condition of the high separability of classes: 1) the distance between the centroid of classes; and 2) the mutual information. Based on these two measures, two different methods are proposed. In the first method, the centroid of each class is computed as the average of patterns belonging to the corresponding class. The distance between two class centroids is a rough estimation of how separate two classes are. These centroid patterns are then hierarchically clustered.



	h_1	h_2	h_3	h_4	h_5
ω_1	1	1	1	1	1
ω_2	1	1	1	2	2
ω_3	1	1	1	1	3
ω_4	2	2	2	3	4
ω_5	2	3	3	4	5
ω_6	2	3	4	5	6

	h_1	h_2	h_3	h_4
ω_1	0	0	1	1
ω_2	0	0	2	0
ω_3	0	0	1	2
ω_4	1	0	0	0
ω_5	2	1	0	0
ω_6	2	2	0	0

Figure 3.1: An example of class partitioning using the hierarchical clustering technique; (a) class partitioning at different clustering level; (b). Class partitioning under each internal node.

In the second method, the class partitioning is based on the mutual information, which has been shown to be an effective criterion in terms of class separation. For more details on computing the mutual information between classes, please refer to [124, 151].

3.4 Experiments

In this section, we evaluate the efficiency of the generic subclass ensemble on a variety of different real classification problems. First, we discuss the settings of the experiments including the data, the comparative methods, and the evaluation measurements.

3.4.1 Experimental settings

Data: The proposed generic subclass approach is evaluated on 24 multiclass datasets from the UCI machine learning repository [16]. Table 3.4 shows the number of classes,

instances, and features of each dataset.

Table 3.4: Summary of the datasets used.

Dataset	# Samples	# Classes	# Features
Abalone	4177	3	8
Car	1728	4	6
Cmc	1473	3	9
Derm	358	6	34
Glass	214	6	9
Iris	150	3	4
Mfeat-fou	2000	10	76
Mfeat-pix	2000	10	240
Mfeat-zer	2000	10	47
Optdigits	5620	10	62
Pendigits	10992	10	16
Plant Leaves (10c)	160	10	63
RobotLP2	48	5	90
Sat	6435	6	36
Semeion	1593	10	256
Thyroid	215	3	5
Vehicle	846	3	18
Vowel	528	11	10
Waveforms	5000	3	40
Wine	178	3	13
WineQualityRed	1599	6	11
WineQualityWhite	4898	7	11
Yeast	1484	10	8
Zoo	101	7	16

Methods: We compare our proposed method with well-known ensemble classification methods, including bagging, AdaBoost, and RSM. For AdaBoost, we implemented the AdaBoost.M1 algorithm [49] which is a stable version of boosting for multiclass classification problems. For RSM implementation, as suggested by Ho [69], about half of the features are selected for each base classifier. The ensemble size, i.e. the number of base classifiers of the bagging and RSM and the number of iterations of the AdaBoost algorithm, is set to 25 [13, 113].

In this study, a multilayer perceptron (MLP) with 10 hidden nodes and the hyperbolic tangent transfer function is chosen as the base learner. The MLP classifier cannot handle the missing values, so the instances with missing values are removed.

Evaluation measurements: The classification accuracy is obtained by means of stratified 10-fold cross-validation to improve the reliability of the results. Moreover, we include statistical tests to look for statistical significance among the obtained methods.

Table 3.5: Classification accuracies of different ensemble methods using MLP neural network as the base learner.

Datasets	**Single**	**Bagging**	**RSM**	**AdaBoost**	**Subclass.v1**	**Subclass.v2**	**Subclass.v3**
Abalone	66.45	66.83	65.85	66.75	**67.17**	66.92	66.91
Car	93.93	93.64	90.33	94.14	95.66	95.18	**96.53**
Cmc	52.64	54.76	53.99	54.29	**55.06**	54.36	54.29
Derm	95.07	96.65	**97.11**	93.10	96.46	96.69	95.99
Glass	56.50	63.99	65.39	62.60	**67.56**	67.27	66.93
Iris	94.22	95.11	94.67	93.56	95.56	**95.78**	95.56
Mfeat-fou	77.05	80.42	80.72	75.20	**83.85**	82.02	80.18
Mfeat-pix	80.33	93.00	95.50	94.58	**97.42**	95.33	92.75
Mfeat-zer	80.08	84.00	83.25	79.50	84.25	**84.42**	81.83
Optdigits	93.30	96.20	95.94	94.34	**98.10**	96.41	96.12
Pendigits	92.04	95.89	91.57	95.09	**99.44**	97.98	99.12
Plant leaves (10c)	60.42	95.83	97.92	95.50	**98.96**	96.88	**98.96**
RobotLP2	42.89	49.11	58.67	55.33	58.00	62.00	**60.67**
Sat	86.27	87.24	87.95	86.57	**89.24**	88.31	89.07
Semeion	57.26	70.64	89.03	75.92	**91.64**	85.27	79.94
Thyroid	95.23	96.14	95.93	95.55	**97.09**	96.19	96.29
Vehicle	80.54	81.52	80.97	79.99	84.08	83.85	**84.43**
Vowel	65.84	79.29	75.12	72.27	92.80	91.28	**96.15**
Waveforms	85.97	86.23	86.39	84.80	**86.49**	86.15	85.73
Wine	95.37	99.07	**100.00**	96.30	97.22	97.22	97.22
WineQualityRed	59.06	62.71	59.48	60.73	63.02	**63.96**	61.67
WineQualityWhite	53.27	54.59	53.37	54.63	55.41	**55.44**	55.20
Yeast	55.89	57.46	50.73	56.68	**58.14**	57.13	56.90
Zoo	90.48	96.83	95.24	94.13	96.83	96.83	**98.41**
Average	**73.95**	**80.47**	**80.99**	**79.55**	**84.15**	**83.30**	**83.02**

3.4.2 Experimental results

The average accuracy of the standard ensemble methods for each dataset is presented in Table 3.5. For reference, we also show the accuracy of a single MLP classifier. In this table, the means of prediction accuracy over 10 runs (expressed in %) is reported for each method on the considered dataset. For each dataset, the best accuracy achieved among all tested methods is in bold. As stated earlier, we proposed three different methods based on the subclass approach. The one based on the equidistant coding, named *Subclass v.1*, and two methods based on the partitioning of the class space using two measures, the distance between class centroids and the mutual information, which we respectively named as *Subclass v.2* and *Subclass v.3*.

Statistical analysis of the classification results

In order to compare the results of the different approaches, statistical analysis is applied. According to the recommendations of Demsar [36], we consider the use of non-parametric tests. Non-parametric tests are safer than parametric tests, since they do not assume normal distribution or homogeneity of variance. In this study, we employ two types of analysis. First, we use the Iman-Davenport test. If there are statistically significant differences in the classification performance, then we can proceed with the Nemenyi test [108] as a post-hoc test, which is used to compare the methods with each other. Second, the win-tie-loss comparison of the methods using the Wilcoxon signed rank test is performed.

□ **Statistical analysis using the Iman-Davenport and the Nemenyi tests:** For the first set of analysis, we first rank competing methods for each dataset. The best performing method gets rank 1, the second best is ranked 2, etc. The method's mean rank is obtained by averaging its ranks across all datasets. Then, we use the Friedman test [36] to compare these mean ranks to decide whether to reject the null hypothesis, which states that all considered methods have the equivalent performance. The Friedman statistic value is computed as follows:

$$\chi_f^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right], \tag{3.1}$$

where k is the number of competing methods and N is the number of experiments. In our case, when comparing five methods in 24 experiments, $\chi_f^2 = 58.16$. Iman and Davenport [74] found that this statistic is undesirably conservative, and proposed a revised one:

$$F_F = \frac{(N-1)\chi_f^2}{N(k-1) - \chi_f^2}, \tag{3.2}$$

which is distributed following an F distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. By applying this correction we obtained $F_F = 21.63$. The critical value of F(5,23) for $\alpha = 0.05$ is 2.64. As the value of F_F is higher than 2.64 we can reject the null hypothesis, that is, using the Iman-Davenport test, there is significant differences between rival methods.

Further, to compare rival methods with each other, we applied the Nemenyi test.

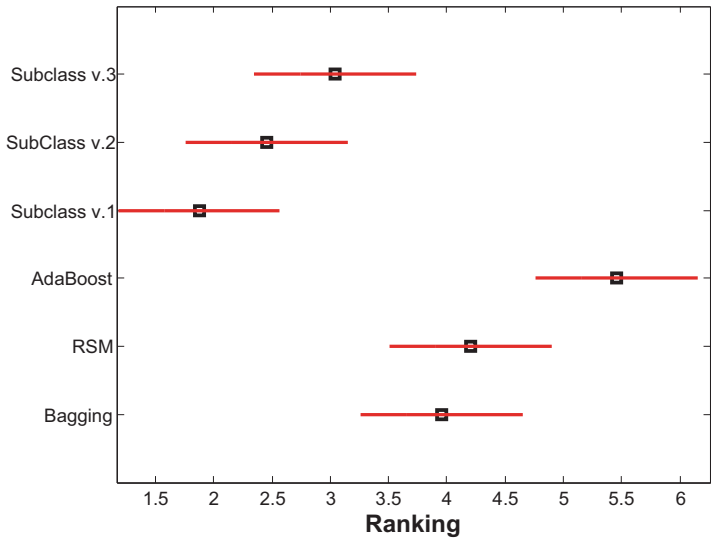


Figure 3.2: Statistical comparison of the results using the Nemenyi test.

Two methods are significantly different if their corresponding average ranks differ by at least the critical difference value (CD):

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{3.3}$$

In our case, the critical value for a 90% of confidence is $CD = 1.39$. The results of the Nemenyi test are illustrated in Figure 3.2. In this figure, the mean rank of each method is indicated by a square. The horizontal bar across each square shows the critical difference. Two methods are significantly different if their horizontal bars do not overlap.

As can be seen in the figure, the first and the second versions of the proposed generic subclass ensemble are significantly superior to all other methods. This finding implies that the proposed approach presents a viable alternative to the most commonly used ensemble classification approaches.

□ **Statistical analysis using the Wilcoxon sign rank test:** Table 3.6 shows the comparison of the different method using the Wilcoxon test. In this table, we show the win-loss-tie (WLT) comparison record of the algorithm in the column against the algorithm in the row. This record presents the number of datasets in which the algorithm in the column was better than the algorithm in the row (win), was worse (loss), or was equal (tie).

Table 3.6: Statistical comparison of the different ensemble methods using the Wilcoxon sign rank test.

	RSM	AdaBoost	Subclass v.1	Subclass v.2	Subclass v.3
Bagging	9/5/10	15/0/9	0/14/10	1/11/12	3/8/13
RSM		11/5/8	1/16/7	3/14/7	5/10/9
AdaBoost			0/20/4	0/18/6	0/17/7
Subclass v.1				11/1/12	10/1/13
Subclass v.2					6/5/13

The results in Table 3.5 and Table 3.6 along with the statistical test presented in Figure 3.2, indicate that overall the generic subclass approach obtained the best results with many datasets. Also, comparing the three versions of the generic subclass approach shows that, in terms of the classification accuracy, the subclass ensemble based on the equidistance decomposition achieves better overall results. The main reason behind this improvement might be the fact that using the equidistant coding more classifiers are usually generated. In ensemble systems, larger numbers of classifiers, especially when non-deterministic classifiers like neural networks are used as the base learner, usually lead to better classification accuracy. Similarly, the results of ECOC studies show that the ECOC method with longer codes is able to significantly improve the accuracy [2, 54]. In addition, in the second and third versions of the subclass approach, the class partitioning is performed by the hierarchical partition of classes that maximizes a discriminative criterion, i.e. the distance between class centroids in the second version and mutual information in the third version. However, there is no guarantee that this partitioning fits the underlying distribution of data. Therefore, the errors from a classifier at higher level are propagated to the lower levels.

As an additional analysis, the improved accuracies of the Subclass.Equidistant method in comparison with other methods are shown in Figure 3.3. In this figure, the obtained results are presented in the order of the number of dataset classes. The results of datasets with a larger number of classes are shown in order from the left side of the figure. From this arrangement, it can be seen that the generic subclass ensemble works better when there is a larger number of classes. In these cases, instead of combining individual classifiers trained with different subsets of samples or features, the more efficient approach is to train classifiers on a subset of classes. In this way, the problem that each classifier is going to be applied to is relatively smaller and can

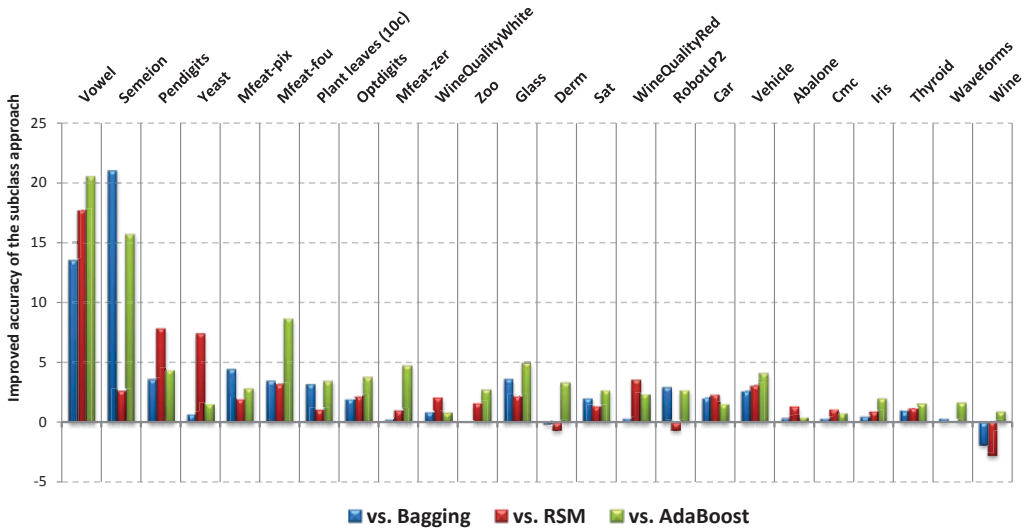


Figure 3.3: The relative accuracy of different ensemble methods compared to the *Subclass.Equidistant* method.

be solved more efficiently. On the other hand, when the number of features is high (in relative to the number of samples), the subspace approach seems to be a good choice. As an example, with the Semeion and Mfeat-pix datasets, which have a relatively large number of features, the subspace approach achieves very high performance.

3.4.3 Analyzing the effect of ensemble size

Here, we investigate the performance of rival ensemble methods with different ensemble sizes; $1 \leq Ens.Size \leq 50$. For the generic subclass approach, however, the initial number of required classifiers for a given problem is fixed; which is equivalent to the number of columns of the class decomposition matrix. Therefore, to evaluate the performance of the ensemble system all classifiers should cast a vote.

In the case of the Subclass.Equidistant method, for problems with a large number of classes, the number of classifiers may be very large. In these cases, as we mentioned earlier, we limit the number of considered classes for each classifier to two or three. That is, each classifier discriminates between samples of two or three classes. For this design, the class decomposition matrix begins with all permutations of two classes, like the one-versus-one technique, and then continues until the number of classifiers is less than a pre-defined number, fixed at 50 in our experiments. On the other hand, for problems with small numbers of classes, the initial ensemble size is relatively small.

One strategy to increase the number of classifiers is to duplicate the decomposition matrix. In this way, the ensemble size is the multiple of the initial number of classifiers, i.e. the length of codewords of the original class decomposition matrix. Even though the same sub-problems will be produced, the larger ensemble system can benefit from the variation of non-deterministic classifiers like neural networks.

Figure 3.4 shows the classification accuracy of rival methods as a function of the ensemble size for 16 representative datasets. From these results, some general findings are summarized below:

- These experiments indicate that, in general, ensemble methods follow a similar trend. That is, their classification performance first improves as the ensemble size increases and then plateaus after a demarcation point, e.g. a value around 15-25. This observation is consistent with the results of many studies [106, 113, 146].

- The underperformance of the Subclass.Equidistant method in problems with a small number of classes is mainly due to the significantly smaller number of classifiers. However, by increasing the ensemble size by duplicating the class decomposition matrix, classification accuracy was improved for many datasets.

- In problems with a larger number of classes, bagging, boosting, and RSM ensemble methods cannot continue to further improve with larger ensemble sizes. In these cases, the subclass approach shows the best performance.

3.5 Summary

In this chapter, we proposed a new general approach to ensemble classification, named generic subclass ensemble. In this approach, an ensemble of classifiers is generated, in which each base classifier aims to discriminate between a subset of target categories. The proposed approach is a general framework that incorporates many methods based on class binarization techniques.

Based on the generic subclass approach, three methods are introduced: Subclass.Equidistant, Subclass.ClsPart_MI, and Subclass.ClsPart_Dist. Using the neural

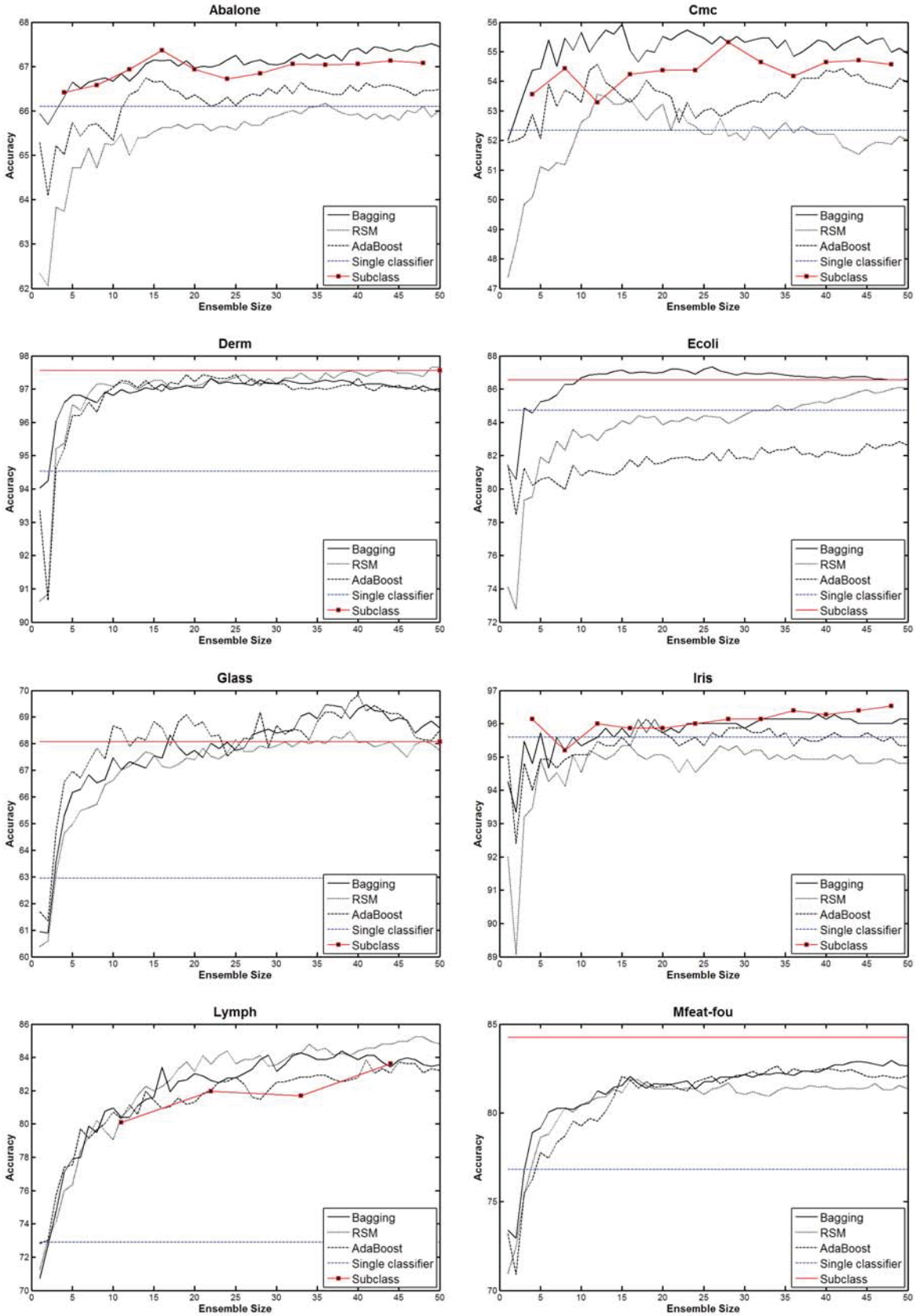


Figure 3.4: Accuracy of ensemble classification methods versus the ensemble size.

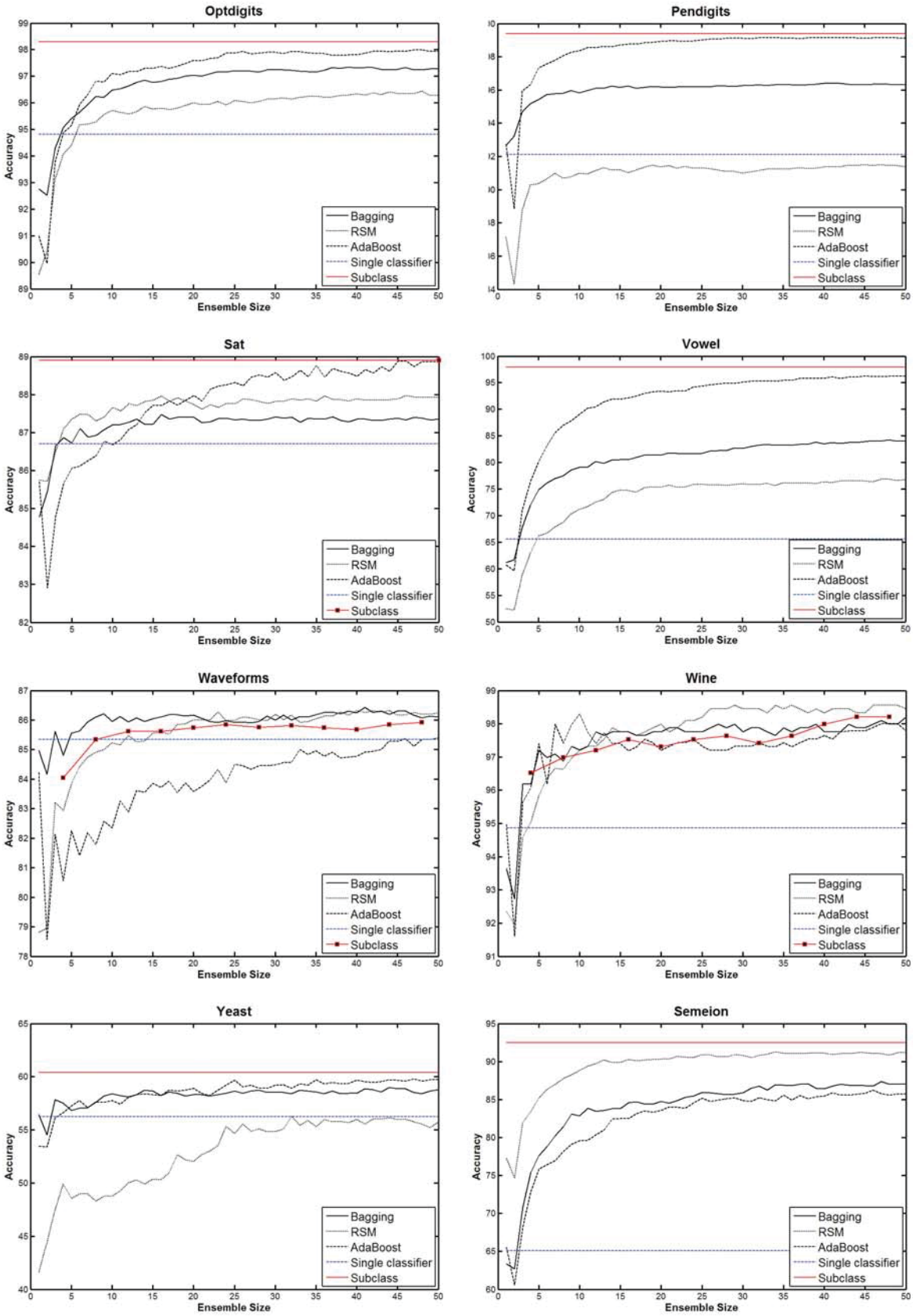


Figure 3.4: Accuracy of ensemble classification methods versus the ensemble size (Cont.)

network as the base learner, we evaluated the efficiency of the generic subclass ensemble on a set of benchmark datasets. Experimental results show that the subclass approach presents a viable alternative to the most commonly used ensemble classification approaches. Specifically, this approach shows a better performance in problems with a larger number of classes. In these cases, instead of combining individual classifiers trained with different subsets of samples or features, the more efficient approach is to train classifiers on a subset of classes.

An important finding is that the generic subclass ensemble works better when there is a larger number of classes. In these cases, instead of combining individual classifiers trained with different subsets of samples or features, the more efficient approach is to train classifiers on a subset of classes. In this way, the problem that each classifier is going to be applied to is relatively smaller and can be solved more efficiently.

Chapter 4

A Genetic-based Subspace Analysis Method for Improving Error-Correcting Output Coding

4.1 Introduction

A common task in many real-world pattern recognition problems is to discriminate among instances that belong to multiple classes, known as *multiclass classification*. There are two general approaches to deal with multiclass problems. One approach is to construct a single decision function by considering all classes (concurrently) and to solve a complex classification problem, known as the single-machine approach [159, 149]. Some classification algorithms, such as the k-Nearest Neighbor (kNN) or Multilayer Perceptron (MLP), are inherently based on this approach. The second approach is to recast the multiclass problem into a series of smaller binary classification problems, which is referred to as "class binarization" [54]. In this way, two-class problems can be solved by binary classifiers and the results can then be combined so as to provide a solution to the original multiclass problem. An extensive comparison of the results demonstrates that the class binarization approach generally achieves a better performance, even for powerful learners [54, 70]. In addition, many established classification algorithms are specifically designed for binary problems, such as Support Vector Machine (SVM) or AdaBoost. Therefore, to solve multiclass classification problems using these binary classifiers, the class binarization approach should be employed.

Among the proposed methods for approaching class binarization, three techniques are well-known: one-versus-all (OVA) [4], one-versus-one (OVO) [66], and Error Correcting Output Codes [37, 2]. In one-versus-all, the multiclass problem is decomposed into several binary problems in the following way: for each class a binary classifier is trained to discriminate among the patterns of the class and the patterns of the remaining classes. In the one-versus-one technique, one classifier is trained to separate

each possible pair of classes. In both approaches, the final classification prediction is usually obtained by means of a voting or committee procedure. More recently, a unified framework was introduced to decompose a multiclass problem into a series of different binary problems, which is known as Error Correcting Output Codes (ECOC). In this framework, each classifier is trained on a two meta-class problem, where each meta-class consists of some combinations of the original classes. The ECOC method can be broken down into two stages: encoding and decoding. The aim of the encoding stage is to design a discrete decomposition matrix (codematrix) for the given problem. Each row of the codematrix, named *codeword*, is a sequence of bits representing each class, where each bit identifies the membership of the class to a classifier [44]. In the decoding stage, the final classification decision is obtained based on the outputs of binary classifiers. Given an unlabeled test sample, each binary classifier casts a vote for one of the two meta-classes used in its training. The output vector is compared to each class codeword of the matrix and the test sample is assigned to the class whose codeword is closest to the output vector, according to a distance measure. Because of the ability of the ECOC framework to correct the bias and variance errors of the base classifiers [85, 86, 172], it has been successfully applied to a wide range of applications [34, 156, 82].

The priority when designing ECOC matrices is to improve the error correcting capability of the codematrix, mainly by maximizing a separability criterion between any pair of rows and/or any pair of columns. In general, optimizing row separation criteria directly leads to more error-correcting capability. According to error-correcting theory, it can easily be shown that a matrix having d bits error-correcting capability implies that there is a minimum Hamming distance of $2d+1$ between any pair of rows (codewords). Assuming that each codebit is transmitted independently, it is then possible to correctly classify a received test codeword having fewer than d bits in error, by assigning that codeword to the closest codeword based on the Hamming distance. Therefore, it is desirable to design a codematrix with a high minimum Hamming distance between any pair of codewords. However, the capability to detect and possibly correct errors is *"dependent on the assumption that each error is independently produced"* [171, 172]. Therefore, the independence of binary classifiers is the cornerstone of the design of ECOC matrices, without which the ECOC method would

be ineffective. The intuition is that if each binary classifier makes different errors, then the ECOC's ability to detect and possibly correct errors would be improved.

The conventional strategy in the ECOC literature for designing independent classifiers is to optimize the distance between ECOC dichotomizers. This property is generally achieved by maximizing the Hamming distance between each column and the others, including their complementaries. Several methods have been proposed that aim to simultaneously optimize row and column separation, such as BCH coding [122], CHC coding [46], and evolutionary techniques [55]. Interestingly, the extensive experimental results show that codes designed using only a row separation criterion performed almost as well as codes designed using column and row separation. However, codes designed using only column separation criteria performed significantly worse [54]. In addition, many researchers agree that a pseudo-random generation of a codematrix is a reasonably good method, and that *"more sophisticated methods might have only marginal effect on testing error"* [37, 54, 140]. These results reveal that conventional strategies to design a codematrix will not promote independence among binary classifiers.

One efficient approach to increase diversity among an ensemble of classifiers is to train each learner with data that consist of different feature subsets, leading to uncorrelated errors by base learners [69, 132]. This idea, usually called *subspace* approach, can effectively make use of the diversity of base learners to reduce the variance as well as the bias errors [56, 154]. Inspired by this idea, we design a new method for the ECOC framework, named Subspace ECOC. The strategy consists of using different feature subsets for each dichotomizer, leading to more independent classifiers and, consequently, increasing the overall system accuracy. In addition to the design of more independent classifiers, the new technique allows for the design of larger codes in comparison to classical methods.

Some previous studies have proposed the use of bagging and boosting within the ECOC framework, mainly by selecting a sampling of data for each dichotomizer in order to increase the diversity of binary problems. In this sense, Schapire proposed a new technique by combining the boosting algorithm with the idea of output codes [140]. Similarly, Windeatt and Ardeshir proposed to combine the AdaBoost, a version of boosting, with output coding using the decision tree as a base learner [169].

Although previous methods have performed sampling of data, they used the same set of available features, so it is likely that some classification errors will be common, arising from noisy or non-discriminant features. To our knowledge, there is no related work that performs feature selection within the ECOC framework independently of the base classifier.

Another relevant factor for a codematrix to achieve a good performance is the problem-dependent design of the codes. That is, for a given problem we need to take into account the characteristics of the problem at hand. While most previous work tried to design a generic codematrix for any classification problem, few studies tried to develop the codematrix by considering the problem characteristics or the classification performance. In this work, we attempt to tackle this issue using an evolutionary algorithm-based optimization approach in order to guide the feature selection of the three-dimensional ECOC matrix for each problem. More specifically, the Genetic Algorithm (GA) is employed, which has been shown to provide an efficient trade-off between the quality of the solution and the search complexity. In this way, the efficiency of the whole ensemble for the problem at hand is considered in the optimization process of the Genetic Algorithm. As a result, our problem-dependent coding design in the ECOC framework based on the feature subspace approach not only provides more independent classifiers, but also increases the overall classification accuracy.

The rest of this chapter is organized as follows: Section 4.2 provides a brief introduction to the ECOC framework. The proposed method based on the feature subspace is explained in detail in Section 4.3. Section 4.4 reports the experiments we performed using benchmark datasets. Finally, Section 4.5 summeriness the chapter.

4.2 Error Correcting Output Codes

First, we briefly describe some notations used in this chapter:

- $T = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_m}, y_m)\}$. A training set; where $\mathbf{x_i} \in R^n$; and each label,y_i, is an integer belonging to $Y = \{1, 2, \ldots, N_c\}$, where N_c is the number of classes.

- $h = \{h_1, h_2, \ldots, h_L\}$: A set of L binary classifiers.

Table 4.1: An example of an ECOC matrix.

Class	h_1	h_2	h_3	h_4	h_5	h_6
c_1		■		■	■	
c_2			■	■		■
c_3	■		■		■	
c_4	■	■		■		

4.2.1 ECOC overview

The basis of the ECOC framework consists of designing a codeword for each of the classes. This method uses a matrix M of $\{1, -1\}$ values of size $N_c \times L$, where L is the number of codewords codifying each class. This matrix is interpreted as a set of L binary learning problems, one for each column. That is, each column corresponds to a binary classifier, called *dichotomizer* h_j, which separates the set of classes into two metaclasses. Instance $\mathbf{x}$, belonging to class i, is a positive instance for the jth classifier if and only if $M_{ij} = 1$ and is a negative instance if and only if $M_{ij} = -1$. Table 4.1 shows a possible binary coding matrix for a 4-class problem $\{c_1, \ldots, c_4\}$ with respective codewords $\{M(r, .)\}$ that uses six dichotomizers $\{h_1, \ldots, h_6\}$. In this table, each column is associated with a dichotomy classifier, h_j, and each row is a unique codeword that is associated with an individual target class. The white cells of the table refer to +1 and the dark cells stand for −1. For example, h_3 recognizes two meta-classes: original classes 1 and 4 form the first meta-class, and the other two form the second one.

When testing an unlabeled pattern, $\mathbf{x}^*$, each classifier outputs a -1 or 1, creating a L long output code vector. This output vector is compared to each codeword in the matrix, and the class whose codeword has the closest distance to the output vector is chosen as the predicted class. The process of merging the outputs of individual binary classifiers is called decoding. The most common decoding method is the Hamming distance. This method looks for the minimum distance between the prediction vector and the codewords:

$$y_H = \arg \min_{r \in \{1,\ldots,N_c\}} \sum_{i=1}^{L} \frac{1 - sign(M(r, i) \cdot f_i(x))}{2}, \tag{4.1}$$

where $sign(z)$ is +1 if $z > 0$, -1 if $z < 0$ and 0 otherwise. $M(r, .)$ designates the codeword r in the matrix and $y_H \in \{1, \ldots, N_c\}$ is the predicted label. For example,

the output $[+1-1-1-1+1-1]$ is closest to c_2 codeword with a Hamming distance of 1, and hence c_2 would be chosen as the predicted label of x^*. Several decoding strategies (combination methods other than distance methods) have been proposed in the literature, such as probabilistic approaches [119, 185] and loss-functions strategies [2]. The reader is referred to [42, 172] for a more detailed analysis.

4.2.2 Coding designs

There are several coding designs that can be used for ECOC methods, which can be broadly divided into two main approaches, which we have named static and dynamic coding designs. The static codings are for designing generic code matrices for any classification problem, regardless of the learning algorithm and the problem to which the codematrix is going to be applied. The design of these matrices is usually based on their error correcting capabilities or on maximizing a separability criterion between rows and columns. On the other hand, the dynamic coding methods take into account the characteristics of the problem at hand. This approach can yield higher classification performance for a specific learning algorithm [55] at the cost of higher computational complexity and less generalization of the designed matrices to other problems. In the following subsections, we review state-of-the-art coding designs based on these two families of approaches.

Static coding designs (problem-independent)

Most of the popular ECOC coding methods fall into the static coding category.In this category, the most well-known coding techniques are dense random coding, consisting of a binary matrix, and sparse random coding, using a third symbol (zero). The zero value in sparse coding means that a given class is not considered in the training phase of a particular classifier. Allwein et al. [2] suggested a length of $10log_2(N_c)$ and $15log_2(N_c)$ bits per code for dense and sparse coding styles, respectively.

Coding methods in this category are defined independently of the problem domain and seek to satisfy two basic criteria:

- **Row separation:** Each codeword should be as far apart from the other codewords. The standard measure of the error-correcting ability of any codematrix is the minimum Hamming distance between any pair of codewords.

- **Column Separation:** In addition to row separation, each dichotomizer, h_i, should be well-separated from the other dichotomizers. This property results in low correlated classifiers in the ensemble.

Taking into account these two criteria, several methods have been proposed in order to optimize the coding design, such as the algebraic-based BCH codes [19], randomized hill climbing [37], simulated annealing and evolutionary computation [14, 55]. Kuncheva [90] used the disagreement diversity measure, a criterion from the literature on classifier ensembles, and suggested an evolutionary algorithm for constructing the codematrix.

Dynamic coding methods (problem-dependent)

Utschick and Weichselberger [157] proposed one of the first problem-dependent ECOC designs. In their work, they developed a method based on the application of maximum-likelihood objective function by means of the Expectation-Maximization (EM) algorithm in order to achieve a suboptimal decomposition of the multiclass problem into binary problems. Escalera et. al proposed a new problem-dependent ECOC design based on subclass information in the ECOC framework [45]. Multiclass problems are solved by splitting the original set of classes into subclasses and embedding the binary problems in the ECOC design. Crammer and Singer proposed a method to find an optimal coding matrix by changing its representation from discrete to continuous values [33]. Pujol et al. [124] proposed a heuristic method, named Discriminant ECOC, to build the ECOC matrix based on a hierarchical partition of the class space that maximizes a discriminative criterion. They also proposed ECOC-optimizing node embedding (ECOC-ONE) [41]. This method uses a coding process that trains relevant binary problems guided by a validation set. Their proposed procedure begins with an initial codematrix and aims to recursively optimize the codematrix by minimizing errors in the confusion matrix by using the validation samples. The authors suggested a length of $2N_c$ bits per code. In [3] a method is proposed to learn the error-correcting output codes from data, where the backpropagation algorithm is used to drive the codewords for each class. In [184], a method is proposed to explore the distribution of data classes and optimize both the decomposition and the number of base learners, named Data-Driven Error Correcting Output Coding (DECOC). DECOC computes

the confidence score of each base classifier based on the structural information of the training data. Sorted confidence scores are then used to selectively include some of the binary learners in the codematrix. Zhong et. al proposed a method that learns the ECOC matrix and dichotomizers simultaneously from data by formulating the learning model as a sequence of concaveconvex programming problems [183]. Recently, Hatami [67] proposed a heuristic method for an application-dependent design of ECOC matrix based on a thinning algorithm, called Thinned-ECOC. The main idea of the method is to successively remove some unnecessary and redundant columns from the initial codematrix based on a metric defined for each column.

4.3 Genetic Algorithm-based Subspace ECOC (GA-SS-ECOC)

As we stated earlier, there exist two main factors affecting the performance of ECOC methods. The first is that the error committed by each of the binary classifiers needs to be uncorrelated, which makes the ECOC approach effective in correcting the errors. The second factor is that the design of the codematrix cannot be considered independent of the problem to which it is going to be applied. In the following two subsections, our strategies to deal with these two factors are explained. We first present in detail the proposed subspace approach to the output coding framework. Then, the GA-based optimization process that performs problem-dependent feature selection is explained.

4.3.1 Subspace ECOC

The central idea of the proposed Subspace ECOC is based on using feature space in the design process of the ECOC matrix. That is, each dichotomizer is trained with a different feature subset, leading to better classification accuracy. From the design process point of view, we generate a three-dimensional codematrix, where the third dimension is the feature space of the problem domain. In order to generate this framework, first, a two-dimensional codematrix is created from a previous set of matrices that maximizes the minimum distances between any pair of codewords. Then, for each column, a random vector of $\{-1, +1\}$ values of size n is generated, where n is the number of features. The meaning of '+1' ('-1') in the vector is that the corresponding feature is (not) included in the corresponding classifier. Note that in

both, sparse and dense coding styles, the value of each cell in the feature space cannot take the 0 value. The representation of the proposed three-dimensional codematrix is illustrated in Figure 4.1.

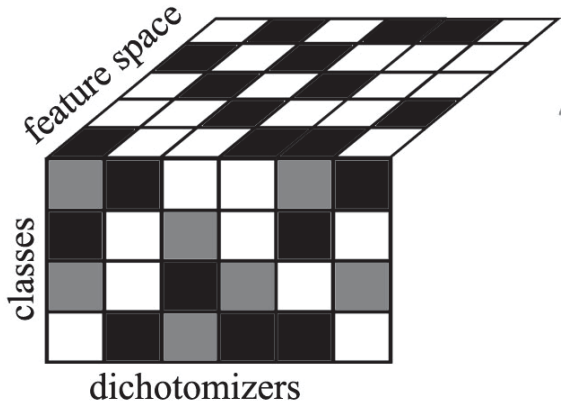


Figure 4.1: The Subspace ECOC approach.

This approach not only creates more independent classifiers, but it can also build longer codewords. It can be shown that the maximum length of codewords in ECOC matrices is $(2^{N_c-1}-1)$ and $(3^{N_c}-2^{N_c+1}+1)/2$ for dense and sparse coding styles, respectively (The proof is given in the Appendix A). Thus, the maximum number of binary classifiers in the classical ECOC methods is small in problems with a relatively small number of classes (i.e. $N_c < 6$). Conversely, the ECOC method with longer codes is able to significantly improve the results [2, 54]. In our proposed approach, since each classifier can be trained using a variety of feature subsets, more diverse classifiers can be built. In an n-dimensional feature space, 2^n-1 different non-empty feature subsets can be selected. So, the number of distinct dichotomizers is $(2^n-1).(2^{N_c-1}-1)$ and $(2^n-1).(3^{N_c}-2^{N_c+1}+1)/2$ for dense and sparse ECOC, respectively. The other advantage of the subspace approach is that each binary classifier requires less training time, since it uses fewer features.

4.3.2 Improving the subspace ECOC coding by GA

As mentioned earlier, most previous work on ECOC was focused on the generation of a coding matrix without considering the characteristics of the problem at hand. Recently, some researchers argue that using the knowledge of the problem domain to learn relevant binary problems has a significant effect on ECOC accuracy. The basic strategy of these studies is to use the training data to guide the design process,

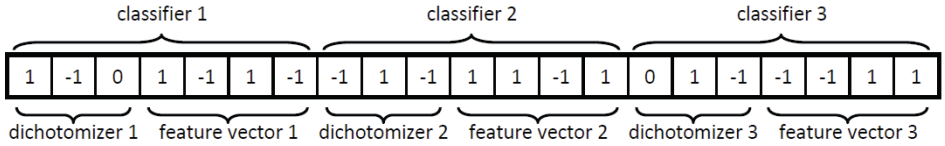


Figure 4.2: A possible encoding of a problem with three dichotomizers, three classes, and four features.

and thus, to develop a coding matrix that focuses on binary problems that better fit the decision boundaries of a given problem. This problem-dependent development can be considered as an optimization design process. One promising strategy for this optimization issue is to use an evolutionary algorithm-based approach. One of the most well known evolutionary approaches is the Genetic Algorithm, which is inspired by an explicit imitation of biological life, in which weaker units (individuals with lower fitness) are eliminated and the strongest (fittest) individuals survive to produce the next generations.

In the proposed GA-based Subspace ECOC method, each chromosome of the population is a three-dimensional codematrix. Consider a problem with N_c classes, L dichotomizers, and n features. Each matrix is encoded by a vector of length $L(N_c + n)$. The first $N_c + n$ bits represent the first classifier (the first dichotomizer and its corresponding feature subset), followed by the $N_c + n$ bits for the second classifier ,h_2, and so on. Based on two ECOC schemes, i.e. dense and sparse coding, two versions of the GA-based Subspace ECOC are proposed. In the dense scheme, each bit of a chromosome has a value of +1,-1, whereas in the sparse scheme, bits related to dichotomizers may take the zero value as well. A possible chromosome encoding for a problem with three dichotomizers, three classes, and four features is illustrated in Figure 4.2.

Due to the fact that a chromosome corresponds to a three-dimensional codematrix, there are few changes required in mutation and particularly in cross-over operations of the standard Genetic Algorithm. Mutation flips the sign of a randomly selected bit of a chromosome. However, since flipping the sign of bits corresponding to dichotomizers may result in a non-valid dichotomizer, the mutation is only applied to bits corresponding to feature vectors [1]. This modification ensures that the premature

[1]As a simple example, if we mutate the third bit of [1 1 -1], a non-valid dichotomizer will be generated.

convergence of the algorithm is avoided. The cross-over operation is more sophisticated, since a chromosome consists of a sequence of two entities: a dichotomizer and its corresponding feature vector, each one having different concepts and clearly different possible values. The proposed technique is to randomly choose the cross-over point from the positions that encoded classifiers end. The cross-over operation is schematically presented in Figure 4.3. The GA-based Subspace ECOC method can be summarized in the following form:

1. Pick the parameters of the Genetic Algorithm:
   - Population size (P),
   - Maximum number of iterations,
   - Mutation probability.
2. Generate a random population of chromosomes and calculate their fitness values as the classification accuracy of each individual (i.e. each ECOC matrix) on the validation data.
3. Perform one-point cross-over and mutation to generate the offspring chromosomes.
4. Calculate the fitness values of the offspring chromosomes.
5. Pool offspring and the current population together and select P chromosome with the highest fitness as the next generation.
6. If the stopping criteria are met, finish; else go to step 3.

It is worth mentioning that this special implementation of the Genetic Algorithm is a kind of hill-climbing strategy, as it guarantees that the fitness value will not decrease in the subsequent generations.

4.4 Experimental Comparison Over Benchmark Datasets

In this section, we first discuss the experimental settings of the experiments including the data, the comparative methods, and the evaluation measurements. We then provide a detailed comparison of results achieved by different methods.

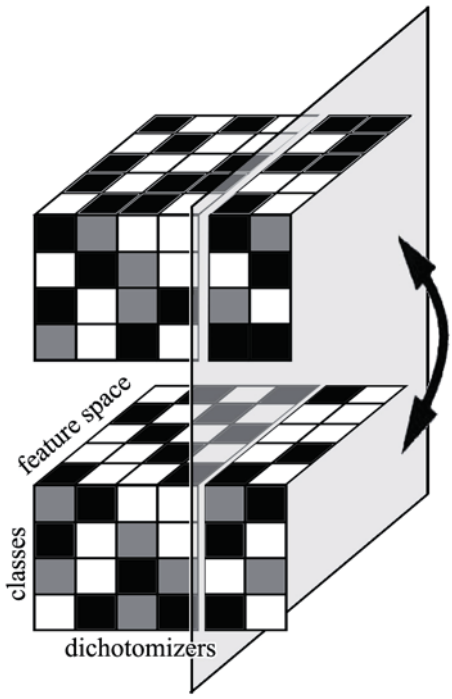


Figure 4.3: A schematic representation of the cross-over operation in GA-SS-ECOC.

4.4.1 Experimental settings

- **Data:** The proposed GA-SS-ECOC method was first validated on 20 multiclass datasets from the UCI machine learning repository [16]. Table 4.2 shows the number of classes, instances, and features of each UCI dataset.

- **Methods:** We compared our proposed method with classical static methods including OVO, OVA, and dense random and sparse random ECOC methods as well as two state-of-the-art problem-dependent coding methods, including Discriminant ECOC and ECOC-ONE. The class of an instance in the ECOC schemes was chosen using the Exponential Loss-Weighted (ELW) decoding [42]. In order to limit the computational complexity of the experiments in the GA-SS-ECOC design, the population size and maximum number of iterations in the GA optimization algorithm was set to 10 and 3, respectively. In addition, if the best fitness did not change in two successive iterations, the optimization process was stopped. The mutation rate was set to $p = 0.1$ in order to promote diversity in the population.

  We set the number of different feature subsets for each nontrivial dichotomizer

Table 4.2: Summary of the datasets used.

	Dataset	# instances	# features	# classes
1	Abalone	4177	8	3
2	Cmc	1473	9	3
3	Derm	366	34	6
4	Ecoli	336	7	8
5	Glass	214	10	7
6	Iris	150	4	3
7	Lymph	148	18	4
8	Mfeat-mor	2000	6	10
9	Mfeat-pix	2000	240	10
10	Mfeat-zer	2000	47	10
11	Optdigits	5620	64	10
12	Sat	6435	36	6
13	Thyroid	215	5	3
14	Vehicle	846	18	3
15	Vertebral	310	6	3
16	Vowel	528	10	11
17	Waveforms	5000	40	3
18	Wine	178	13	3
19	Yeast	1484	8	10
20	Zoo	101	16	7

as 10. Thus, codewords are 10 times longer in the Subspace ECOC design for both dense and sparse ECOC methods. In this study, two base learners were chosen: a classification and regression tree (CART) with the Gini-index as a split criterion and a multilayer perceptron (MLP) with 10 hidden nodes and the hyperbolic tangent transfer function. The MLP classifier cannot handle the missing values, so the instances with missing values were removed.

- **Evaluation measurements:** The classification accuracy was obtained by means of 10-fold cross-validation to improve the reliability of the results. In order to have a fair comparison, the training and test sets of all methods were the same for each repetition of experiments. Moreover, using non-parametric tests we showed that the performance of rival methods was statistically different.

4.4.2 Experimental results

The average accuracy of the rival methods for each dataset is presented in Table 4.3 and Table 4.4. In these tables, the means of prediction accuracy over 10 runs (expressed in %) are reported for each method on the considered datasets. For each dataset, the best accuracy achieved among all tested methods is in bold.

Table 4.3: Classification accuracies of different methods using CART.

	OVO	**OVA**	**Dense ECOC**	**Sparse ECOC**	**Dis. ECOC**	**ECOC-ONE**	**GA-SS dense ECOC**	**GA-SS sparse ECOC**
Abalone	58.50	58.20	58.20	59.72	57.50	58.84	62.80	**63.55**
Cmc	50.28	49.87	49.87	51.85	49.70	50.01	52.14	**52.72**
Derm	95.93	94.81	97.49	96.82	94.42	96.37	**97.88**	97.72
Ecoli	83.22	77.24	85.62	84.67	81.60	85.00	**87.32**	85.84
Glass	72.44	61.94	72.30	73.41	64.90	72.01	76.76	**78.27**
Iris	93.33	93.20	93.73	93.33	93.71	93.33	**94.13**	93.73
Lymph	75.69	75.94	77.18	77.07	74.07	77.18	81.09	**81.20**
Mfeat-mor	68.42	65.36	**72.14**	71.14	66.67	68.72	70.53	70.22
Mfeat-pix	89.41	81.57	96.52	96.49	78.54	90.26	96.94	**97.04**
Mfeat-zer	68.97	58.78	79.75	78.81	61.42	69.47	**80.56**	79.58
Optdigits	91.69	83.84	97.22	96.82	81.67	91.60	**97.73**	97.50
Sat	86.70	83.95	90.66	91.18	85.02	86.75	91.88	**91.89**
Thyroid	92.05	92.50	92.50	92.05	93.31	92.85	94.58	**95.23**
Vehicle	72.53	73.30	73.16	74.87	73.63	73.24	**76.71**	76.48
Vertebral	79.35	78.28	78.28	78.71	78.82	78.92	**80.75**	80.65
Vowel	79.25	71.80	95.38	93.60	72.02	80.80	**96.44**	95.72
Waveforms	75.78	72.48	72.46	79.10	74.82	74.85	82.86	**83.71**
Wine	93.74	91.50	91.50	94.53	92.47	94.53	**97.76**	97.43
Yeast	56.69	52.80	59.60	**60.68**	53.48	56.44	59.94	60.43
Zoo	86.73	90.18	93.18	92.91	87.27	89.09	**94.45**	93.27
Mean	**78.53**	**75.38**	**81.34**	**81.89**	**75.75**	**79.01**	**83.66**	**83.61**

In order to show the superiority of the proposed ECOC method in terms of the classification accuracy, statistical analysis is necessary. According to the recommendations of Demsar [36], we consider the use of non-parametric tests. Non-parametric tests are safer than parametric tests, such as ANOVA and t-test, since they do not assume normal distribution or homogeneity of variance. In this study, we employed the Iman-Davenport test. If there are statistically significant differences in the classification performance, then we can proceed with the Nemenyi test [108] as a post-hoc test, which is used to compare the methods with each other.

We first rank competing methods for each dataset. The best performing method gets a rank of 1, the second best is ranked 2, etc. The method's mean rank is obtained by averaging its ranks across all datasets. Then, we use the Friedman test [36] to

Table 4.4: Classification accuracies of different methods using MLP.

	OVO	**OVA**	**Dense ECOC**	**Sparse ECOC**	**Dis. ECOC**	**ECOC-ONE**	**GA-SS dense ECOC**	**GA-SS sparse ECOC**
Abalone	66.13	66.02	65.99	66.62	66.27	66.47	66.76	**67.02**
Cmc	53.58	51.73	51.68	53.65	50.66	50.76	54.25	**55.12**
Derm	92.83	93.40	97.41	97.30	91.27	94.34	97.40	**97.68**
Ecoli	82.33	83.53	87.10	86.89	83.37	82.33	**87.49**	87.01
Glass	63.03	59.30	65.44	64.98	65.00	65.15	66.74	**67.80**
Iris	95.56	95.56	95.56	**96.44**	95.56	96.22	96.22	95.56
Lymph	75.40	69.59	79.28	80.69	73.25	79.02	85.64	**86.21**
Mfeat-mor	74.52	76.60	75.00	74.20	74.78	74.35	**76.92**	74.91
Mfeat-pix	94.55	89.50	91.11	93.88	90.87	94.32	94.50	**95.18**
Mfeat-zer	81.50	79.39	82.14	82.07	80.45	82.20	**82.72**	82.31
Optdigits	95.91	96.62	97.51	97.33	94.88	96.80	**97.82**	97.73
Sat	89.01	86.74	89.29	89.46	88.04	89.56	89.59	**90.06**
Thyroid	95.23	95.68	96.36	95.91	96.36	**96.82**	96.30	96.75
Vehicle	80.79	81.22	81.44	82.76	82.08	82.81	83.34	**83.46**
Vertebral	81.45	82.90	83.23	83.77	84.10	**84.52**	83.87	84.19
Vowel	93.74	82.25	97.29	96.82	83.59	93.50	97.43	**97.64**
Waveforms	85.27	84.37	84.34	86.11	85.90	86.15	86.64	**86.96**
Wine	93.30	95.24	94.93	97.50	94.56	97.67	**98.06**	**98.06**
Yeast	58.99	54.34	59.69	**60.66**	57.77	58.78	59.98	59.96
Zoo	94.09	92.27	95.45	**95.91**	90.45	90.30	95.45	95.45
Mean	**82.36**	**80.81**	83.51	84.15	81.46	83.10	84.86	84.95

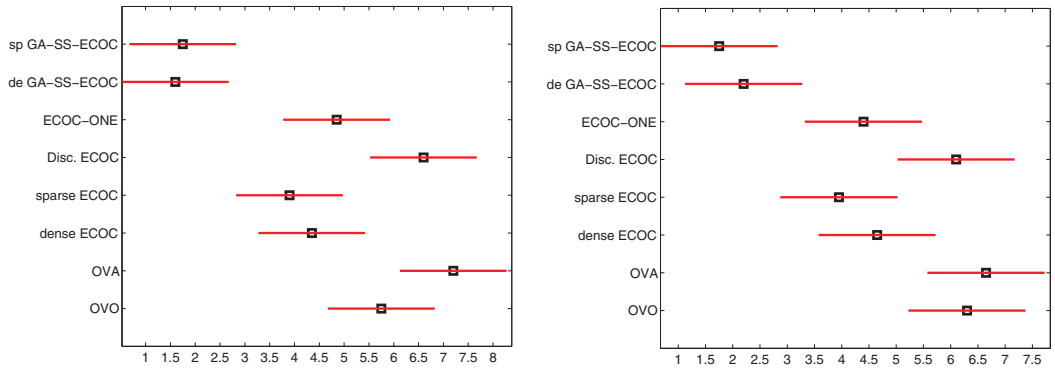
compare these mean ranks to decide whether to reject the null hypothesis, which states that all considered methods have the equivalent performance. The Friedman statistic value is computed as follows:

$$\chi_f^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right], \tag{4.2}$$

where k is the number of competing methods and N is the number of experiments. In our case, when comparing six methods in 20 experiments, $\chi_f^2 = 99.13$ and $\chi_f^2 = 78.70$ for CART and MLP, respectively. Iman and Davenport [74] found that this statistic is undesirably conservative, and proposed a revised one:

$$F_F = \frac{(N-1)\chi_f^2}{N(k-1)-\chi_f^2}, \tag{4.3}$$

which is distributed following an F distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. By applying this correction we obtained $F_F = 46.08$ and 24.39 for CART and MLP, respectively. The critical value of F(7,19) for $\alpha = 0.05$ is 2.54. As the values



(a) CART Classifier (b) MLP Classifier

Figure 4.4: Comparison results of rival methods based on the Nemenyi test.

of F_F are higher than 2.54 we can reject the null hypothesis, that is, the results are not obtained because of randomness.

Further, to compare rival methods with each other, we applied the Nemenyi test. Two methods are significantly different if their corresponding average ranks differ by at least the critical difference value (CD):

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{4.4}$$

In our case, the critical value for a 90% of confidence is $CD = 2.78\sqrt{0.6} = 2.15$. The results of the Nemenyi test are illustrated in Figure 4.4. In this figure, the mean rank of each method is indicated by a square. The horizontal bar across each square shows the critical difference. Two methods are significantly different if their corresponding average ranks differ by at least the critical difference value. That is, their horizontal bars do not overlap. Looking at the rankings of each coding strategy, we can conclude that the proposed GA-SS-ECOC schemes are significantly better than other strategies in the present experiments.

The results in Table 4.3 and Table 4.4, along with the statistical tests presented in Figure 4.4, indicate that overall, the GA Subspace approach achieves the best performance among all methods. As a general conclusion, the advanced performance of the proposed method does not differ much depending on the base classifier. Using both neural network and decision tree as the base learner, we found significant differences between GA-SS-ECOC and classical ECOC for both dense and sparse

Figure 4.5: Some examples of shapes in the MPEG7 dataset.

schemes. An analysis of the results shows that when the number of training patterns is relatively small compared with the dimensionality of data, the subspace approach is usually a better choice. Ho [69] showed that while most classification approaches suffer from the curse of dimensionality, the subspace approach can take advantage of high dimensionality.

4.5 Machine Vision Applications

Then, we applied the proposed GA-SS-ECOC to two machine vision problems: logo recognition and shape categorization. As in previous experiments with UCI datasets, CART and MLP classifiers were chosen and their adjustable parameters were the same as previous experiments. Again, 10-fold cross-validation method was used for performance evaluation.

4.5.1 Shape categorization

The first real image classification problem is shape classification, in which we used the MPEG7 dataset[2]. This dataset consists of $C = 70$ classes (bone, chicken, cellular phone, etc.) with 20 instances per class, which represents a total of 1400 object images. All samples were described using the Blurred Shape Model descriptor [43]. This technique describes each shape by means of 100 features. Thus, the MPEG7 is a 100 dimensional dataset. Figure 4.5 shows a few samples for some categories of this dataset.

[2]MPEG7 Repository dataset: http://www.cis.temple.edu/ latecki/

(a) Logo60 (b) Logo30 (c) Logo10

Figure 4.6: Some examples of logos in the database used in our experiments.

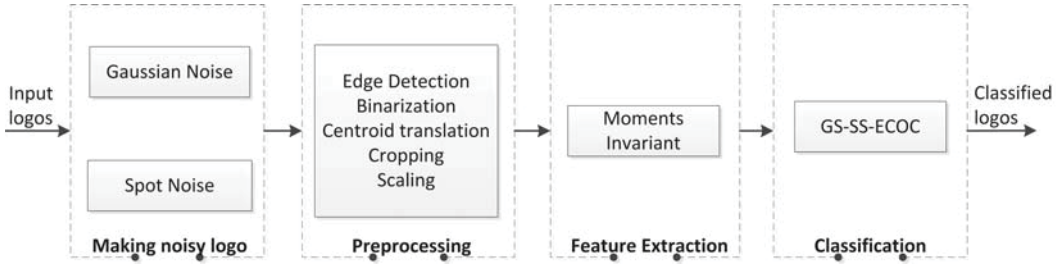


Figure 4.7: The framework of logo recognition process.

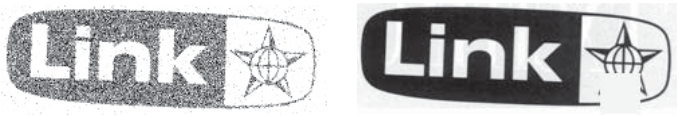
4.5.2 Logo recognition

The ECOC approach was then used in the logo recognition problem. The logo images were based on a database of logos which contains pure pictorial logos (e.g. logo 60, Figure 4.6), text-like logos (e.g. logo 30, Figure 4.6), and text-graphics mixture logos (e.g. logo 10, Figure 4.6). The complete dataset contains 105 images and was obtained from the database distributed by the Document Processing Group, Center for Automation Research, University of Maryland [111]. The logos in the dataset have very different sizes; the smallest one is 121×145 pixels and the largest one is 802×228 pixels. Figure 4.7 shows the framework of our logo recognition process.

This dataset provides only a single instance of 105 individual logo classes. In order to increase the number of samples, for each logo class, some artificially degraded images were generated by using the noise models described in the following subsection.

4.5.3 Noise models

We investigated the robustness of the methods when the logos are corrupted using two different image degradation methods: 1) Gaussian noise (a global degradation as shown in Figure 4.8b); and 2) spot noise (a local degradation as shown in Figure 4.8a). For each method, we degraded each image in the database, varying the amount of degradation in equally spaced steps. We generated a set of 40 examples for each class of logo images by adding both the Gaussian white noise of



(a) Gaussian noise (b) spot noise

Figure 4.8: Examples of noisy logo patterns derived by applying the Gaussian and spot noise model.

$mean = [0, 0.1, 0.2, \ldots, 0.5]$ and $var = [0, 0.01, \ldots, 0.05]$ and the spot noise of different sizes ($width = [10, 15, \ldots, 30]$ pixels).

4.5.4 Feature extraction

Some researchers have studied the problem of logo recognition by applying different feature extraction methods, such as algebraic and differential invariants [38, 76], Zernike and pseudo-Zernike moments [167, 77], line segment Hausdorff distance [28], and template matching [76]. In this work, logo images are described in terms of seven invariant moments, which have been proven to be an effective descriptor of logo and trademark images [76]. Consequently, the logo dataset is a seven dimensional dataset.

Moment invariants

These descriptors, also called geometric moment invariants, were first introduced in 1962 by Hu [71] based on the theory of algebraic forms. These moment features have the desirable properties of being invariant under rotation, translation, scale, and reflection of images and have been widely used in many applications due to their invariance properties. For a 2-D image, $f(x, y)$, the central moment of order $(p + q)$ is defined by:

$$\mu_{pq} = \sum_{x} \sum_{y} (x - \overline{x})^p (y - \overline{y})^q f(x, y), \quad p, q = 0, 1, 2, \ldots \tag{4.5}$$

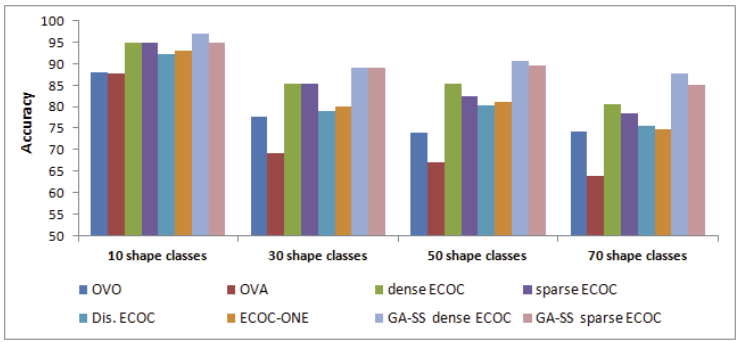
where the pixel point $(\overline{x}, \overline{y})$ is the centroid of the image. Seven moment invariants $(M_1 - M_7)$ based on the 2nd and 3rd order moments are defined [71]:

$$
\begin{aligned}
[b] M_1 =& (\mu_{20} + \mu_{02}), \\
M_2 =& (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2, \\
M_3 =& (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2, \\
M_4 =& (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2, \\
M_5 =& (\mu_{30} + \mu_{12})(\mu_{30} - 3\mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] + \\
& (3\mu_{21} - \mu_{03})(\mu_{21} + 3\mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2], \\
M_6 =& (\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] + \\
& 4\mu_{11}(\mu_{30} + \mu_{12}(\mu_{21} + \mu_{03})) \\
M_7 =& (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] - \\
& (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})[3(\mu_{03} + \mu_{21})^2 - (\mu_{21} - \mu_{03})^2]
\end{aligned}
$$

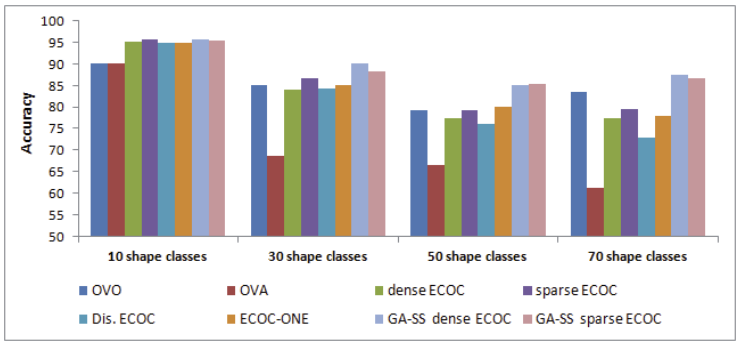
4.5.5 Experimental results and analysis

The average accuracy of all considered methods over 10 runs is illustrated in Figure 4.9 and Figure 4.10 for shape and logo datasets, respectively. The experiments were conducted using different numbers of classes for both datasets. The results demonstrate that the behavior of the six methods follows the general guidelines obtained in the validation of the method using benchmark datasets. It is important to note the high performance of the GA-Subspace approaches in comparison with the standard techniques. This improvement is clearer when the number of classes of the datasets increases. In that case, the inter-class variability is reduced, and thus, it is easier to confuse patterns from different classes. However, given the diversity introduced by the feature selection approach in the proposed GA-Subspace method, we are able to maintain a high level of performance for problems with a large number of classes, while also outperforming the other strategies.

The other finding is that the average accuracy improvement is more significant using the shape dataset. The main reason behind this improvement is that shape patterns are relatively high-dimensional patterns (containing 100 features) in comparison with logo patterns (containing only 7 features). Therefore, the Subspace ECOC approach takes advantage of this high-dimensionality.

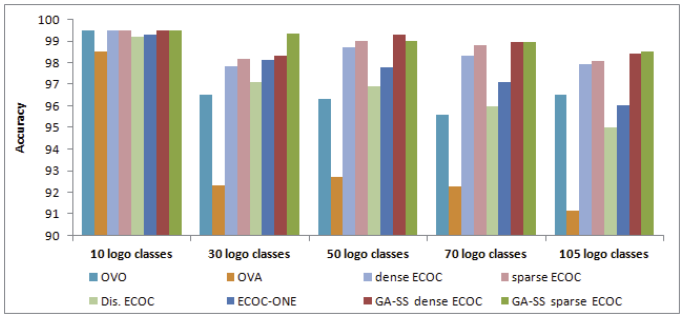


(a) CART Classifier

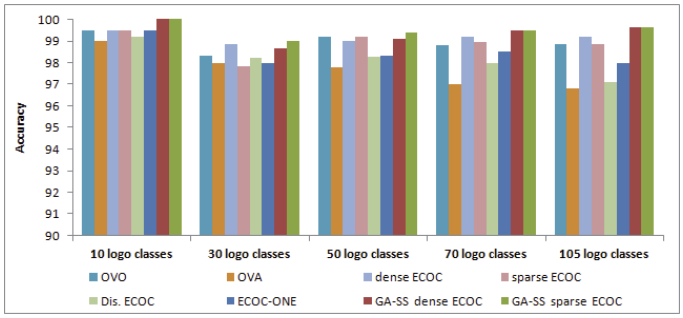


(b) MLP Classifier

Figure 4.9: Average accuracy of different class binarization methods on the MPEG7 dataset.



(a) CART Classifier



(b) MLP Classifier

Figure 4.10: Average accuracy of different class binarization methods on the Logo dataset.

4.6 Summary

In this chapter, a subspace analysis-based approach for the design of the application-dependent codematrix is proposed in the ECOC framework, in which the genetic algorithm is applied for search optimization. The new approach defines a third ECOC matrix dimension which corresponds to the feature space. In this sense, different subsets of features can be activated for a given dichotomy. The proposed method takes advantage of some basic concepts of ensemble classification, such as diversity of classifiers, and also benefits from the evolutionary algorithm-based approach to optimize the three-dimensional codematrix, taking into account the characteristics of data. As a result, we obtain a problem-dependent coding design with more independent classifiers, which reduces the bias and variance errors of the multiclass problem and, consequently, increases the discrimination power of the ensemble. The method was evaluated on several UCI datasets as well as two image classification problems using two different base learners. The experimental results show significant performance improvement of the proposed method compared to state-of-the-art approaches.

Chapter 5

A Framework of Multi-Classifier Fusion for Human Action Recognition

5.1 Introduction

The fast and reliable recognition of human actions from captured videos has been a goal of Computer Vision for decades. Robust action recognition has diverse applications including gaming, sign language interpretation, human-computer interaction (HCI), surveillance, and health care. Understanding gestures/actions from a real-time visual stream is a challenging task for current Computer Vision algorithms.

Over the last decade, spatial-temporal (ST) volume-based approaches and local ST feature representations have achieved good performance on some action datasets. However, they are still far from being able to express the effective visual information for high-level interpretation, because of the weakness of semantics and lack of intrinsic structures. On the other hand, interpreting human actions from tracked body parts is a natural solution that follows the mechanism of human visual perception. The early work conducted by Johansson in 1973 shows that the tracking of joint positions itself encodes significant discriminative information and is sufficient for human beings to recognize different actions [79]. In addition, according to an influential computational model of human visual attention theory [152], visual attention leads to visual salient entities, which provide selective visual information to make human visual perception efficient and effective. Skeleton joints are visual salient points of the human body and their movements in 4D space reflect motion semantics.

From a technological point of view, the development of low-cost depth sensors with acceptable accuracy has greatly simplified the task of action recognition. Most importantly, the recent release of the Microsoft Kinect camera and its evolving skeleton joints detection technique in late 2011 [81] led to a revolutionary effect in the field of Computer Vision and created a wide range of opportunities for demanding

applications. The Kinect sensor captures depth information of a scene, in addition to the RGB image acquired by a camera. Therefore, Kinect provides synchronized color and depth images, usually called RGB-D (RGB plus depth). Shotton et al. [144] proposed one of the greatest advances in the extraction of the human body pose from depth data, which is provided as a part of the Kinect platform. Their work enables us to recover 3D positions of skeleton joints in real time and with reasonable accuracy [57, 144].

In this chapter, we focus on human action recognition by using skeleton joint information extracted from depth sequences. From the pattern recognition point of view, action recognition is considered a multi-class classification task, where each action type is a separate target class. In this view, the classification system involves two main stages, the selection and/or extraction of informative features and the construction of a classification algorithm. In such a system, a desirable feature set can greatly simplify the construction of a classification algorithm, and a powerful classification algorithm can work well even with a low discriminative feature set.

Here, we first present two efficient action description techniques by only considering skeleton joint information. Then, we propose the use of an ensemble classification framework in order to improve the efficiency of the recognition stage. To do this, we employ different action recognition learners and aim to efficiently fuse them by the Dempster-Shafer fusion method.

In summary, the contributions of this chapter are as follows: (1) We introduce two simple, yet efficient, action description techniques only considering skeleton joint location information; (2) We apply an ensemble framework to address the action/gesture recognition problem; (3) We efficiently combine individual classifier outputs by means of the Dempster-Shafer fusion method, taking benefit from diversity of base classifiers trained on different source of information.

5.2 Related work

Until 2011, the main source for human action recognition was based on a sequence of RGB images, i.e. a color video. However, for many applications that rely on precise tracking without specific body part positions, color-based approaches can easily fail due to occlusion, dynamic environments and intra-class variations. With the

development of depth sensors, active research has been conducted on action/gesture recognition using depth and skeleton data. In this section, we first briefly review studies of human activity analysis using RGB data. Then, we focus on related work on human action recognition using depth and skeleton data.

5.2.1 Activity recognition using color images

Various representational methodologies have been proposed to recognize human actions/gestures from color image sequences. They can be broadly categorized into four approaches. The first approach is based on the sequential representation of a video using a sequence of feature vectors. Each vector may contain color, location, orientation, size and shape features of one or more images. The second approach is to represent 2D images as a 3D Space-Time (ST) volume, constructed by concatenating XY images along the time T. 3D ST volumes can be viewed as rigid objects. The volume-based representations are generated using the salience features of the rigid objects. A well-known methodology is developed by Bobick and Davis [17]. They proposed two 2D images: a binary motion energy image (MEI) and a scalar-valued motion history image (MHI). These methods are constructed from a sequence of foreground images. MEI and MHI essentially are weighted XY projections of the original 3D ST volume. The third approach is trajectory-based representation. In this approach, a person is usually represented as a set of 2D or 3D points, corresponding to his joints. A human body part estimation is necessary for obtaining the joint positions. As an example, Wang et al. [162] recently proposed the use of dense trajectories to describe videos. Dense trajectories are constructed by matching dense points in the optical flow field between frames. After removing noise trajectories, the motion patterns are encoded by the trajectory shapes. Human motion is represented by a set of histogram-based descriptors that represent the local and global properties of the dense trajectories.

The fourth approach, which has gained more attention in the related literature, is based on local feature representation. In this approach, appropriate salient points or regions, representing an action by 3D ST volume, are extracted and then the action is described by the local feature-based representations around those salience entities within the ST volume. Laptev and Lindeberg extracted space-time interest

points (STIPs) by using the Harris3D detector [91]. Some other ST saliency detectors are Hessian [168] and dense spatiotemporal saliency [126]. Based on these extracted salient points, several local ST descriptor methods, such as HOG/HOF [92] and extended SURF [39] have been proposed. The use of skeleton-based descriptors has been included in action recognition models such as the Hidden Markov model (HMM) [101] and the Conditional Random Field (CRF) model [65]. Another common strategy for video representation is based on the Bag-of-Word (BoW) approach of RGB image sequences [97, 162]. These intermediate level local feature descriptors contain image semantics, being robust to noise.

Figure 5.1 shows the general timeline of outstanding research studies in the field of human activity analysis.

5.2.2 Activity recognition using depth and skeleton data

Within the last two years after Kinect was released, a large number of works have appeared in diverse Computer Vision domains. Figure 5.2 shows a tree structure taxonomy of vision problems that can be addressed or enhanced by means of Kinect [64]. In this subsection, we focus on activity recognition studies by Kinect.

The story of human action recognition using depth and skeleton data begins with a recent work by Shotton et al [144]. They proposed a Random Forest-based classification method to find body joints from depth images. This approach has been recently enhanced by other works [57, 143], which provide accurate 3D estimations of skeleton joint locations.

Li et al. [95] adopted an action graph to model the temporal dynamics of the actions and proposed the use of a bag of 3D points, extracted from the depth map. Also, they propose a projection based sampling technique to sample the bag of 3D points from a body surface to characterize the posture being performed in each frame. Their experimental results on the MSR-Action3D dataset showed advanced performance of their method, compared to the 2D silhouette based recognition. Reyes et al. [128] proposed an automatic feature weighting approach within the Dynamic Time Warping method for real-time gesture recognition. In their method, weights are assigned to features based on inter-intra class action variability.

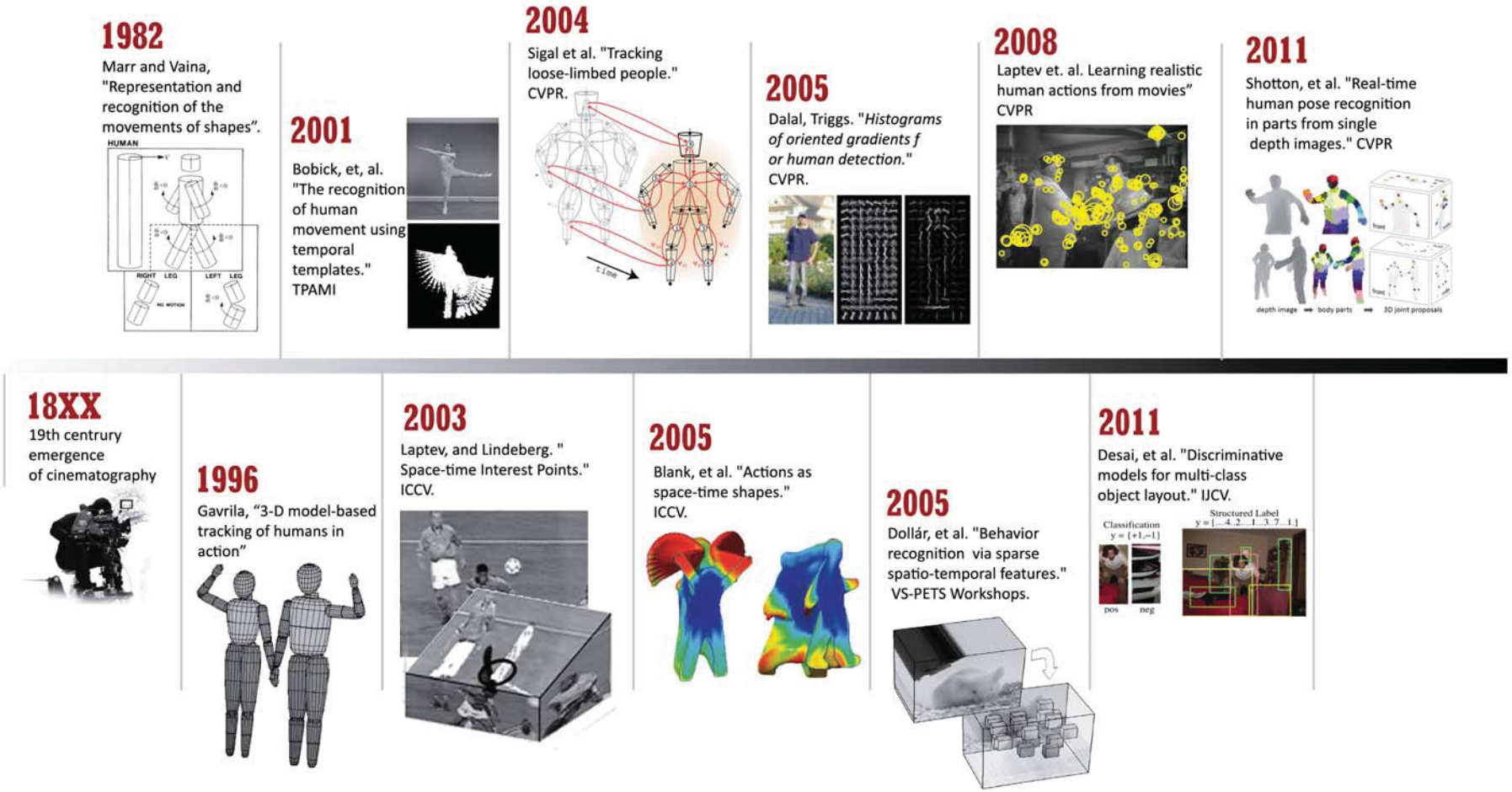


Figure 5.1: Timeline of outstanding research studies in the field of human activity analysis.

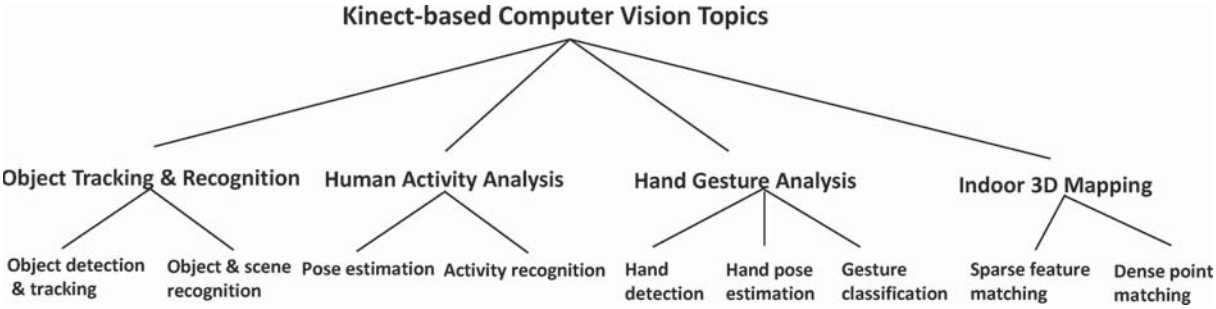


Figure 5.2: Taxonomy of applications of Kinect in vision problems [64].

In [174], a histogram of 3D joint locations (HOJ3D) for body posture representation is proposed. In this representation, the 3D space is partitioned into bins using a spherical coordinate system, and the HOJ3D histogram is constructed by casting joints into certain bins. After applying linear discriminant analysis (LDA) for dimensionality reduction, HOJ3D vectors are clustered into k posture visual words. The temporal behaviour of these visual words is coded by discrete Hidden Markov models (HMMs). In [176] and [177], visual features for activity recognition are computed based on the spatial and temporal differences among detected joints. This feature set contains information about static posture, motion, and offset. In their method, a feature descriptor will be generated for each frame. For classification, they employed the Naive Bayes Nearest Neighbor (NBNN) method.

There are studies that have fused different sources of information, i.e. RGB, depth, and skeleton data. Wang et al. [164] extracted two types of features from each frame: 1) the pairwise relative difference between joints' positions (skeleton data); and 2) local occupancy patterns (LOP) at each joint extracted from depth maps. This LOP feature encodes the local occupancy information based on the 3D point cloud around a joint. In addition, the Fourier temporal pyramid is then employed to represent the temporal dynamics. For the recognition phase, they introduced the concept of actionlet, which is a conjunction of the features for a subset of the joints, indicating a structure of the features. Then, they proposed a data mining solution to discover discriminative actionlets. Thus, an action is represented as an Actionlet Ensemble, which is a linear combination of the actionlets. Their discriminative weights are learnt via a multiple kernel learning method. Indeed, their actionlet ensemble follows the concept of feature selection strategies.

Figure 5.3: Different data modalities of the Chalearn dataset.

In [68], Bag-of-Visual-and-Depth-Words is proposed, containing a vocabulary of HOG/HOF and PFH/FPFH descriptors from RGB and depth sequences, respectively. This novel representation was also used within the classical Dynamic Time Warping method, including Gaussian-like probabilities to compute the warping path, performing multi-modal action recognition. Sung et al. [147] used both RGB and depth channels to recognize human daily activities. They proposed a 459-element feature vector from various body joints for each frame, and then a two-layered Maximum Entropy Markov Model (MEMM) was applied to recognize single person activities. In [98], the authors introduced an adaptive learning methodology to simultaneously extract spatio-temporal features from RGB and depth data. The feature learning is optimized using a restricted graph-based genetic programming (RGGP) approach. More recently, Oreifej and Liu[116] proposed an efficient action representation technique that aims to capture shape and motion features simultaneously, named HON4D. They described the depth sequence using a histogram capturing the distribution of the surface normal orientation in the 4D space of time, depth, and spatial coordinates. Their results on three action datasets showed that their descriptor outperforms the state-of-the-art methods.

5.3 Action Recognition Problems

Here, we tackle two action recognition problems:

1) Gesture recognition: Our first problem is the classification of 20 gestures, taken from the Multi-modal Gesture Recognition Challenge 2013 (Chalearn) dataset [40]. This dataset is a newly released large video database of 13,858 gestures from a lexicon of 20 Italian gesture categories recorded with a Kinect camera, including audio,

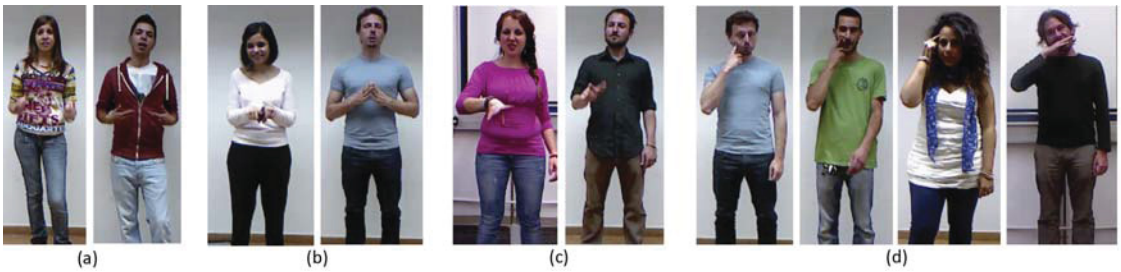


Figure 5.4: Some example gestures in the Chaleran dataset are very easy to be confused, even from human visual perception.
(a) *Che vuoi* vs. *Che due palle*. For the *Che vuoi* gesture, both hands are in front of the chest area, where for *Che due palle* gesture they are near the waist region. (b) *Vanno d'accordo* vs. *Cos hai combinato*: both hand positions are very close and with the same motion directions; (c) both gestures, *Si sono messid'accordo* and *non ce ne piu*, require hand rotations; (d) four gestures, *Furbo, seipazzo, buonissimo*, and *cosatifarei* are required with the finger pointing to the head area, which cannot be easily determined, even with human eyes.

skeletal model, user mask, RGB and depth images (see Figure 5.3). This dataset contains image sequences capturing 27 subjects performing natural communicative gestures and speaking in fluent Italian, and is divided into development, validation and test parts. Each sequence lasts between 1 and 2 minutes and contains between 8 and 20 gesture samples, around 1,800 frames. It provides audio, RGB, depth, and user mask videos for a sequence. Examples of RGB image sequences for some gestures are shown in Figure 5.4.

2) Human actions classification: The second problem is the recognition of 20 actions, taken from the MSRAction3D dataset [95]. This dataset is a well-known benchmark dataset for 3D action recognition and contains 20 actions, including *high arm wave, horizontal arm wave, hammer, hand catch, forward punch, high throw, draw x, draw tick, draw circle, hand clap, two hand wave, side-boxing, bend, forward kick, side kick, jogging, tennis swing, tennis serve, golf swing, pick up & throw*. Each action was performed 2 or 3 times by each subject. Skeleton joint data of each frame is available having a variety of motions related to arms, legs, torso, and their combinations. In total, there are 567 depth map sequences with a resolution of 320 $\times$ 240. Some examples of the depth sequences are shown in Figure 5.5.

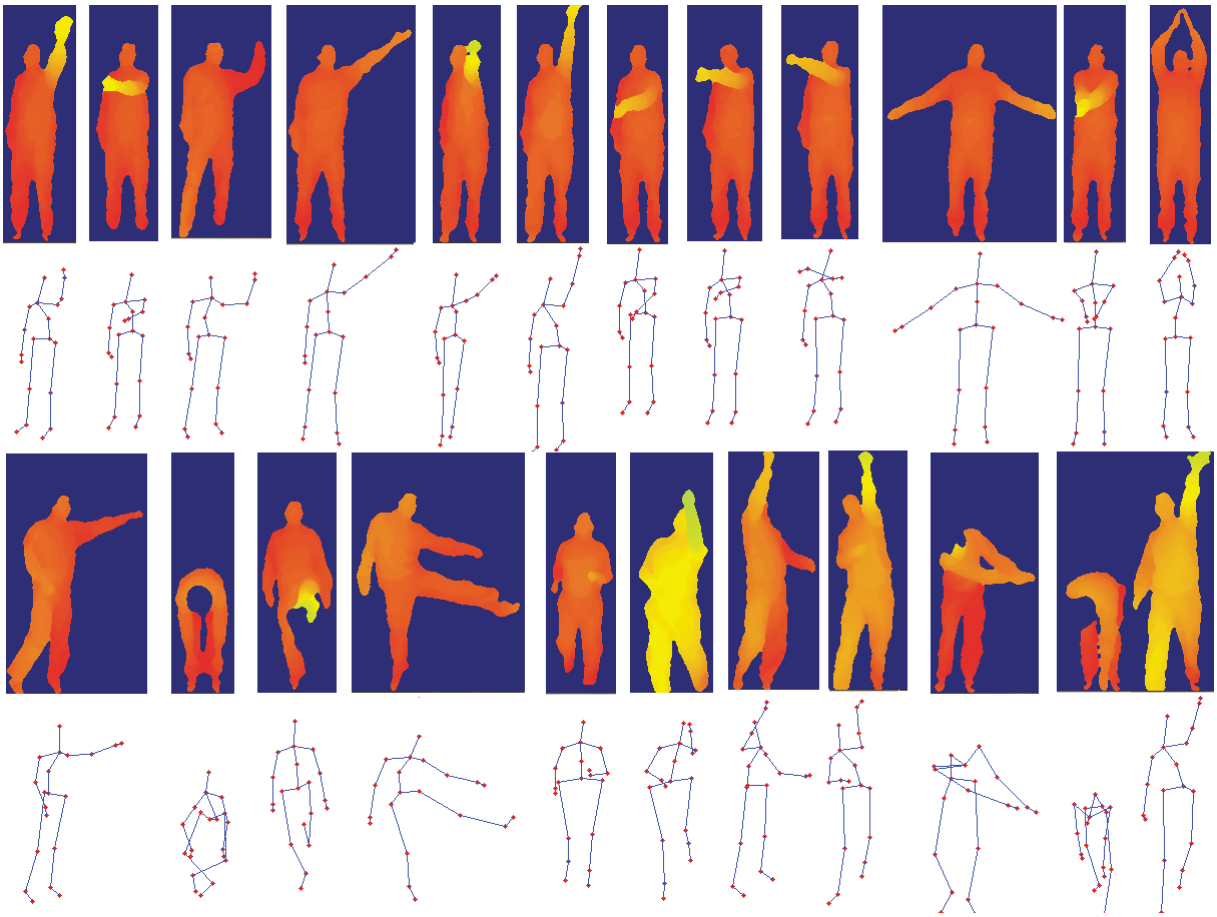


Figure 5.5: Example frames and the corresponding skeleton joints of 20 actions of MSRAction3D dataset.
The first two rows, from left to right: *high arm wave, horizontal arm wave, hammer, hand catch, forward punch, high throw, draw x, draw tick, draw circle, hand clap*, The last two rows, from left to right *two hand wave, side-boxing, bend, forward kick, side kick, jogging, tennis swing, tennis serve, golf swing, pick up & throw*

5.4 Action Learning Techniques

In this work, we have employed five different action recognition techniques using only positions of skeleton joints, described in the following subsections. Among them, the first three methods are the existing ones in the related literature and the last two methods are our proposed techniques.

1) Dynamic Time Warping + KNN

Dynamic Time Warping (DTW) is a well-known algorithm which aims to compare and align two temporal sequences, taking into account that sequences may vary in length (time) [128]. DTW employs the dynamic programming technique to find the

minimal distance between two time series, where sequences are warped by stretching or shrinking the time dimension. Although it was originally developed for speech recognition [138], it has also been employed in many other areas like handwriting recognition, econometrics, and action/video recognition.

Here, depending on the problem, the relative distance of suitable joint points is obtained at each frame. Then, given two actions represented by two multi dimensional time series, DTW calculates the distance between two actions. To classify an unlabeled test action (sample), its distance to all training samples is calculated. Consequently, the nearest neighbor algorithm should be employed for classification. Given a test action, we calculate its distance to all training actions using DTW, and the target of the closest sample is predicted as the target class.

2) Spatial and temporal differences among joints + Naive-Bayes-Nearest-Neighbor

In [176], visual features for activity recognition are computed based on the spatial and temporal differences between skeleton joints, named *EigenJoints* features. This feature set contains information about static posture, motion, and offset. In their method, a feature descriptor is generated for each frame. Therefore, each action has a different number of feature descriptors, depending on the number of frames. For classification, they employed the Naive Bayes Nearest Neighbor (NBNN) method, which is a very efficient method recently introduced for image classification [18].

For the next three sets of action descriptors, i.e. bag of visual words, wavelet coefficients of time series, and extreme features, all standard classifiers can be used in the classification stage. Here, we chose Support Vector Machine (SVM) with the Gaussian Kernel as the classification algorithm, since SVM is a state-of-the-art classifier and is commonly used for image and video recognition.

3) Bag of skeleton words + SVM

We also apply the BoW approach to 3D joint data, such that each visual word is constructed using a set of spatio-temporal descriptors of skeleton joint positions. In fact, each visual word, conceptually, is the cluster centre and represents a unique posture. We further choose words, i.e. clusters, with high discrimination capability. To this end, we compute the entropy of each word using the distribution of class

samples in the corresponding cluster, and select the words with top half entropy. As an example, an initial word might represent the neutral posture. Since each action usually includes frames showing the neutral posture, this word will be removed from the dictionary. Then, each action is represented using the frequency of each word in the codebook, obtaining a histogram of words for each action. These histograms can be used as the input features for a particular classifier, such as neural network or support vector machines. The generated feature set will be used for

4) Wavelet coefficients + SVM

Here, we propose a new technique for action description by using the multilevel wavelet decomposition technique based on the Mallat algorithm [103]. The Mallat algorithm is a classical scheme, known as two-channel subband coding in the signal processing community. When a signal passes through the filters, the low frequency components ,often called approximation in wavelet theory, and high frequency components (details) are emerged. The low frequencies are usually the most important part of a signal and represent its identity. The decomposition process can be repeated on the approximation components, so that one signal is broken down into many lower resolution components. This procedure is known as multilevel discrete wavelet transform (DWT) or simply multilevel decomposition.

In the context of action recognition using skeleton joint information, the relative position of each skeleton joint during the time is considered as a signal (time series). Then, we apply the multilevel wavelet decomposition method to extract low level wavelet coefficients.

In our implementation, three series are generated for each skeleton joint, according to the three dimensions of the real world position of the joint (X, Y, Z). The final feature vector for each action is generated by concatenating the wavelet coefficients of all time series.

5) Extreme features + SVM

Here, we propose a simple, but effective, action description technique. This method is based on the idea that for many short actions, like those in the benchmark datasets, only a very few salient postures can be a unique representative of the action. These postures are unique in the sense that the relative position, i.e. the distance, between

a set of skeleton joints will reach its extreme value.

In this method, given an action, we compute the pair distance between each appropriate joint point at each frame. The feature vector is then generated by taking the maximum and minimum value of each pair distance of all frames:

$$ExtremeFeatures = \bigcup \{min_t(PD^t_{ijk}), max_t(PD^t_{ijk})\} \tag{5.1}$$

where PD^t_{ijk} is the pairwise distance of ith and jth joint points during the time in the kth dimension (i.e. x, y, z). Depending on the problem, we may use all available points, e.g. the 20 points provided by the Kinect, or a subset of appropriate points. In this method, for each joint pair, six features will be generated, which are the maximum and minimum distances of two joints in X, Y, and Z dimensions.

Although, many advanced algorithms may consider the movement of skeleton joints during the time, many short actions have a few salient postures, and the global pair distance between joint points encodes the most informative salient features for those postures. In addition to its extremely fast computation, the proposed method has the advantage of generating a fixed-length feature vector for each action. Thus, it can be used with any type of classifier. The experimental results also show that even with large numbers of classes, this method achieves very high performances on two standard action recognition datasets.

For a better understanding of why this method works, in Figure 5.6 we illustrate some sample actions using sequences of frame images of the Chalearn gesture recognition dataset [40], together with each action's salient postures, depicted by red borders. Actions in this dataset are all performed by two hands. Therefore, we chose only five skeleton joint points as informative joints, including the *head, left elbow, left hand, right elbow*, and *right hand*. Therefore, there are 10 pairs and in total 60 features will be generated. As an example, the salient postures of *cheduepalle* action can be encoded by a set of extreme pair distances. In the salient posture, the followings extreme pair distances happen:

Maximum X of Left Elbow vs. Left Hand

Maximum X of head vs. Right Hand

Minimum X of Left Hand vs. Right Hand

Minimum X of Head vs. Right Hand

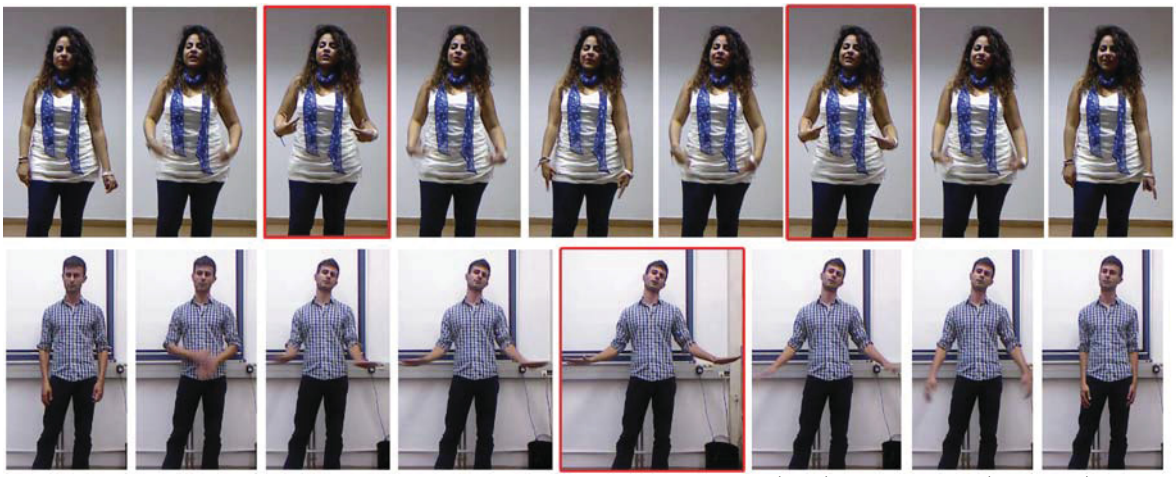


Figure 5.6: Sequences of frame images showing cheduepalle (top) and basta (bottom) gestures.

Minimum X of Left Elbow vs. Right Hand
Minimum Y of Right Elbow vs. Right Hand
Minimum X of Head vs. Left Elbow
Maximum Y of Left Hand vs. Right Elbow
Minimum Y of Head vs. Left Hand
Minimum Y of Left Elbow vs. Left Hand
Maximum X of Left Elbow vs. Right Elbow
Maximum Z of Head vs. Right Hand
Maximum Z of Left Elbow vs. Left Hand
Maximum Z of Left Elbow vs. Right Hand
Maximum Z of Right Elbow vs. Right Hand
Minimum X of Right Elbow vs. Right Hand
Minimum Z of Left Hand vs. Right Elbow
Minimum Y of Head vs. Right Elbow
Maximum Z of Head vs. Left Hand
Minimum Y of Left Elbow vs. Right Hand

The set of values of these salient features within the *cheduepalle* class encodes discriminating information and will lead to high recognition performance.

Once a set of features are extracted, they will be used as input to a classifier. In the classification stage, we used a modified version of the GA-SS-ECOC method, introduced in Chapter 3. In the modified version, instead of optimizing the whole three-dimensional

ECOC matrix, we find the most discriminative features for each binary problem using the Genetic algorithm feature selection. This modification results in much faster implementation.

5.5 Experiments

Here, we first present the results of our proposed action representation techniques and compare it with other methods. Then, the merits of using an ensemble classification framework is presented.

5.5.1 Classification by individual learners

For Chalearn dataset, the classification performance is obtained by means of stratified 5-fold cross-validation. For MSRAction3D dataset, most studies follow the experimental setting of Li et al. [95], such that they first divide the 20 actions into three subsets, each having 8 actions. For each subset, they perform three tests. In test one and two, 1/3 and 2/3 of the samples were used as training samples and the rest as testing samples. In the third test, half of the subjects are used as training and the rest subjects as testing. The experimental results on the first two tests are generally very promising, mainly more than 90% accuracy. On the third test, however, the recognition performance dramatically decreases. It shows that many of these methods do not have good generalization ability when a different subject is performing the action, even in the same environmental settings. In order to have more reliable results, we followed the same experimental setup of [164, 116]. In this setting, actors 1,3,5,7, and 9 are used for training and the rest for testing.

As mentioned before, we chose Support Vector Machine (SVM) with the Gaussian Kernel as the base classifier for the last three sets of action descriptors. Also, dynamic time warping is a distance-based method and therefore we employed the nearest neighbor algorithm for classification. In addition, in the method proposed in [176], each action has different number of feature descriptors. Therefore, standard classifiers, like SVM or neural networks, cannot be used as the classifier. Similar to their paper, we implemented and used the NBNN classifier.

The summaries of the results are reported in Table 5.1 and Table 5.2 for Chalearn and MSRAction3D datasets. In addition, the confusion matrices of the last four individual action learning techniques are presented in Figure 5.7 and Figure 5.8.

Table 5.1: Classification accuracy of individual action learning techniques on the Chalearn gesture dataset.

	Single classifier trained only on				
	EigenJoints +NBNN	DTW +KNN	BoVW +SVM	Wavelet Coeff. +SVM	Extreme Feat +SVM
5 classes	63.50	97.40	95.40	91.60	96.80
10 classes	58.70	89.80	88.00	86.60	88.40
15 classes	56.17	86.82	85.27	79.64	87.00
20 classes	54.30	77.85	73.05	70.40	73.65
Average	58.17	87.97	85.43	82.06	86.46

Table 5.2: Classification accuracy of individual action learning techniques on the MSRAction3D dataset.

	Single classifier trained only on				
	EigenJoints +NBNN	DTW +KNN	BoVW +SVM	Wavelet Coeff. +SVM	Extreme Feat +SVM
5 classes	72.97	95.95	83.78	81.62	96.95
10 classes	47.62	85.14	82.43	85.14	87.84
15 classes	44.14	81.60	77.30	63.19	78.46
20 classes	47.81	75.76	64.65	58.92	72.39
Average	53.14	84.61	77.04	72.22	83.91

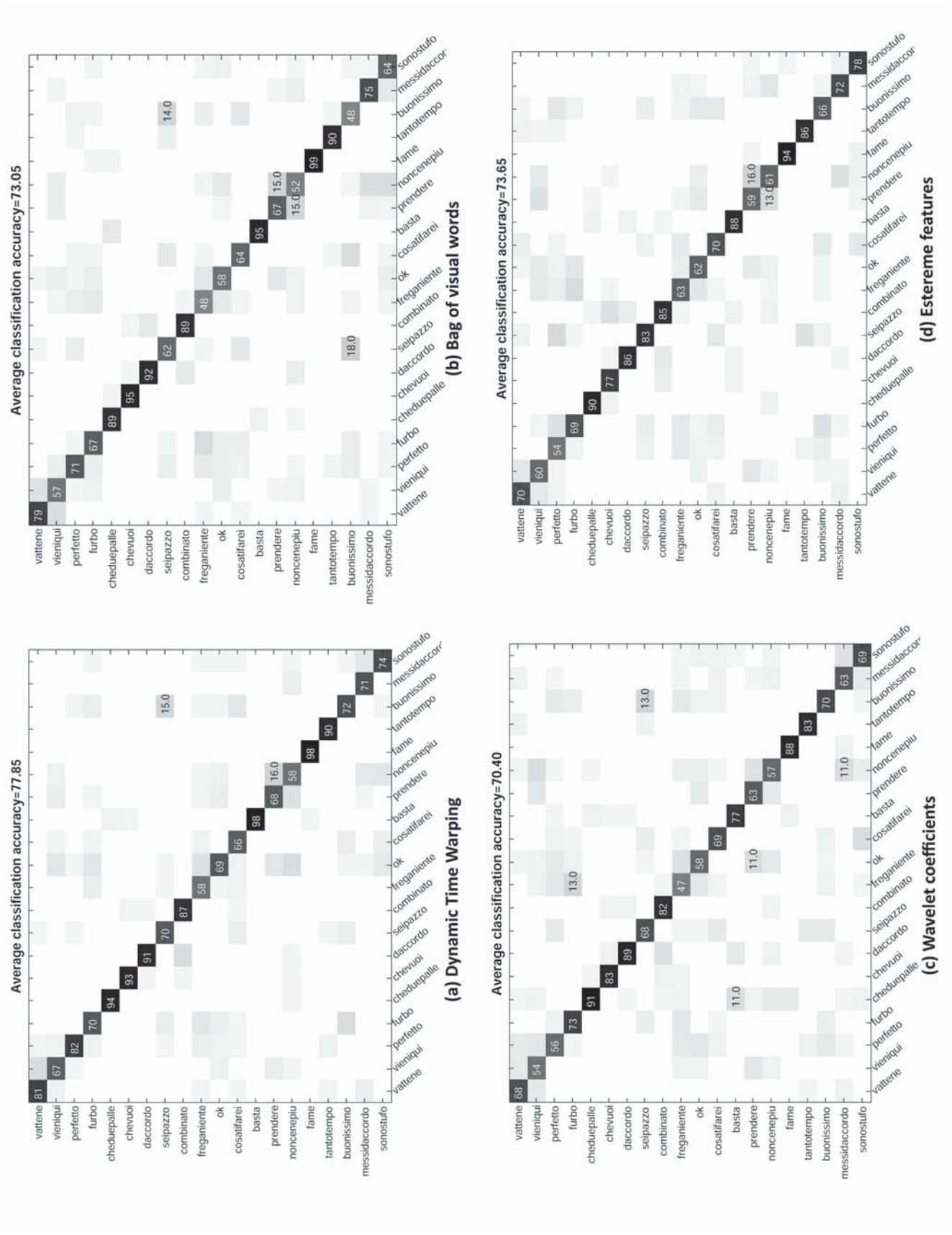


Figure 5.7: Confusion matrices of individual classifiers trained with different methods on the Chalearn dataset.

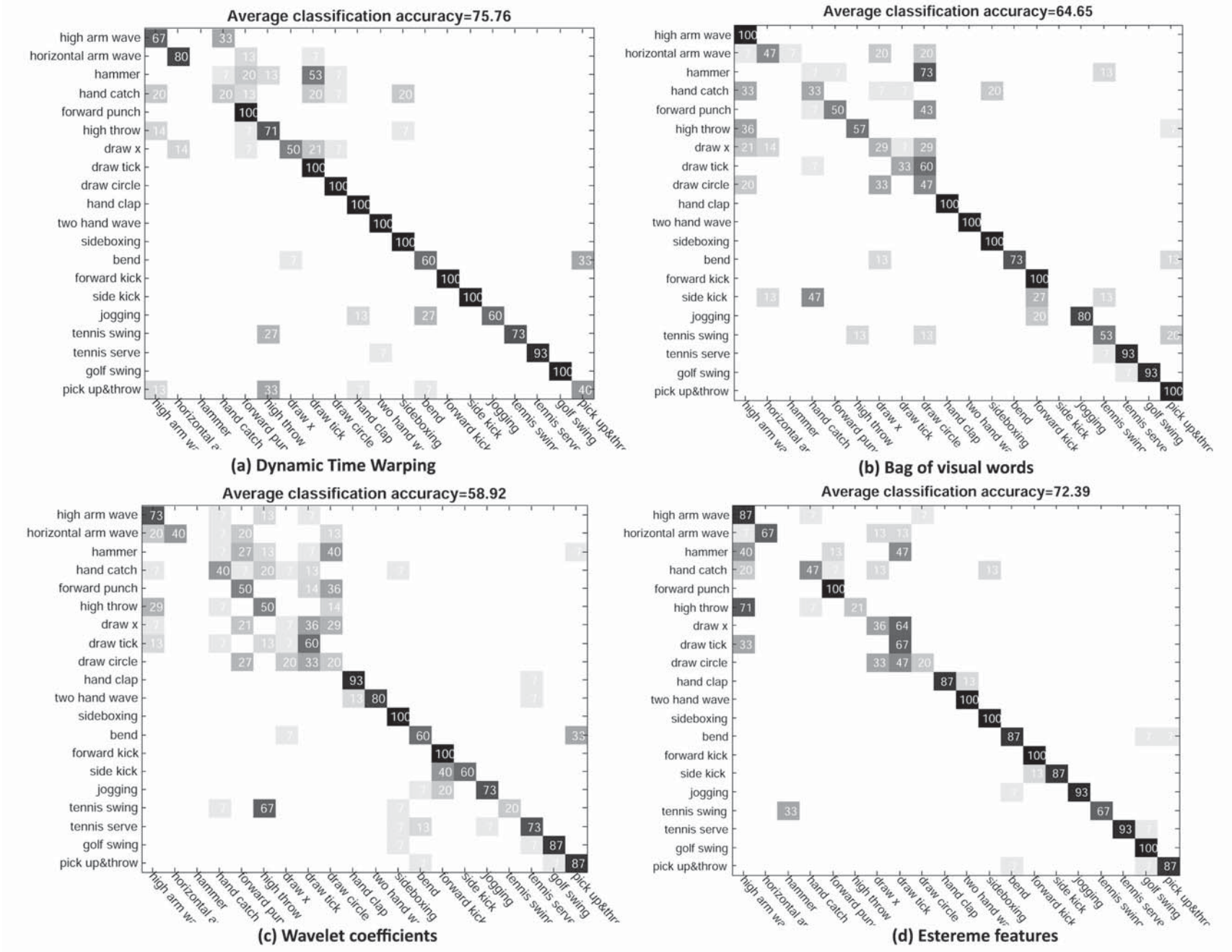


Figure 5.8: Confusion matrices of individual classifiers trained with different methods on the MSRAction3D dataset.

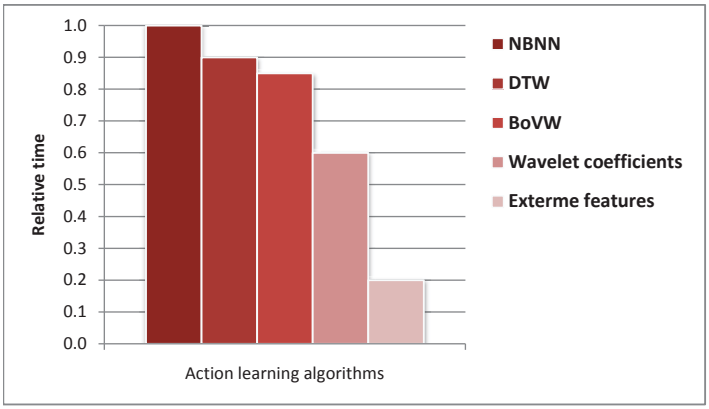


Figure 5.9: Average execution times of different action learning methods.

In these tables, the effectiveness of the proposed action description technique based on the extreme pair distance of joint points is notable. For both considered datasets, our extremely fast method achieved very high accuracy in many cases.

As an additional analysis, we compare the execution times of different action learning algorithms. In order to visually compare the results with a graph that is easy to interpret, the relative average time of different methods is presented in Figure 5.9.

5.5.2 Improving the recognition rate by the Dempster-Shafer fusion of individual learners

Here, we argue that there is a potential improvement in classification through classifier fusion by ensemble-based methods. The underlying rationale of the fusion approach is two fold. First, different learners employ varying structures of input descriptors/features to be trained. These varying structures cannot be attached and used by a single learner. Second, in an ensemble classification system, the combined efficiency of multiple classifiers can compensate for a deficiency in one classifier. Thus, combining the outputs of several learners can reduce the risk of an unfortunate selection of a poorly performing learner. This leads to having a more robust and general-applicable framework.

Motivated by this, we aim to employ different learners and efficiently fuse them. To this end, we propose the use of the Dempster-Shafer fusion method to effectively combine the outputs of different learners, taking into account the characteristics of a given test action and the behavior of ensemble learners in similar cases. We evaluate the performance of an ensemble of five learners, employed in the previous subsection. Figure 5.10 shows the framework of our ensemble classification system.

In the followings, we first explain the Dempster-Shafer fusion method and then provide

the results obtained by the ensemble classification framework.

Dempster-Shafer fusion method

Inspired by the Dempster-Shafer (DS) theory of evidence [35], a combination method is proposed in [133], which is commonly known as the Dempster-Shafer fusion method. By interpreting the output of a classifier as a measure of evidence provided by the source that generated the training data, the DS method fuses an ensemble of classifiers.

Let $x \in R^n$ be a feature vector and $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$ be the set of class labels. Each classifier h_i in the ensemble $H = \{h_1, h_2, \ldots, h_L\}$ outputs c degrees of support. Without loss of generality, we can assume that all c degrees are in the interval $[0, 1]$. The support that classifier h_i, gives to the hypothesis that $\mathbf{x}$ comes from class ω_j is denoted by $d_{i,j}(x)$. Clearly, the larger the support, the more likely the class label ω_j. The L classifier outputs for a particular instance $\mathbf{x}$ can be organized in a decision profile, $DP(x)$, as the following matrix [88]:

$$DP(x) = \begin{pmatrix} d_{1,1}(x) & \cdots & d_{1,j}(x) & \cdots & d_{1,c}(x) \\ \vdots & & \vdots & & \vdots \\ d_{i,1}(x) & \cdots & d_{i,j}(x) & \cdots & d_{i,c}(x) \\ \vdots & & \vdots & & \vdots \\ d_{L,1}(x) & \cdots & d_{L,j}(x) & \cdots & d_{L,c}(x) \end{pmatrix}$$

The Dempster-Shafer fusion method uses the decision profile to find the overall support for each class and subsequently labels the instance $\mathbf{x}$ in the class with the largest support. In order to obtain the ensemble decision based on DS fusion method, first, the c decision templates, $DT_1, \ldots, DT_c$, are built from the training data. Roughly speaking, decision templates are the most typical decision profile for each class ω_j. For each test sample, $\mathbf{x}$, the DS method compare the decision profile, $DP(x)$, with decision templates. The closest match will label $\mathbf{x}$. In order to predict the target class of each test sample, the following steps are performed [88][133]:

1. Build decision templates: For $j = 1, \ldots, c$, calculate the means of the decision profiles for all training samples belonging to ω_j. Call the mean a decision template of class ω_j, DT_j.

$$DT_j = \frac{1}{N_j} \sum_{z_k \in \omega_j} DP(z_k) \tag{5.2}$$

where N_j in the number of training samples belong to ω_j.

2. Calculate the proximity: Let DT_j^i denote the ith row of the decision template DT_j, and D_i the output of the ith classifier, that is, the ith row of the decision profile $DP(x)$. Instead of similarity, we now calculate proximity Φ, between DT_j^i and the output of classifier D_i for the test sample x:

$$\Phi_{j,i}(x) = \frac{(1 + \|DT_j^i - D_i(x)\|)^{-1}}{\sum_{k=1}^{c}(1 + \|DT_j^i - D_i(x)\|)^{-1}} \tag{5.3}$$

where $\|.\|$ is a matrix norm.

3. Compute belief degrees: Using Eq. (2), calculate for each class $j = 1, \ldots, c$ and for each classifier $i = 1, \ldots, L$, the following belief degrees, or evidence, that the ith classifier is correctly identifying sample $\mathbf{x}$ into class ω_j:

$$b_j(D_i(x)) \frac{\Phi_{j,i}(x) \prod_{k \neq j}(1 - \Phi_{k,i}(x))}{1 - \Phi_{j,i}(x)[1 - \prod_{k \neq j}(1 - \Phi_{k,i}(x))]} \tag{5.4}$$

4. Final decision based on class support: Once the belief degrees are achieved for each source (classifier), they can be combined by Dempster's rule of combination, which simply states that the evidences (belief degree) from each source should be multiplied to obtain the final support for each class:

$$\mu_j(x) = K \prod_{i=1} b_j(D_i(x)),\ j = 1, \ldots, c$$

where K is a normalizing constant ensuring that the total support for ω_j from all classifiers is 1. The DS combiner gives a preference to class with largest $\mu_j(x)$.

It is worth mentioning that classification using DTW and NBNN are based on the nearest neighbour approach. Therefore, they generally do not provide the probabilistic outputs for train and test samples, which are necessary for DS fusion method. In order to tackle this problem, given a test sample, we used the normalized distances of nearest neighbours of samples of each class as the estimation of the probability of the corresponding class. Also, to have probabilistic outputs for training samples, we employed 2-fold cross validation on the training samples, such that in each fold, about half of the training samples are used as training and the others as validations samples, and generated the desired probabilistic

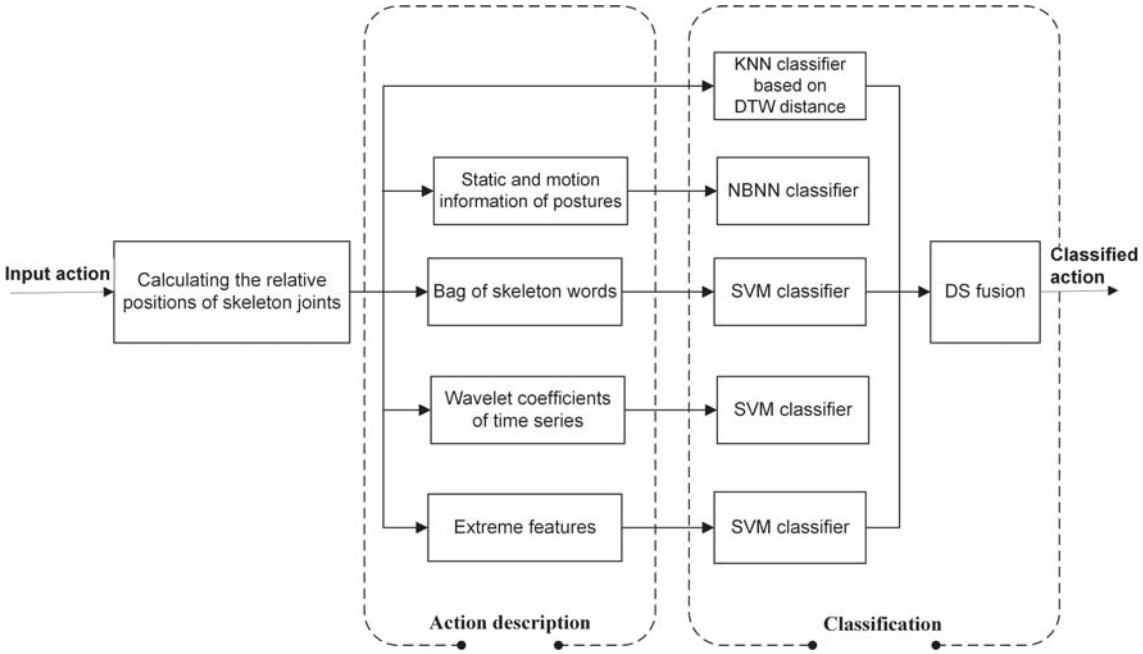


Figure 5.10: The framework of the proposed action classification system based on the Dempster-Shafer fusion of multiple classifiers.

Table 5.3: Classification accuracy of single and fused classifiers on the Chalearn and MSRAction3D datasets.

	Single classifier trained only on					**DS fusion**
	EigenJoints + NBNN	DTW +KNN	BoVW +SVM	Wavelet Coeff. +SVM	Extreme Feat. +SVM	
Chalearn	54.30	77.85	73.05	70.40	73.65	**82.60**
MSRAction3D	47.81	75.76	64.65	58.92	72.39	**80.81**

outputs for each training sample.

The ensemble classification results

The results of individual classifiers along with the ensemble fused system are presented in Table 5.3 for both datasets. The results are based on the same experimental settings of the last section. In addition, the confusion matrix of the ensemble classification system for both datasets are demonstrated in Figure 7.3. It is important to note the superiority of the fused results in comparison with the individual classifier. Since the learning method based on the NBNN has a relatively very low recognition rate, we omitted its outputs in the fusion framework. The results are quite promising, considering the fact that the skeleton tracker sometimes fails and the tracked joint positions are quite noisy.

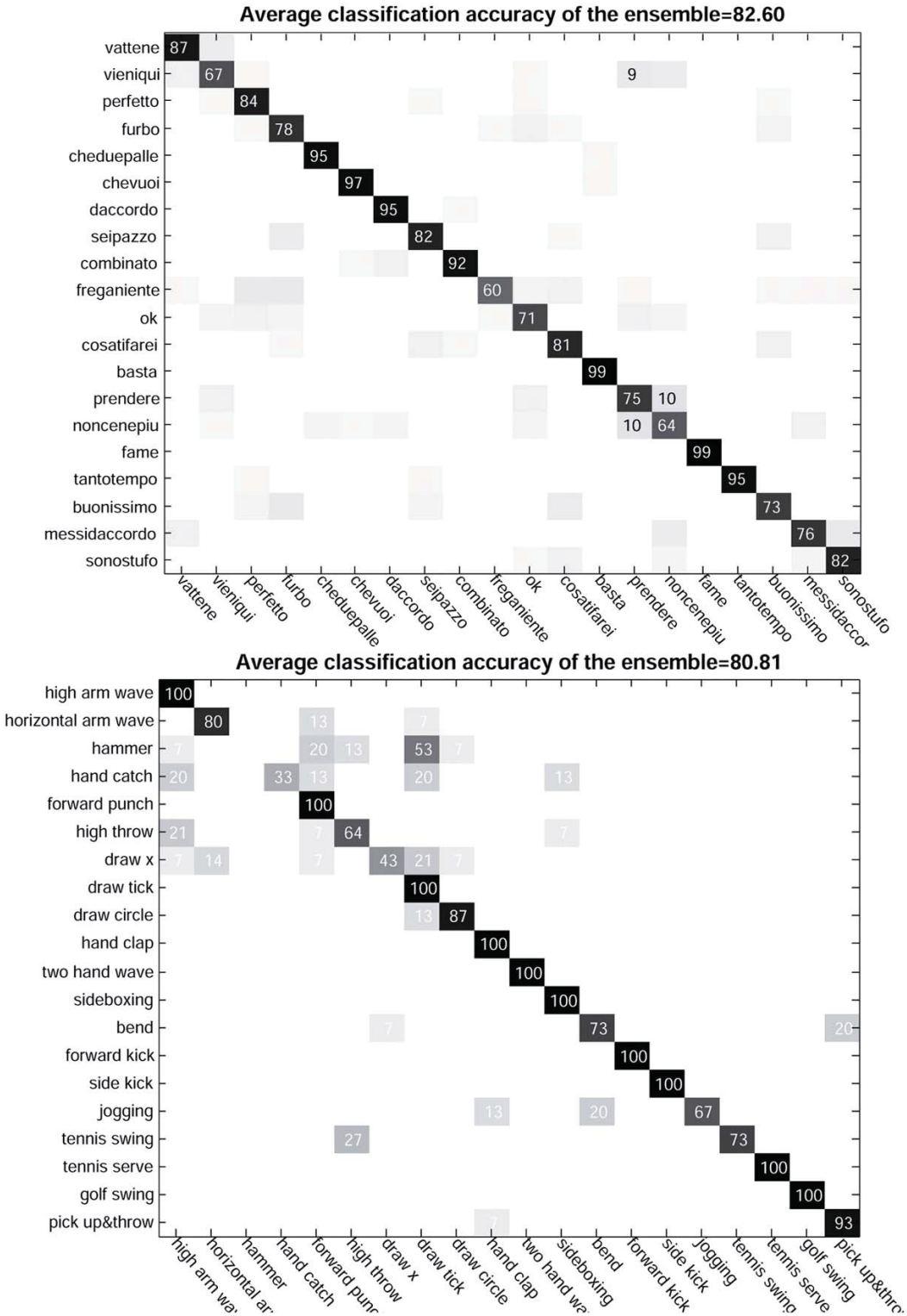


Figure 5.11: Confusion matrices of the ensemble classification system on the Chalearn (top) and MSRAction3D datasets (bottom).

Table 5.4: Comparing classification accuracy of our ensemble framework with the state-of-the-art methods on the MSRAction3D dataset.

Method	**Accuracy**
Recurrent Neural Network [105]	42.5
Hidden Markov Model [101]	54
Action Graph on Bag of 3D Points [95]	74.7
HOG 3D [83]	81.43
HON4D [116]	85.85
Dollar + BOW [39]	72.40
STIP [91] + BOW	69.57
Vieira et al. [161]	78.20
Our method	**80.81**

We then compare our ensemble classification method on MSRAction3D dataset with the state-of-the-art methods on the cross-subject test setting [95]. Table 5.4 shows the accuracy of our method and the rival methods on this dataset with the same experimental settings. Some of the methods in this table, like HOG 3D [83] and HON4D [116], use depth data in addition to skeleton joint information. However, processing sequences of depth images is much more computationally intensive. Even though the accuracy of the proposed framework is slightly lower than some other methods, the advantage of our method is its fast implementation, makes it feasible for real-time applications.

5.6 Summary

This chapter presented an ensemble classification framework to address certain action/gesture recognition problems. We designed a set of classifiers, each trained over a type of feature. We focused on feature spaces defined by a 3D skeletal model of the human body and proposed two simple, yet effective, feature representations for this problem. The overall performance of the ensemble of classifiers was improved by fusing the classifiers using the Dempster-Shafer combination theory. We compared the classification results of the individual classifiers with those obtained from fusing the classifiers by the Dempster-Shafer combination method on two public datasets, showing significant performance improvements of the proposed methodology. We also showed performance improvements in relation to the state of the art results on the considered data sets. In conclusion, we found that using ensemble methods for human actions and gestures classification is an effective approach.

Chapter 6

Support Vector Machines with Time Series Distance Kernels for Action Classification

6.1 Introduction

In the previous chapter, we addressed the problem of human action classification by employing spatio-temporal information of skeleton data, i.e. the real positions of body joints over the time. More specifically, we used the 3D trajectories of dominant body joints, obtained by the Kinect camera. From the classification point of view, these trajectories may be considered as multi-dimensional time series. The traditional recognition technique in the literature is based on time series dis(similarity) measures (such as Dynamic Time Warping). For these general dis(similarity) measures, k-nearest neighbor algorithms are a natural choice.

In practice, given two actions represented by two multi dimensional time series, a time series distance measure calculates the distance between two actions. To classify an unlabeled test action (sample), its distance to all training samples is calculated. Consequently, the nearest neighbor algorithm is employed for classification. Given a test action, we calculate its distance to all training actions , e.g. by using DTW, and the target of the closest sample is predicted as the target class.

In general, the k-NN classification algorithm works reasonably well; but is known to be sensitive to noise and outliers. Since SVMs often outperform k-NNs on many practical classification problems where a natural choice of positive semidefinite (PSD) kernels exists, it is desirable to extend the applicability of kernel SVMs.

In our action classification problem, however, time series distances measures are generally non-PSD kernels and basic SVM formulations are not directly applicable. To include non-PSD kernels in SVM, several ad-hoc strategies have been proposed. The straightforward strategy is to simply overlook the fact that the kernel should be non-PSD. In this case, the existence of a Reproducing Kernel Hilbert Space is not guaranteed [142] and it is no longer clear what is going to be optimized.

Another strategy, which has been applied in our work, is based on *pairwise proximity*

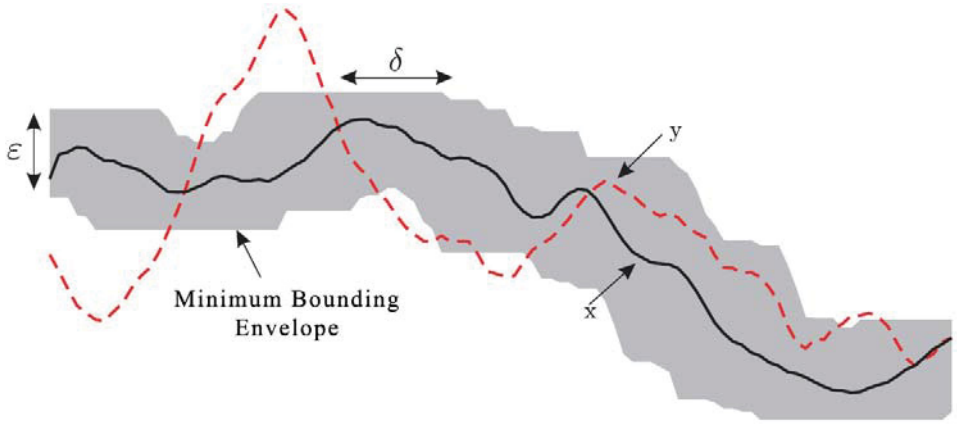


Figure 6.1: Matching within δ in time and ϵ in space. Everything outside the bounding envelope can never be matched (Reprinted from [59]).

function SVM (ppfSVM) [60]. This strategy involves the construction of a set of inputs such that each sample is represented with its dis(similarity) to all other samples in the dataset. The ppfSVM is related to the arbitrary kernel SVM, a special case of the generalized Support Vector Machines [104]. The name is due to the fact that no restrictions such as positive semi-definiteness, differentiability or continuity are put on the kernel function.

In this chapter, we investigate the effectiveness of this strategy for human action classification when the pairwise similarities are based on time-series distances measures. More specifically, we demonstrate the effectiveness of two trajectory-based distances measures - including Longest Common Subsequence (LCSS) and Dynamic Time Warping (DTW) as well as their derivatives- as SVM kernel functions. We build an ensemble of four ppfSVMs using the computed kernels and combine them using the product fusion technique. The experimental results on two benchmark datasets prove the outperformance of the proposed method compared to the state-of-the-art techniques.

The rest of the chapter is organized as follows: Section 6.2 briefly introduces LCSS and DTW. Section 6.3 presents our methodology for action recognition. Section 6.4 evaluates the proposed method and Section 6.5 concludes the chapter.

6.2 Related Work

6.2.1 Longest Common Subsequence (LCSS)

The longest common subsequence dissimilarity measure is a variation of the edit dissimilarity measure, initially used in speech recognition. The underlying idea is to match two sequences by allowing them to stretch, without rearranging the sequence of the elements but allowing

some elements to be unmatched or left out (e.g., outliers). Roughly speaking, LCSS counts the number of pairs of points from two sequences that match. The LCSS measure has two parameters, δ and ϵ, as shown in Fig. 6.1. The constant δ controls how far in time we can go in order to match a given point from one trajectory to a point in another trajectory. This parameter is a warping threshold and controls the window size for matching a given point from one trajectory to a point in another one, which is usually set to a percentage of the sequence length. The constant $0 < \epsilon < 1$ is the matching threshold: two points from two sequences are considered to match if their distance is less than ϵ.

Longest common subsequences of the time series x and y of length n and m are recursively defined as follows:

$$L(i,j) = \begin{cases} 0 & \text{for } i = 0 \\ 0 & \text{for } j = 0 \\ 1 + L(i-1, j-1) & \text{for } |x_i - y_j| < \epsilon \\ & \text{and } |i - j| \leq \delta \\ max(L(i-1,j), L(i,j-1)) & \text{in other cases} \end{cases}$$

$L(n,m)$ is the similarity between x and y, because it corresponds to the length of the longest common subsequence of elements between time series. The dissimilarity between x and y has been defined as follows:

$$LCSS(x,y) = \frac{(n + m - 2L(n,m))}{(n+m)} \tag{6.1}$$

6.2.2 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) is a well-known algorithm which aims to compare and align two temporal sequences, taking into account that sequences may vary in length (time) [128]. DTW employs the dynamic programming technique to find the minimal distance between two time series, where sequences are warped by stretching or shrinking the time dimension. Although it was originally developed for speech recognition, it has also been employed in many other areas like handwriting recognition, econometrics, and action recognition.

An alignment between two time series can be represented by a warping path which minimizes the cumulative distance. The DTW distance between time series x and y of

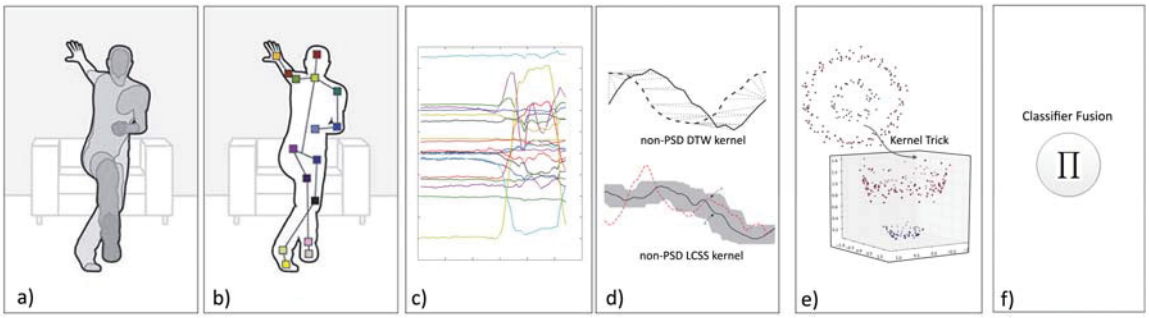


Figure 6.2: The framework of the proposed *Time Series based Kernel SVM for action classification*; a) an initial depth map; b) positions of 20 joints obtained by Kinect [144]; c) extract features: relative trajectories of joints over the time; d) compute non-PSD kernels using DTW and LCSS; e) Train ppfSVMs; f) classifier fusion.

length n and m will be recursively defined as:

$$DTW(i,j) = d(i,j) + min \begin{cases} DTW(i,j-1) \\ DTW(i-1,j) \\ DTW(i-1,j-1) \end{cases}$$

Here, $d(i,j)$ is the square Euclidean distance of x_i and y_j.

6.3 Time Series based Kernel SVM

The proposed algorithm works as follows:

1. **Feature extraction:** Given a depth image, 20 joints of the human body can be tracked by the skeleton tracker (Fig. 6.2. a & b). In frame t, the position of each joint k is uniquely defined by three coordinates $P_k(t) = [x_i(t), y_i(t), z_i(t)]$. Instead of using the positions of joints, we employ the relative position of each joint to the torso at each frame, as more discriminative and intuitive 3D joint features (Fig. 6.2. c).

2. **Compute non-PSD kernels:** We compute the non-PSD kernels based on pairwise distance of each normalized 3D trajectory to other trajectories, using LCSS and DTW (Fig. 6.2. d), as described in the next subsection.

3. **Classification:** As described in subsection 6.3.2, we train four ppfSVMs using the computed kernels (Fig. 6.2.e) and simply fuse these classifiers (Fig. 6.2.f).

6.3.1 Kernel from pairwise data

Given labeled training data of the form $\{(x_i, y_i)\}_{i=1}^{m}$, with $y_i \in \{-1, +1\}$ [1], the standard form of SVM finds a hyperplane which best separates the data by minimizing a constrained optimization problem:

$$\begin{aligned} \tau(w, \xi) =& \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to: } & y_i((w.x_i) + b) + \xi_i \geq 1 \\ & \xi_i \geq 0 \end{aligned} \tag{6.2}$$

where ξ_i are slack variables and $C > 0$ is the tradeoff between a large margin and a small error penalty.

The cornerstone of SVM is that non-linear decision boundaries can be learnt using the so called 'kernel trick'. A *Kernel* is a function $\mathcal{K} : \mathbf{X} \times \mathbf{X} \mapsto \mathbb{R}$, such that for all x_i, $i \in \{1, \ldots, m\}$ yields to a symmetric positive semi-definite (PSD) matrix K, where $K_{ij} = \kappa(x_i, x_j)$. Indeed, the kernel function implicitly maps their inputs into high-dimensional *feature spaces*, $x \mapsto \Phi(x)$. Two common kernel functions are the Gaussian Kernel and the Linear kernel.

In the dual formulation, the SVM algorithm maximizes:

$$\begin{aligned} W(a) =& \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{ij}\alpha_i\alpha_j y_i y_j \kappa(x_i, x_j) \\ \text{subject to: } & 0 \leq \alpha_i \leq C \text{ and } \sum_{\alpha_i y_i} = 0 \end{aligned} \tag{6.3}$$

The decision function is given by:

$$f(x) = sign\left(\sum_{i=1}^{m} y_i\alpha_i\kappa(x, x_i) + b\right) \tag{6.4}$$

where the threshold b is defined as:

$$b = y_i - \sum_{i=1}^{m} y_i\alpha_i\kappa(x_i, x_j) \tag{6.5}$$

[1] In our formulation, the input samples, x_i, are not restricted to be a subset of R^n and can be any set, e.g. set of images or videos.

In our action classification problem, however, time series distances measures are generally non-PSD kernels and basic SVM formulations are not directly applicable. To deal with this problem, we follow the strategy proposed in [60], which can be applied to general pairwise similarity measures. This strategy involves the construction of a set of inputs such that each sample is represented with its dis(similarity) to all other samples in the dataset. The basic SVM is then applied to the transformed data in the usual way. As a consequence, sparsity of the solution may be lost.

According to [60], it is assumed that instead of a standard kernel function, all that is available is a proximity function, $P : \mathbf{X} \times \mathbf{X} \mapsto R$. No restrictions are placed on the function P, not symmetry nor even continuity. The mapping $\Phi(x)$ is defined by:

$$\Phi(x) : x \mapsto (P(x, x_1), P(x, x_2), \ldots, P(x, x_m)^T \tag{6.6}$$

where $x_i, i = 1, \ldots, m$ are the examples in dataset. Here, we represent each sample x_i by $x_i = \Phi_m(x_i)$ i.e. an m-dimensional vector containing proximities to all other samples in the dataset. Let P denote the $m \times m$ matrix with entries $P(x_i, x_j), i, j \in \{1, \ldots, m\}$. Using the linear kernel on this data representation, the resulting kernel matrix becomes $K = PP^T$. In this case the decision rule (3) simplifies to

$$f(x) = sign\left(\sum_{i=1}^{m} y_i \alpha_i P \Phi_m(x) + b \right) \tag{6.7}$$

All elements of $\Phi_m(x_i)$ must be computed when classifying a point x.

In this study, kernels from pairwise data is obtained by pairwise time-series distance measures, including DTW and LCSS measures. In addition, as described in the next subsections, we also calculate the pairwise distances using the derivatives of these two time-series measures.

Derivatives of Time Series Distance Measures

Despite the success of time series dis(similarity) measures, i.e. DTW and LCSS, they may fail in some situations. For example, since the DTW algorithm aims to explain variability in the Y-axis by warping the X-axis, it may results in unintuitive alignments where a single point on one sequence maps onto a large subsection of the other sequence; which is referred to as *"singularity"* in the related literature [80]. Also, they may fail to find obvious, natural alignments of two time series simply because a feature (i.e peak, valley, inflection point,

plateau etc.) in one series is slightly higher or lower than its corresponding feature in the other time series.

To address such problems, the derivatives versions of DTW and LCSS are also employed in this work in order to enhance the level of feature representation. These modified versions are called *Derivative DTW (DDTW)* and *Derivative LCSS (DLCSS)*. More formally,

$$DDTW \triangleq DTW(\nabla x, \nabla y) \tag{6.8}$$

$$DLCSS \triangleq LCSS(\nabla x, \nabla y) \tag{6.9}$$

where ∇x and ∇y are estimated derivatives of two time series x and y, respectively.

6.3.2 Classifier fusion

In order to utilize the information encoded in the function values of time series and values of their first derivatives, we employed a simple ensemble classification framework [8]. In this framework, four SVMs are trained with four different types of kernels, i.e. DTW, DDTW, LCSS, DLCSS. In testing phase, the class of each sample, $\boldsymbol{x}$, is determined by:

$$c(\boldsymbol{x}) = \arg\max_{i} \prod_{t=1}^{4} w_t \mu_{t,i}(\mathbf{x}), i = 1, \ldots, N_c \tag{6.10}$$

where $c(\boldsymbol{x})$ is the ensemble class prediction, N_c is the number of classes, and $\mu_{t,i}(\boldsymbol{x}) \in [0, 1]$ represents the support given by the tth classifier to the ith class. w_t represents the weight of tth classifier, which is based on the classifier's accuracy on the training data.

6.4 Experiments

Here, we present the experimental details of evaluation, including the datasets used, settings of the experiments, as well as the obtained results. The codes was implemented in C/C++ with an interface in Matlab and is available upon request.

6.4.1 Datasets

We evaluated our framework on three public benchmark datasets: MSRAction3D [95], Cornel activity dataset (CAD-60) [148], and the Multi-modal Gesture Recognition Challenge 2013 (Chalearn) [40].

MSRAction3D dataset: This dataset [95] is a well-known benchmark dataset for 3D action recognition. This dataset contains 20 actions, including *high arm wave, horizontal*

Figure 6.3: Examples of depth maps from the CAD-60 dataset.

arm wave, hammer, hand catch, forward punch, high throw, draw x, draw tick, draw circle, hand clap, two hand wave, side-boxing, bend, forward kick, side kick, jogging, tennis swing, tennis serve, golf swing, pick up & throw. Each action was performed 2 or 3 times by each subject. Skeleton joint data of each frame is available having a variety of motions related to arms, legs, torso, and their combinations. In total, there are 567 depth map sequences with a resolution of 320 $\times$ 240. Some examples of the depth sequences are shown in the previous chapter in Figure 5.5.

Chalearn dataset: This dataset is a newly released large video database of 13,858 gestures from a lexicon of 20 Italian gesture categories recorded with a Kinect camera, including audio, skeletal model, user mask, RGB and depth images [40]. It contains image sequences capturing 27 subjects performing natural communicative gestures and speaking in fluent Italian, and is divided into development, validation and test parts. We conducted our experiments on the depth images of development and validation samples which contains

11,116 gestures across over 680 depth sequences. Each sequence lasts between 1 and 2 minutes and contains between 8 and 20 gesture samples, around 1,800 frames. Examples of RGB image sequences for some gestures are shown in the previous chapter in Figure 5.4.

CAD-60 dataset: This dataset [148] contains 60 RGB-D videos collected by a Kinect sensor with the distance ranges from 1.2m to 3.5m, the resolution of the depth sequences is 640 480, and captured at 15 fps. There are 12 actions performed by 4 different subjects (two male and two female, one of them being left-handed) in 5 different environments: office, kitchen, bedroom, bathroom, and living room. The 12 activities are: *rinsing mouth, brushing teeth, wearing contact lens, talking on the phone, drinking water, opening pill container, cooking (chopping), cooking (stirring), talking on couch, relaxing on couch, writing on whiteboard, and working on computer*. For each action, around 45 seconds of data for each person have been acquired, so resulting in more than 30 minutes of video recording. All the RGB, depth and skeleton data are provided in this dataset. Figure 6.3 shows some example depth images from this dataset.

6.4.2 Classification results

For Chalearn dataset, the classification performance is obtained by means of stratified 5-fold cross-validation. For MSRAction3D dataset, many studies follow the experimental setting of Li et al. [95], such that they first divide the 20 actions into three subsets, each having 8 actions. For each subset, they perform three tests. In test one and two, 1/3 and 2/3 of the samples were used as training samples and the rest as testing samples. In the third test, half of the subjects are used as training and the rest subjects as testing. The experimental results on the first two tests are generally very promising, more than 90% accuracy. On the third test, however, the recognition performance dramatically decreases. It shows that many of these methods do not have good generalization ability when a different subject is performing the action, even in the same environmental settings. In order to have more reliable results, we followed the same experimental setup of [164, 116]. In this setting, actors 1,3,5,7, and 9 are used for training and the rest for testing.

The summaries of the results are reported in Table 6.1, Table 6.2, and Table 6.3 for Chalearn, MSRAction3D, and CAD-60 datasets. In these tables, accuracies of traditional k-NN-based techniques using DTW and LCSS distance measures along with the corresponding accuracies using combined ppfSVMs are reported. It is important to note the superiority of the results in comparison with the traditional kNN-based classifiers. The result are quite promising, considering the fact that the skeleton tracker sometimes fails and the tracked

Table 6.1: Classification accuracy of different learning strategies on the Chalearn gesture dataset.

	DTW	**DDTW**	**LCSS**	**DLCSS**	**Product fusion**
Kernel SVM	69.30	71.85	73.05	73.40	**83.42**
kNN	61.11	63.15	67.21	69.18	–

Table 6.2: Classification accuracy of different learning strategies on the MSRAction3D dataset.

	DTW	**DDTW**	**LCSS**	**DLCSS**	**Product fusion**
Kernel SVM	80.47	83.84	75.76	76.77	**90.57**
kNN	75.42	77.78	72.05	65.66	–

joint positions are quite noisy.

We then compare our classification results on MSRAction3D and CAD-60 datasets with state-of-the-art methods [2]. Table 6.4 shows the accuracy of our method, as well as the rival methods on these datasets based on the cross-subject test setting. As can be seen, most studies use depth data in addition to skeleton data; and a few of them have better performance than ours, such as [112] and [125]. However, processing sequences of depth maps is much more computationally intensive. Even though the accuracy of the proposed framework is slightly less than those methods, the advantage of our method is its fast implementation and also do not need fine-tuning of many parameters, which makes it feasible for real-time applications. The training phase of MSRAction3D dataset (including Kernel computation) takes less than a second with a Corei7 CPU and 8 GB of RAM. Most importantly, since kernel computation is based on pairwise distances between samples, it can be easily conducted in parallel. This way, the training phase can be fast on large datasets as well.

The results provided in Table 6.1 to Table 6.4 demonstrate the superiority of the proposed methodology. By only considering the skeleton data, the obtained results outperform the best accuracies on MSRAction3D and CAD-60 datasets. Considering the fact that we have only employed the skeleton data, not depth sequences, the results are promising.

6.5 Summary

In this chapter, we tackled the problem of human action classification using the 3D trajectories of body joint positions over the time. To do that, we utilized two time series distance measures, including Dynamic Time Warping and Longest Common subsequences, as well

[2]Some papers do not follow the standard cross subject settings (e.x. they divide the 20 actions into three subsets, each having 8 actions. Therefore, we do not compare our results with those papers

Table 6.3: Classification accuracy of different learning strategies on the CAD-60 dataset.

	DTW	**DDTW**	**LCSS**	**DLCSS**	**Product fusion**
Kernel SVM	73.33	75.00	71.67	70.00	**76.67**
kNN	68.33	68.33	65.00	66.67	–

Table 6.4: Comparing classification accuracy of our methodology with the state-of-the-art methods on the MSRAction3D and CAD-60 datasets.

MSRAction3D	
	Accuracy
Studies employed depth data	
Action Graph [95]	74.70
HON4D [116]	85.85
Vieira et al. [161]	78.20
Random Occupancy Patterns [163]	86.50
HOPC [125]	91.64
JAS(Cosine)+MaxMin+HOG2 [112]	94.84
DMM-LBP-FF [26]	87.90
Studies employed only skeleton data	
Actionlet Ensemble [165]	88.20
Histogram of 3D Joint [174]	78.97
GB-RBM & HMM [110]	80.20
Points in a Lie Group [160]	89.48
Ensemble classification [10]	84.85
Proposed method	**90.57**
CAD-60	
	Accuracy
Studies employed depth data	
MTO-Sparse coding [109]	65.30
Studies employed only skeleton data	
Actionlet Ensemble [165]	74.70
Sung et al. (2012) [148]	51.30
Proposed method	**76.67**

as their derivatives. However, instead of employing these general measures as a distance measure for k-NN, we transformed these measures using the pairwise proximity function in order to be used for the powerful SVM classification algorithm. Comparing the recognition results of the proposed methods with state-of-the-art techniques on two action recognition datasets showed significant performance improvements. Remarkably, we obtained 90.57% accuracy on the well-known MSRAction3D dataset using only 3D trajectories of body joints obtained by Kinect.

Chapter 7

Locality Regularized Group Sparse Coding for Action Recognition

7.1 Introduction

The bag of visual words (BoVW) framework is one of the most widely used approaches for image/video representation and recognition, which has shown its effectiveness in many applications [145, 78]. Despite remarkable progress, it remains challenges concerning the efficiency and effectiveness of the framework on various domains.

The pipeline of BoVW generally consists of four stages: (i) feature extraction, (ii) codebook generation, (iii) feature encoding, and (iv) pooling and normalization. Given a training dataset, the first step of the framework is to extract local features. Then, a dictionary (codebook), which is a set of bases (codewords), is built to represent visual descriptors. Classical approaches are based on clustering techniques, such as K-means [100]. Feature encoding is meant to decompose local features over a codebook in order to obtain a representation of features in terms of the generated codewords. Given a descriptor, a coding technique activates a number of codewords and generates a coding vector, whose length is equal to the number of codewords. The next step is pooling the obtained codes to reach a compact signature for a specific sample. The max pooling technique is commonly used, leading to signatures that are appropriate to linear classifiers. Also, in image classification, the Spatial Pyramid Matching (SPM) step [93] is usually employed to include some spatial layout information in the final representation. Such final vectors of fixed size can be fed to a classifier, such as SVM.

Of all the above four stages, feature encoding is the core component [31, 129, 73], which greatly influences the recognition performance in terms of both accuracy and execution time. A large number of algorithms have been proposed in recent literature. The initial technique is vector quantization (VQ), also known as hard coding [145]. This encoding technique assigns a local feature to the closest visual word, which results in exactly one nonzero coefficient. A more robust voting scheme is the soft coding [158, 99], which assigns a descriptor to all (or a subset of) the codewords according to their distances.

Sparse coding (SC) for image classification, proposed by Yang et al. [175], is likely a milestone in this line of research. SC represents a local feature by a linear combination of a sparse set of basis vectors. Yu et al. [179] empirically observed that SC tends to be local, i.e. nonzero coefficients are often assigned to bases near the encoded data. But this locality cannot be ensured theoretically and they suggested a modification to SC, and proposed Local Coordinate Coding (LCC) [179], which explicitly encourages the coding to be local. They also demonstrated that the property of locality is more essential than sparsity, because the locality must lead to sparsity but not vice-versa. Based on LCC, Wang et al. proposed Locality-constrained Linear Coding (LLC) [166] and demonstrated its superiority compared to SC. Indeed, LLC may be viewed as a fast implementation of LCC that utilizes the locality constraint to project each descriptor into its local-coordinate system. Similarly, Liu et al. proposed Localized Soft Assignment (SA-k), where each descriptor is assigned to its $k-$nearest codewords [99]. In essence, LLC and SA-k share a common philosophy, since they both consider locality when mapping descriptors into codewords. In general, the incorporation of local structure in encoding is able to improve the stability and reduce the sensitivity to noise in descriptors [120].

These locality-based encoding methods use a fixed number of non-zero coefficients for each descriptor. However, the appropriate number of required codewords to well represent a specific descriptor in terms of the reconstruction error may vary depending on the descriptor. Fig.7.1 shows the number of dictionary codewords used to encode the skeleton-based descriptors of the first sample of MSRAction3D dataset. This sample has 54 frames, and a descriptor based on the pair-wise distances of joint points is obtained at each frame. A dictionary is generated by the standard dictionary learning algorithm of the SPAMS toolbox [102] and each descriptor is then encoded by regular Lasso. As the figure shows, the appropriate number of codewords is very different for each descriptor. This limitation of LLC and SA-k may affect the performance of these coding strategies in more complicated scenarios.

All of the encoding techniques discussed above, consider each descriptor in the sample as a separate encoding problem and encode each descriptor individually. Consequently, the final representation of a sample is generated by pooling or averaging among all descriptors of that sample. This strategy, however, does not take into account the fact that feature encoding is just an intermediate stage in creating a bag of features representation for the whole sample. Therefore, this approach is unsatisfactory due to the following reasons: 1) by treating each descriptor independently, the encoding phase completely ignores the

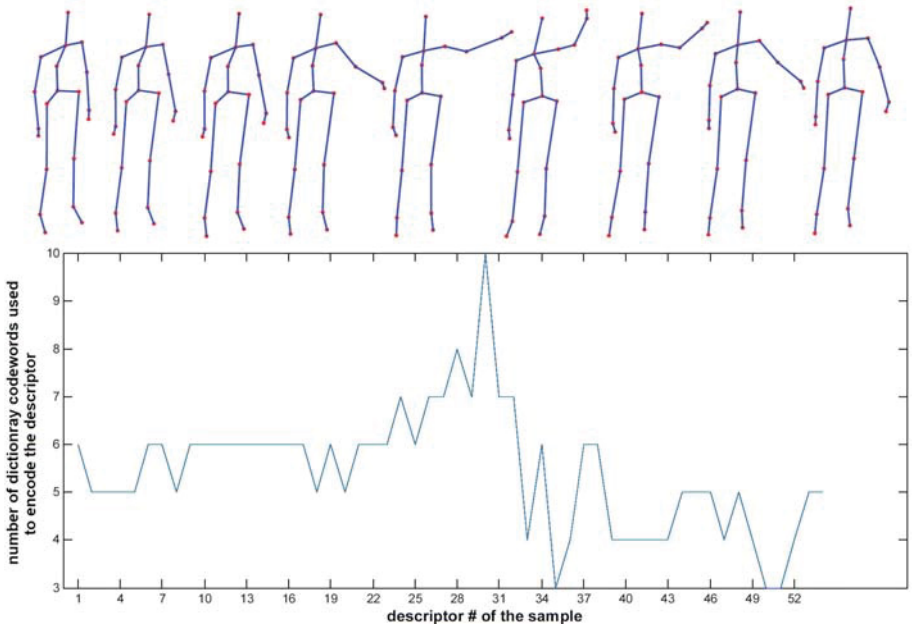


Figure 7.1: Number of codewords used to encode the skeleton-based descriptors of the first sample of MSRAction3D dataset [95]; the encoding is performed with regular Lasso [102].

commonality shared by these related descriptors; 2) sparse coding of each visual descriptor does not guarantee sparse coding of the whole sample [15].

In order to address this problem, the idea of group coding - appearing in the literature under various names such as group sparsity or grouped variable selection - has been introduced in [180, 15]. Group coding encourages the sparse coefficients in the same group to be zero or nonzero simultaneously [30].

In this chapter, we address the problem of action recognition by depth cameras using the BoVW framework and finally classification by classifier fusion. More specifically, we investigate the feature encoding stage and propose a new coding scheme, named *Locality Regularized Group Sparse Coding*, LGSC. The proposed method utilizes the advantages of locality coding such as its robustness to noise in descriptors, as well as the strengths of the group coding strategy by simultaneously taking into account the potential relation among descriptors of a sample. Indeed, the proposed method is an enhanced version of LLC; since it addresses its limitations, including setting a pre-defined number of coefficients for all descriptors as well as encoding descriptors of a sample individually. To efficiently implement our proposed method, we consider the Alternating Direction Method of Multipliers (ADMM) framework, which results in quadratic complexity in the problem size. In order to utilize the information of both depth and skeleton data, we employed two different feature

extraction methods: skeleton position and Depth Motion Maps (DMM). These two sets of features are fused at the classification level. The experimental results on three benchmark action recognition datasets prove the efficiency of the proposed method compared to the state-of-the-art techniques.

The rest of this chapter is organized as follows: Section 7.2 reviews works on related encoding techniques. Section 7.3 presents the proposed algorithm in detail. Section 7.4 reports the experimental results of the proposed method and compares with state-of-the-art. Finally, Section 7.5 concludes the chapter.

7.2 Related Work

In this section, we describe several encoding methods investigated in this chapter. Let $\mathbf{B} = [\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_M}] \in \mathbb{R}^{d \times M}$ be the given dictionary and $\mathbf{x} \in \mathbb{R}^d$ a local feature of a sample, where d is the dimensionality of a visual word (or a basis vector). Also, let $\mathbf{c} \in \mathbb{R}^M$ be the coding coefficient vector of $\mathbf{x}$, with c_i being the coefficient with respect to codeword $\mathbf{b_i}$. In the following, we review commonly used encoding schemes.

Hard-assignment coding (Vector Quantization): Each local descriptor is assigned to the nearest visual word. When Euclidean distance is used:

$$c_i = \begin{cases} 1, & \text{if } i = \arg\min_k \|\mathbf{x} - \mathbf{b_k}\|_2^2 \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

This representation is maximally sparse per descriptor since it picks a single codeword for each descriptor, but they may not be sparse for the sample as a whole. Furthermore, such representation does not consider codeword ambiguity and often introduces large quantization error.

Sparse Coding: Using sparse coding (SC) [175] as an alternative algorithm has significantly improved VQ robustness. In SC, the coding coefficient is obtained by solving the ℓ-norm regularized approximate problem:

$$\mathbf{c} = \arg\min_c \|\mathbf{x} - \mathbf{B}\mathbf{c}\|_2^2 + \lambda \|\mathbf{c}\|_1, \lambda \in \mathbb{R}, \tag{7.2}$$

where λ balances the sparsity of coefficients.

Locality-constrained Linear Coding: In LLC, coding is performed by solving the following optimization [166]:

$$\begin{aligned} &\mathbf{c} = \arg\min_{c} \|\mathbf{x} - \mathbf{B}\mathbf{c}\|_2^2 + \lambda \|\mathbf{d} \odot \mathbf{c}\|_2^2, \\ &s.t. \quad \mathbf{1}^T\mathbf{c} = 1, \end{aligned} \tag{7.3}$$

where $\mathbf{d}$ measures the Euclidean distance between $\mathbf{x}$ and each $\mathbf{b}$, and $\odot$ denotes element-wise multiplication. Also, an approximation is proposed to improve its computational efficiency. Ignoring the second term in Eq.7.3, it directly selects the k nearest codewords of $\mathbf{x}$ to minimize the first term by solving a much smaller linear problem. This gives the coefficient for the selected k codewords, and other coefficients are simply set to zero.

Group Sparse Coding: In this method [15], a set of descriptors $\mathbf{X} \in \mathbb{R}^{d\times N}$ are encoded jointly:

$$\arg\min_{\mathbf{C}} \frac{1}{2}\|\mathbf{X} - \mathbf{B}\mathbf{C}\|_F^2 + \lambda \sum_{i=1}^{M} \|\mathbf{C}_i\|_2, \tag{7.4}$$

where $\mathbf{C}_i$ is the i-th row of the coefficient matrix, $\mathbf{C} \in \mathbb{R}^{M\times N}$.

Fisher Vector coding: FV encoding assumes the generation process of local descriptors $\mathbf{X}$ can be modeled by a probability density function $P(.;\theta)$ with parameters θ. Then the sample can be described by the gradient vector of log likelihood with respect to the model parameters:

$$G_\theta^{\mathbf{X}} = \frac{1}{N}\nabla_\theta \log P(\mathbf{X};\theta). \tag{7.5}$$

The probability density function is usually modeled by Gaussian Mixture Model (GMM), and $\theta = \{\omega_i, \mu_i, \sigma_i; i = 1, \ldots, k\}$ are the model parameters, denoting the mixture weight, mean vector and variance matrix (assumed diagonal) of Gaussian i, respectively. Perronnin et al. [121] developed an improved fisher vector:

$$\mathcal{G}_{\mu,k}^{\mathbf{x}} = \frac{1}{\sqrt{\pi_k}}\gamma_k \left(\frac{\mathbf{x} - \mu_k}{\sigma_k}\right), \tag{7.6}$$

$$\mathcal{G}_{\sigma,k}^{\mathbf{x}} = \frac{1}{\sqrt{2\pi_k}}\gamma_k \left[(\frac{(\mathbf{x} - \mu_k)^2}{\sigma_k^2}) - 1\right], \tag{7.7}$$

where γ_k is the weight of local descriptor $\mathbf{x}$ to $k-$th Gaussian:

$$\gamma_k = \frac{\pi_k \mathcal{N}(\mathbf{x};\mu_k,\sigma_k)}{\Sigma_{i=1}^{K}\pi_i \mathcal{N}(\mathbf{x};\mu_i,\Sigma_k)}, \tag{7.8}$$

where $\mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)$ is $d-$dimensional Gaussian distribution. The final Fisher vector is obtained by the concatenation of these two gradients:

$$\text{FV}: \qquad \mathbf{C} = [\mathcal{G}^{\mathbf{x}}_{\mu,1}, \mathcal{G}^{\mathbf{x}}_{\sigma,1}, \ldots, \mathcal{G}^{\mathbf{x}}_{\mu,K}, \mathcal{G}^{\mathbf{x}}_{\sigma,K}]. \tag{7.9}$$

7.3 Locality Regularized Group Sparse Coding

In this section, we present the details of our proposed LGSC algorithm. Let $\mathbf{X}^g = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}] \in \mathbb{R}^{d\times N}$ denote a group of input data (e.x. visual descriptors), $\mathbf{C}^g \in \mathbb{R}^{M\times N}$ be its corresponding coefficient matrix, and $\mathbf{B} = [\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_M}] \in \mathbb{R}^{d\times M}$ be the given dictionary. Furthermore, we define a matrix $\mathbf{D} \in \mathbb{R}^{M\times N}$ that represents the pairwise distances between group instances and dictionary codewords, where d_{ij} denotes the Euclidean distance between $\mathbf{b_i}$ and $\mathbf{x_j}$.

Our main goal is to jointly encode groups of inputs, usually visual descriptors, in terms of a set of dictionary bases (codewords). To do that, we incorporate three terms in our objective function. The first term is the well-known reconstruction error, i.e. $\|\mathbf{X} - \mathbf{BC}\|$. Inspired by the local coding philosophy [179, 166, 99], the second term enforces locality by using a pairwise distance between the set of descriptors and the codewords in the given dictionary. In essence, distance captures the dissimilarity between descriptors and the codewords. Let $c_{ij} \in [0, 1]$ denote the contribution of $\mathbf{b_i}$ to $\mathbf{x_j}$. Therefore, the cost of encoding $\mathbf{x_j}$ with $\mathbf{b_i}$ is $d_{ij}c_{ij} \in [0, d_{ij}]$; and the cost of encoding x_j using all codewords is $\sum_{i=1}^{M} d_{ij}c_{ij}$. Thus, the total dissimilarity-based cost of encoding $\mathbf{X}^g$ via $\mathbf{B}$ is $\sum_{j=1}^{N}\sum_{i=1}^{M} d_{ij}c_{ij}$.

The third term regularizes the sparsity. Given $\mathbf{X}^g$ and $\mathbf{B}$, we aim to find as few codewords as possible that efficiently encode the group of descriptors simultaneously. When $\mathbf{b_i}$ is a representative of some of the instances of $\mathbf{X}^g$, we have $\mathbf{C_i} \neq 0$; i.e., the i-th row of $\mathbf{C}$ is nonzero. Having few codewords as representatives of inputs in $\mathbf{X}^g$ corresponds to having few nonzero rows in matrix $\mathbf{C}$.

Putting these three terms together, LGSC can be cast as an optimization problem that

minimizes the following objective function:

$$\begin{aligned}\mathcal{Q}_c(\mathbf{C},\mathbf{X},\mathbf{D}) &= \sum_g \mathcal{Q}_c(\mathbf{C}^g;\mathbf{X}^g,\mathbf{D}^g) \\ &= \sum_g \left(\frac{1}{2}\|\mathbf{X}^g - \mathbf{B}\mathbf{C}^g\|_F^2 + \lambda_1 \mathrm{tr}(\mathbf{D}^{g\top}\mathbf{C}^g) + \lambda_2 \sum_{i=1}^{M} I(\|\mathbf{C}_i^g\|)\right) \\ s.t. \quad & \mathbf{C}^g \geq 0,\end{aligned} \tag{7.10}$$

where $\mathrm{tr}(\cdot)$ denotes the trace operator, $I(\cdot)$ denotes the indicator function, which is zero when its argument is zero or one otherwise, $\|\cdot|_2$ denotes the ℓ_2-norm, $\mathbf{C}_i^g$ is the i'th row of $\mathbf{C}^g$, and λ_1 and λ_2 are parameters that balance the tradeoff between the reconstruction error, locality and sparsity.

For the sake of clarity, we present the optimization steps for one group of data $\mathbf{X}$ and its corresponding coefficients $\mathbf{C}$. Thus, our objective function becomes:

$$\begin{aligned}&\arg\min_{\mathbf{C}} \frac{1}{2}\|\mathbf{X} - \mathbf{B}\mathbf{C}\|_F^2 + \lambda_1 \mathrm{tr}(\mathbf{D}^T\mathbf{C}) + \lambda_2 \sum_{i=1}^{M} I(\|\mathbf{C}_i\|) \\ &s.t. \quad \mathbf{C} \geq 0.\end{aligned} \tag{7.11}$$

Since the problem in Eq.7.11 counting the number of nonzero rows of $\mathbf{C}$ is NP-hard, we employ the classical relaxation version by considering the $\ell_{1,2}$-norm on the matrix $\mathbf{C}$:

$$\begin{aligned}&\arg\min_{\mathbf{C}} \frac{1}{2}\|\mathbf{X} - \mathbf{B}\mathbf{C}\|_F^2 + \lambda_1 \mathrm{tr}(\mathbf{D}^T\mathbf{C}) + \lambda_2 \|\mathbf{C}\|_{1,2} \\ &s.t. \qquad \mathbf{C} \geq 0,\end{aligned} \tag{7.12}$$

where,

$$\|\mathbf{C}\|_{1,2} \triangleq \sum_{i=1}^{M} \|\mathbf{C}_i\|_2.$$

In this setting, the $\ell_{1,2}$-norm of $\mathbf{C}$ is the ℓ_1-norm of the ℓ_2-norm of the rows of $\mathbf{C}$, where, instead of counting the numbers of nonzero rows of $\mathbf{C}$, we use the sum of ℓ_2-norms of the rows of $\mathbf{C}$.

Unlike LLC, our formulation does not initially set the number of codewords that represent a given descriptor. Most importantly, LCC and LLC encode each descriptor in a sample individually. Consequently, the final representation vector may not be sparse for the sample as a whole.

We consider the implementation of our proposed optimization in Eq. 7.12 using the Alternating Direction Method of Multipliers (ADMM) framework [20]. The ADMM is a simple, but powerful, algorithm that is intended to blend the decomposability of dual ascent with the superior convergence properties of the method of multipliers. In ADMM form, Eq. 7.12 can be written as:

$$\begin{aligned} &\frac{1}{2}\|\mathbf{X}-\mathbf{BC}\|_F^2+\lambda_1\mathrm{tr}(\mathbf{D}^T\mathbf{C})+\lambda_2\|\mathbf{Z}\|_{1,2} \\ &s.t. \quad \mathbf{Z}\geq 0, \quad \mathbf{C}-\mathbf{Z}=0. \end{aligned} \tag{7.13}$$

As in the method of multipliers, augmenting the last equality constraint of Eq. 7.13 to the objective function via the Lagrange multiplier matrix $\Lambda \in R^{M\times N}$ and adding the penalty weighting parameter ρ, we form the augmented Lagrangian:

$$\begin{aligned} \mathcal{L}_\rho =&\frac{1}{2}\|\mathbf{X}-\mathbf{BC}\|_F^2+\lambda_1\mathrm{tr}(\mathbf{D}^T\mathbf{C})+\lambda_2\|\mathbf{Z}\|_{1,2} \\ &+\Lambda(\mathbf{C}-\mathbf{Z})+\frac{\rho}{2}\|\mathbf{C}-\mathbf{Z}\|_F^2. \end{aligned} \tag{7.14}$$

In the ADMM framework, we initialize all unknown variables and sequentially update these variables and the Lagrangian multipliers until some convergence criteria are met. In each update, we minimize the Lagrangian by only varying one variable of interest. This method allows for parallel implementation, which can reduce the computational time.

More specifically, the ADMM iterations consist of 1) minimizing $\mathcal{L}$ with respect to $\mathbf{C}$ while fixing other variables; 2) minimizing $\mathcal{L}$ with respect to $\mathbf{Z}$ while fixing other variables; and 3) updating the Lagrange multiplier matrix Λ, having other variables fixed. Algorithm 1 shows the steps of the ADMM implementation of the proposed algorithm.

The ADMM update at iteration $k+1$ can be derived as follows:

- **Updating C**: It can be shown that this reduces to solving a minimization of a quadratic function in terms of the matrix variable $\mathbf{C}$, which has the explicit solution:

$$\mathbf{C}^{k+1}=P^{-1}(\mathbf{B}^\top\mathbf{X}+\rho(\mathbf{Z}^k-\Lambda^k)-\lambda_1\mathbf{D}), \tag{7.18}$$

  where $P=\mathbf{B}^T\mathbf{B}+\rho I$ is a fixed matrix and can be inverted only once in advance. If the dimension of P is large, it is possible to reduce the computational cost by applying Cholesky factorization on P.

- **Updating Z**: Considering the non-negativity constraint on $\mathbf{Z}$, the $\mathbf{Z}$ update can be

Algorithm 1 : LGSC implementation via ADMM

Initialization: Set $\rho = 10^{-1}$; $\epsilon = 10^{-5}$, maxIter $= 10^3$.
Initialize $k = 0$; $\mathbf{C}^{(0)} = I$; $\mathbf{Z}^{(0)} = I$; $\Lambda^{(0)} = 0$, error1$= 2\epsilon$, error2$= 2\epsilon$, $\lambda_1 = 0.1$ and $\lambda_2 = 0.15$.

1: **while** (error1 $\geq \epsilon$ and error2 $\geq \epsilon$) and ($k <$ maxIter) **do**

2: Update $\mathbf{C}$ by

$$\begin{aligned}\mathbf{C}^{k+1} &= \arg\min_{\mathbf{C}} \mathcal{L}_\rho(\mathbf{C}, \mathbf{Z}^k, \Lambda^k) \\ &= \arg\min_{\mathbf{C}} \Big(\frac{1}{2}\|\mathbf{X} - \mathbf{B}\mathbf{C}\|_F^2 + \lambda_1 \mathrm{tr}(\mathbf{D}^T\mathbf{C}) + \\ &\frac{\rho}{2}\|\mathbf{C} - \mathbf{Z}^k + \Lambda^k\|_F^2\Big)\end{aligned} \tag{7.15}$$

3: Update $\mathbf{Z}$ by

$$\begin{aligned}\mathbf{Z}^{k+1} &= \arg\min_{\mathbf{Z}} \mathcal{L}_\rho(\mathbf{C}^{k+1}, \mathbf{Z}, \Lambda^k) \\ &= \arg\min_{\mathbf{Z}} (\lambda_2\|\mathbf{Z}\|_{1,2} + \frac{\rho}{2}\|\mathbf{C}^{k+1} - \mathbf{Z} + \Lambda^k\|_F^2) \\ &s.t. \quad \mathbf{Z} \geq 0\end{aligned} \tag{7.16}$$

4: Update the Lagrange multiplier matrix by

$$\Lambda^{k+1} = \Lambda^k + \rho(\mathbf{C}^{k+1} - \mathbf{Z}^{k+1}) \tag{7.17}$$

5: Calculate the objective function, Obj^k using Eq. 7.12

6: Calculate errors by

$$\begin{aligned}\text{error1} &= \text{Obj}^k - \text{Obj}^{k-1}, k = 2, \ldots \\ \text{error2} &= \|\mathbf{C}^{k+1} - \mathbf{Z}^{k+1}\|_\infty\end{aligned}$$

7: $k \leftarrow k + 1$

8: **end while**

Output: Optimal solution $\mathbf{C}^* = \mathbf{C}^k$

performed by:

$$\mathbf{Z}_i^{k+1} = [\mathcal{S}_{\lambda_2/\rho}(\mathbf{C}_i^{k+1} + \Lambda^k)]_+, \;\; i = 1, \ldots, M, \tag{7.19}$$

where the vector soft thresholding $\mathcal{S}$ is defined as $\mathcal{S}_\kappa(\mathbf{a}) = (1 - \kappa/\|\mathbf{a}\|_2)_+\mathbf{a}$.

- **Updating** Λ: Updating the Lagrange multiplier is straightforward. It is worth mentioning that, as in the method of multipliers, the dual variable update uses a step size equal to the augmented Lagrangian parameter.

7.4 Experiments

Here, we present the experimental details of evaluation, including the datasets used, feature extraction techniques employed for each dataset, the settings of the experiments, as well as the obtained results.

7.4.1 Datasets

We evaluated our framework on three public benchmark datasets: MSRAction3D [95], Cornel activity dataset (CAD-60) [148], and the Multi-modal Gesture Recognition Challenge 2013 (Chalearn) [40].

- **MSRAction3D dataset:** This dataset [95] is a well-known benchmark dataset for 3D action recognition. It contains 20 actions, including *high arm wave, horizontal arm wave, hammer, hand catch, forward punch, high throw, draw x, draw tick, draw circle, hand clap, two hand wave, side-boxing, bend, forward kick, side kick, jogging, tennis swing, tennis serve, golf swing, pick up & throw.* Each action was performed 2 or 3 times by each subject. Skeleton joint data of each frame is available having a variety of motions related to arms, legs, torso, and their combinations. In total, there are 567 depth map sequences with a resolution of 320 $\times$ 240. Some examples of the depth sequences are shown in Chapter 5 in Figure 5.5.

- **CAD-60:** The CAD dataset [148] contains 12 actions performed by 4 different subjects (two male and two female, one of them being left-handed) in 5 different environments: office, kitchen, bedroom, bathroom, and living room. The 12 activities are: *rinsing mouth, brushing teeth, wearing contact lens, talking on the phone, drinking water, opening pill container, cooking (chopping), cooking (stirring), talking on couch, relaxing on couch, writing on whiteboard, and working on computer.* All the

RGB, depth and skeleton data are provided in this dataset. Figure 6.3 in the previous chapter shows some example depth images from this dataset.

- **Chalearn dataset:** This dataset is a large video database of 13,858 gestures from a lexicon of 20 Italian gesture categories recorded with a Kinect camera, including audio, skeletal model, user mask, RGB and depth images [40]. It contains image sequences capturing 27 subjects performing natural communicative gestures and speaking in fluent Italian, and is divided into development, validation and test parts. Examples of RGB image sequences for some gestures are shown in Chapter 5 in Figure 5.4.

7.4.2 Feature extraction

In order to utilize the information of both depth and skeleton data, we employed two different feature extraction methods:

- **Skeleton position:** Given a depth image, 20 joints of the human body can be tracked by the skeleton tracker. At frame t, the position of each joint k is uniquely defined by three coordinates $P_k(t) = [x_i(t), y_i(t), z_i(t)]$. Instead of using the positions of joints, we employed the relative position of each joint to the torso at each frame.

- **Depth Motion Maps [27]:** In this simple method, depth motion maps (DMMs) generated from three projection views are used to capture the motion characteristics of an action sequence. Different from the original method [27], however, we split a sample along the time dimension into 10 segments and extracted DMM features from each segment. These features are considered sample descriptors. Finally, PCA is applied for redundancy and dimensionality reduction.

We intentionally chose these techniques as they are very efficient in terms of computation. Moreover, no parameter is involved in extracting local descriptors by these methods. Some well-known depth-based feature extraction method, such as DCSF [173], require many parameters adjustment and need fine-tuning. Our initial results showed that the classification performance strongly depends on the parameter settings. Therefore, changing the feature extraction parameters can significantly change the classification results. These two sets of features are finally fused at the classification level.

7.4.3 Temporal pyramid matching

Since the BoVW model usually does not encode the temporal information of local descriptors, we employed the simple, but efficient, Temporal Pyramid Matching (TPM) in order to

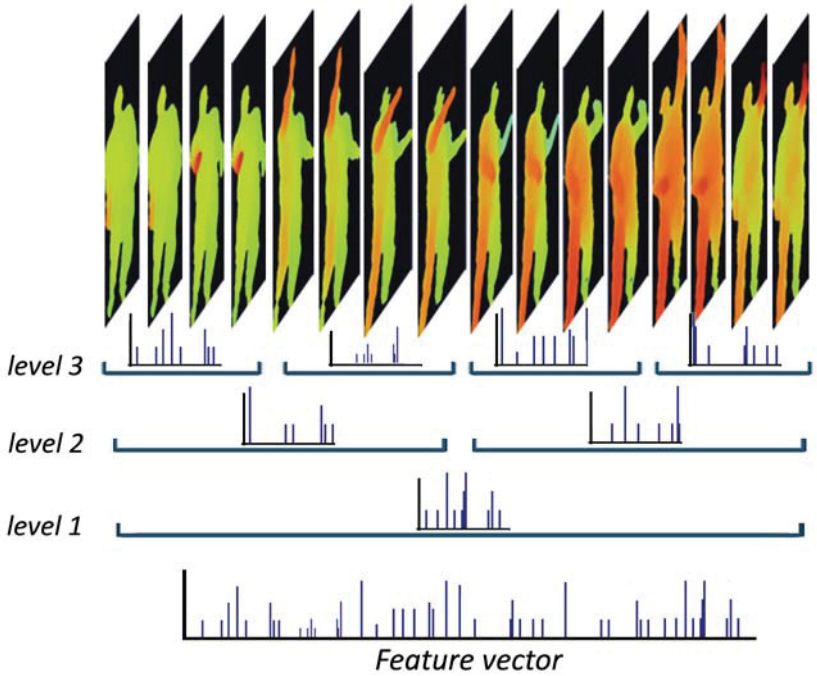


Figure 7.2: With TPM, the final feature vector of a sample is obtained by concatenating the encoded features at different levels.

incorporate the temporal information into the classification stage. TPM generally divides the video sequence into several segments along the time dimension. Feature representations obtained from segments are concatenated to form the representation. Standard TPM procedure, first, encodes all local descriptors from all frames and then applies TPM at different levels and segments. Here, we first divided the depth sequence into 3 levels with each containing 1, 2 and 4 segments, respectively. Then, we jointly encoded descriptors of each segment at each level using the LGSC method. The final representation of an action sample is obtained by concatenating the encoded features of all segments, as shown in Fig. 7.2.

7.4.4 Dictionary learning

Experimental results of [31, 129] show that, for the recognition task, codebook design is less critical than subsequent phases (i.e. encoding and pooling). The results of [166] also reveal that the codebook generated by K-Means can produce satisfactory results.

In this work, we build the dictionary with a similar method to [15] and also with the online dictionary learning algorithm of [102].

Table 7.1: Classification accuracy of different encoding methods on each dataset.

	VQ	SC	LLC	FV	GSC	LGSC
MSRAction3D	81.25	83.79	80.15	86.19	90.31	**93.45**
CAD-60	81.66	81.66	83.33	83.33	85.00	**88.30**
Chalearn	72.51	78.62	70.58	78.05	76.80	**79.64**

7.4.5 Parameter settings

For our method, we set λ_1 and λ_2 to 0.15 and 0.10, respectively. Also, the number of bases (codebook size) is experimentally set to 512. These values are chosen based on cross-validation on the training data. For Fisher Vector, we changed the codebook sizes from 16 to 256 and the best result is reported.

For the Chalearn dataset, for each group, we randomly used 100 sample for training and 50 samples for test. For the MSRAction3D dataset, many studies follow the experimental setting of Li et al. [95], such that they first divide the 20 actions into three subsets, each having 8 actions. For each subset, they perform three tests. In test one and two, 1/3 and 2/3 of the samples were used for training and the rest for testing. In the third test, half of the subjects are used as training and the rest of the subjects as testing. The experimental results on the first two tests are generally very promising, showing more than 90% accuracy. On the third test, however, the recognition performance dramatically decreases. It shows that many of these methods do not have good generalization performance when a different subject is performing the action, even in the same environmental settings. In order to have more reliable results, we followed the same experimental setup of [116]. In this setting, actors 1,3,5,7, and 9 are used for training and the rest are used for testing. For CAD-60 dataset, we followed the leave-one-subject-out settings as in [148].

7.4.6 Classification results

The summary of the results is reported in Table 7.1. In this table, we present the results of different encoding methods on each dataset.

In addition, the confusion matrices of the BoVW framework based on the proposed LGSC encoding algorithm for MSRAction3D and CAD-60 datasets are shown in Figure 7.3. As the figure shows, different types of actions have been classified with a satisfactory accuracy. The proposed method is robust against actions with large movements, such as *pickup and throw* and *two hands wave* in the MSRAction3D dataset. However, for actions in the same category with slight difference, such as *draw X* and *draw tick* or *draw circle* it shows less accuracy. For instance, the misclassified samples of *hand catch* class are classified in

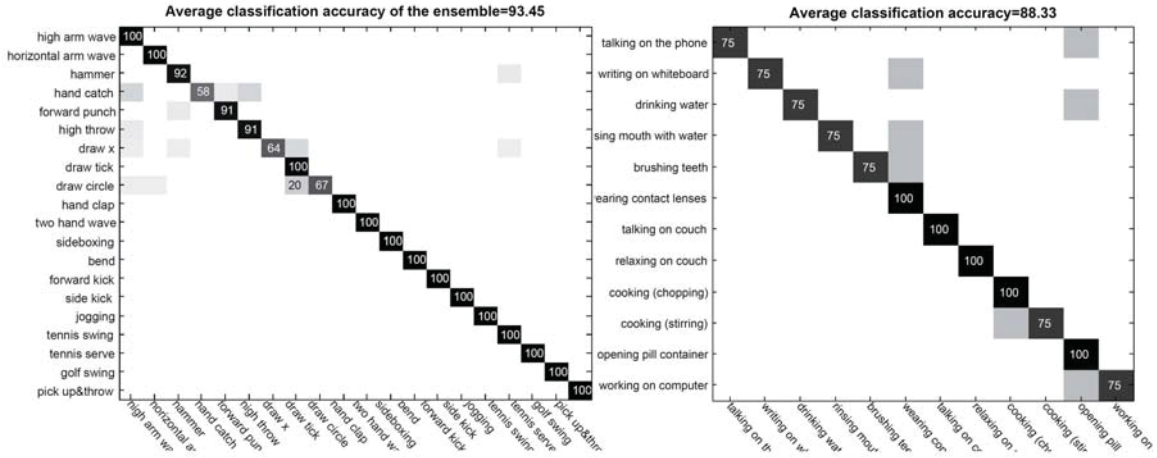


Figure 7.3: Confusion matrices of the classification system on the MSRAction3D dataset (left) and CAD-60 dataset (right).

similar classes, such as *forward punch*.

Table 7.2: Comparing the classification accuracy of our methodology with the state-of-the-art methods on MSRAction3D and CAD-60 datasets.

MSRAction3D	
	Accuracy
Studies employed depth data	
Action Graph [95]	74.70
HON4D [116]	85.85
Vieira et al. [161]	78.20
Random Occupancy Patterns [163]	86.50
DMM-HOG [178]	85.52
HOPC [125]	91.64
DMM-LBP-FF [26]	87.90
Studies employed only skeleton data	
Actionlet Ensemble [165]	88.20
Histogram of 3D Joint [174]	78.97
GB-RBM & HMM [110]	80.20
Points in a Lie Group [160]	89.48
Proposed LGSC Algorithm	**93.45**
CAD-60	
	Accuracy
Studies employed depth data	
Brun et al. [24]	86.50
MTO-Sparse coding [109]	65.30
Studies employed only skeleton data	
Actionlet Ensemble [165]	74.70
Sung et al. (2012) [148]	51.30
Proposed LGSC Algorithm	**88.33**

We then compare our classification results on MSRAction3D and CAD-60 datasets with state-of-the-art methods. Table 7.2 shows the accuracy of our method, as well as the rival

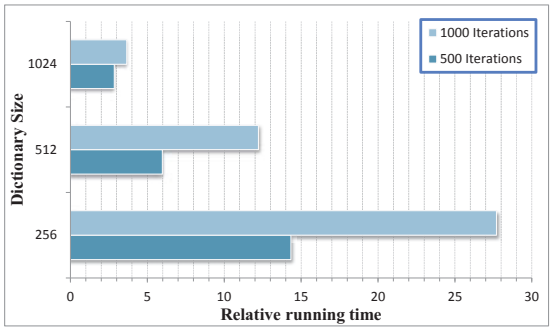


Figure 7.4: The relative running time of the implementation in [15] compared to our proposed ADMM framework.

methods on these datasets based on the standard settings.

The results provided in Table 7.1 and Table 7.2 along with the confusion matrices depicted in Figure 7.3 demonstrate the superiority of the proposed methodology. The results are quite promising, considering the facts that we used relatively simple depth and skeleton feature extraction methods without manipulating the depth data, which are quite noisy.

7.4.7 Comparison of the running time

By setting the weight of locality term to zero, our ADMM-based implementation can easily be employed for regular group sparse coding. Here, we further compare the execution time of group sparse coding using the ADMM algorithm with the algorithm proposed in [15]. To compare the execution time of these two methods, the simplest way is to compare the running time in the same conditions. In order to visually compare the results with a graph that is easy to interpret, the average of relative running times of both implementation methods to encode a group of descriptors of size 1000 over a dictionary of different sizes is presented in Fig. 7.4. In this figure, the relative running time of the method in [15] is shown with respect to our method, when the number of iterations of both methods is set to 500 and 1000. As an example, when the dictionary size is 512, our implementation is about 6 and 12 times faster with 500 and 1000 iterations, respectively. As the figure shows, the proposed implementation operates significantly faster than the one in [15], especially when the size of dictionary decreases.

7.5 Summary

The objective of feature encoding in BoVW models consists of representing features by decomposing them over an existing codebook. In this chapter, we proposed a novel encoding algorithm that jointly encodes a group of local features by considering a locality regularizer in the encoding phase, which improves the encoding stability and robustness to noises. Compared to the method in [15], the execution time of our method is significantly reduced by an efficient implementation using the ADMM framework. In order to utilize the information of both depth and skeleton data, we employed two different feature extraction methods: skeleton position and Depth Motion Maps (DMM). These two sets of features are finally fused at the classification level. The proposed method was evaluated on three benchmark action recognition datasets. Experimental results show significant performance improvements of the proposed method in comparison to state-of-the-art approaches on the considered datasets. Remarkably, we obtained 88.33% and 93.45% accuracy on the CAD-60 and MSRAction3D datasets, respectively.

Our plan for future work in this area consists of 1) utilizing the block structure strategy for dictionary learning and feature encoding phases; 2) incorporating the temporal information into the encoding optimization problem, such that the encoded feature vector includes the temporal behavior of the sample; and 3) evaluating the proposed methodology on a wider range of applications, especially for image categorization.

Chapter 8

Conclusion and Future Work

This dissertation presents a considerable step towards enhanced recognition of visual data by ensemble learning. We believe that with the ever increasing amount of digital contents generated by consumers and institutions, this direction of research becomes more important in both algorithmic development and real world application. In this chapter, we summarize the contributions and key observations of our work, and discuss future research directions that go beyond our current achievement.

8.1 Summary

The objective of this dissertation is to explore efficient algorithms for visual data recognition based on the ensemble learning. We first investigate ensemble classification, with special interest in subclass ensemble and Error Correcting Output Codes. The ensemble approach is then utilized for a visual classification application: action recognition by depth data.

We first reviewed the existing ensemble classification methods and presented a unifying framework for multiple classifier systems that conceptually unites a large variety of ensemble classification methods, including existing class binarization techniques such as ECOC. In the framework, various ensemble methods are broadly categorized into four general approaches. Among them, three approaches generate ensemble classifiers by manipulating data, which we have named subsample, subspace, and subclass approaches. The fourth approach, learner manipulation, is usually based on using different learning algorithms or variations of the parameters of base learners. The proposed framework is unique in a sense that it links two research lines in machine learning: multiclass classification based on the class binarization techniques and the strategies of ensemble classification. According to this proposed framework, we provided a brief survey of ensemble creation methods as well as the principal techniques proposed to combine them.

Inspired by the proposed framework, we presented a new general approach to ensemble classification, named Generic Subclass Ensemble. The proposed approach differs from existing methods that manipulate the target attribute, since in our approach individual classification problems are not restricted to two-class problems. In light of this approach,

class binarization techniques are considered special cases of the generic subclass ensemble approach. Based on the generic subclass approach, three methods are introduced: Subclass.Equidistant, Subclass.ClsPart_MI, and Subclass.ClsPart_Dist. Using the neural network as the base learner, we evaluated the efficiency of the generic subclass ensemble on a set of benchmark datasets. Experimental results show that the subclass approach presents a viable alternative to the most commonly used ensemble classification approaches. Specifically, this approach shows a better performance in problems with a larger number of classes. In these cases, instead of combining individual classifiers trained with different subsets of samples or features, the more efficient approach is to train classifiers on a subset of classes.

Then, we focused on the subclass approach, specifically on Error Correcting Output Codes (ECOC). We first proposed a subspace approach to the ECOC framework by defining a three dimensional ECOC matrix, where the third dimension corresponds to the feature space. In this sense, different subsets of features can be activated for a given dichotomy. Also, a Genetic Algorithm-based technique is employed for the optimal design of an application-dependent subspace ECOC. The proposed method takes advantage of some basic concepts of ensemble classification, such as diversity of classifiers. By taking into account the problem domain, this method also benefits from the evolutionary approach for optimizing the three-dimensional codematrix. As a result, we obtain a problem-dependent coding design with more independent classifiers, which reduces the bias and variance errors of the multiclass problem and, consequently, increases the discrimination power of the ensemble. The method was evaluated on several UCI datasets as well as two shape categorization problems using two different base learners. The experimental results showed significant performance improvement of the proposed method compared to state-of-the-art approaches.

The ensemble model is then utilized for a visual classification application. More specifically, the recent release of the Microsoft Kinect camera inspired us to employ the ensemble-based models for classification of actions using depth cameras. We presented three new methods that improve the recognition of depth-based action videos.

First, we generated an ensemble classification framework to address two action recognition problems. We designed a model consists of a set of classifiers, each one trained over different feature sets. The individual classifier outputs are then efficiently combined by means of the Dempster-Shafer fusion method, taking benefit from diversity of base classifiers trained on different sources of information. We compared the classification results of

the individual classifiers with those obtained from fusing the classifiers by the Dempster-Shafer combination method on two public datasets, showing performance improvements of the proposed methodology. We showed performance improvements in relation to the state of the art results on the considered datasets. We also introduce two fast action representation techniques, using only the skeleton joint' position during the time. The advantages of our methods are that: 1) They will generate a fixed size feature vector for an action, that may vary in time based on the action and subject that performs the action. Thus, they can be used as an input for any type of classifier. 2) The proposed representation techniques are relatively very fast. Thus, they are computationally much more efficient for real time applications.

Second, an efficient method that is applicable to trajectory classification, such as action recognition, is developed by incorporating two efficient time-series distances measures. More specifically, we utilized Dynamic Time Warping and Longest Common subsequences, as well as their derivatives. However, instead of employing these general measures as a distance measure for k-NN classifier, we transformed these measures using the pairwise proximity function in order to make use of non-positive semi-definite kernels in the SVM formulation. We build an ensemble of four ppfSVMs using the computed kernels and combine them using the product fusion technique. Comparing the recognition results of the proposed methods with state-of-the-art techniques on three action recognition datasets showed significant performance improvements.

Finally, a new encoding algorithm was proposed that jointly encodes a group of local features by considering a locality regularizer in the encoding phase. In order to utilize the information of both depth and skeleton data, we employed two different feature extraction methods: skeleton position and Depth Motion Maps (DMM). These two sets of features are finally fused at the classification level. The proposed method was evaluated on three benchmark action recognition datasets with skeleton and depth feature sets. Results show significant performance improvements of the proposed method in comparison to state-of-the-art approaches on the considered datasets. Remarkably, we obtained 88.33% and 93% accuracy on the CAD-60 and MSRAction3D datasets, respectively.

8.2 Future Directions

For future research, we aim to investigate three initial directions, described in the following sections.

8.2.1 Investigating deep features

Since the success of deep neural networks on achieving impressive image classification accuracy on the *ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012* [87], many approaches have tried to incorporate deep networks to tackle the problems of image and video recognition. Our main future research relies on exploring deep neural network. More specifically, we aim to explore the performance of an ensemble of Convolutional Neural Networks, trained on different sources of data.

8.2.2 Exploiting the RGB-D data

The proposed action representation techniques as well as the ensemble classification framework is based on using the depth data. The recognition performance when both RGB and depth features are available needs to be evaluated and fusion schemes should be explored. In future, we aim to employ other sources of information, specially the RGB data. Tracking of skeleton joints' position and depth maps generally has discriminative information. However, they may fail in cases that two classes are closely similar in terms of positions of skeleton joints during the time. In addition, in some applications, depth data and/or skeleton position may not be available. Our hypothesis is that employing RGB-D data can significantly improve the recognition performance.

8.2.3 Continuous action recognition

The proposed methods in this dissertations are mostly appropriate for single-actions. In the future, we would like to investigate more complicated real-world scenarios. More specifically, we are interested in developing new approaches for continuous action recognition, i.e. detecting a series of actions performed by a single subject and/or multiple subjects.

Bibliography

[1] D. Aha and R. Bankert. Cloud classification using error-correcting output codes. *Artificial Intelligence Applications: Natural Resources, Agriculture, and Environmental Science*, 11(1):13–28, 1997.

[2] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001.

[3] E. Alpaydin and E. Mayoraz. Learning error-correcting output codes from data. In *International conference on artificial neural networks (ICANN99)*, volume 2, page 43748, 1999.

[4] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka. Efficient classification for multiclass problems using modular neural networks. *IEEE Transactions on Neural Networks*, 6(1):117–124, 1995.

[5] M.A. Bagheri, Q. Gao, and S. Escalera. Efficient pairwise classification using local cross off strategy. In *25th Canadian conf. on Artificial Intelligence*, volume 7310, pages 25–36, Toronto, Canada, 2012.

[6] M.A. Bagheri, Q. Gao, and S. Escalera. Logo recognition based on the dempster-shafer fusion of multiple classifiers. In *26th Canadian conf. on Artificial Intelligence*, Regina, Canada, 2013.

[7] M.A. Bagheri, Qigang Gao, and Sergio Escalera. Rough set subspace error-correcting output codes. In *IEEE International Conference on Data Mining (ICDM)*, pages 822– 827, Belgium, Brussels, 2012.

[8] Mohammad Ali Bagheri, Qigang Gao, and Sergio Escalera. A framework towards the unification of ensemble classification methods. In *12th International Conference on Machine Learning and Applications (ICMLA)*, volume 2, pages 351–355, 2013.

[9] Mohammad Ali Bagheri, Qigang Gao, and Sergio Escalera. A genetic-based subspace analysis method for improving error-correcting output coding. *Pattern Recognition*, 46(10):2830–2839, 2013.

[10] Mohammad Ali Bagheri, Gang Hu, Qigang Gao, and Sergio Escalera. A framework of multi-classifier fusion for human action recognition. In *22nd International Conference on Pattern Recognition (ICPR)*, pages 1260–1265, 2014.

[11] Mohammad Ali Bagheri and Gholam Ali Montazer. Ensemble classifier strategy based on transient feature fusion in electronic nose. In *14th international symposium on olfaction and electronic nose*, New York City, NY, (USA), 2011.

[12] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):173–180, 2007.

[13] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1):105–139, 1999.

[14] Miguel ngel Bautista, Sergio Escalera, Xavier Bar, Petia Radeva, Jordi Vitri, and Oriol Pujol. Minimal design of error-correcting output codes. *Pattern Recognition Letters*, 33(6):693–702, 2012.

[15] Samy Bengio, Fernando Pereira, Yoram Singer, and Dennis Strelow. Group sparse coding. In *NIPS*, pages 82–89, 2009.

[16] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, department of information and computer sciences, university of california, irvine, 1998.

[17] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):257–267, 2001.

[18] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[19] R.C. Bose and D.K. Ray-Chaudhuri. On a class of error-correcting binary group codes. *Information and Control*, 3:68–79, 1960.

[20] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[21] L. Breiman. Pasting small votes for classifcation in large databases and on-line. *Machine Learning*, 36(36):85–103, 1999.

[22] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[23] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[24] Luc Brun, Gennaro Percannella, Alessia Saggese, and Mario Vento. Action recognition by using kernels on aclets sequences. *Computer Vision and Image Understanding*, 2015.

[25] Robert Bryll, Ricardo Gutierrez-Osuna, and Francis Quek. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6):1291–1302, 2003.

[26] Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. Action recognition from depth sequences using depth motion maps-based local binary patterns. In *WACV*, pages 1092–1099, 2015.

[27] Chen Chen, Kui Liu, and Nasser Kehtarnavaz. Real-time human action recognition based on depth motion maps. *Journal of Real-Time Image Processing*, pages 1–9, 2013.

[28] Jingying Chen, Maylor K. Leung, and Yongsheng Gao. Noisy logo recognition using line segment hausdorff distance. *Pattern Recognition*, 36(4):943–955, 2003.

[29] Shih-Chieh Chen, Shih-Wei Lin, and Shuo-Yan Chou. Enhancing the classification accuracy by scatter-search-based ensemble approach. *Applied Soft Computing*, 11(1):1021–1028, 2011.

[30] Yu-Tseh Chi, Mohamed Ali, Ajit Rajwade, and Jason Ho. Block and group regularized sparse modeling for dictionary learning. In *CVPR*, pages 377–382, 2013.

[31] Adam Coates and Andrew Y Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, pages 921–928, 2011.

[32] Andr LV Coelho and Diego SC Nascimento. On the evolutionary design of heterogeneous bagging models. *Neurocomputing*, 73(16):3319–3322, 2010.

[33] Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2-3):201–233, 2002.

[34] Amit David and Boaz Lerner. Support vector machine-based image classification for genetic syndrome diagnosis. *Pattern Recognition Letters*, 26(8):1029–1038, 2005.

[35] A.P. Dempster. Upper and lower probabilities induced by multivalued mappings. *Annals of Mathematical Statistics*, 38(2):325339, 1967.

[36] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[37] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[38] D. Doermann, E. Rivlin, and I. Weiss. Applying algebraic and differential invariants for logo recognition. *Machine Vision and Applications*, 9(2):73–86, 1996.

[39] Piotr Dollar, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72. IEEE, 2005.

[40] S. Escalera, J. Gonzlez, X. Bar, M. Reyes, O. Lopes, I. Guyon, V. Athistos, and H.J. Escalante. Multi-modal gesture recognition challenge 2013: Dataset and results. In *ICMI*, 2013.

[41] S. Escalera, O. Pujol, and P. Radeva. ECOC-ONE: A novel coding and decoding strategy. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 578–581, 2006.

[42] S. Escalera, O. Pujol, and P. Radeva. On the decoding process in ternary error-correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):120–134, 2010.

[43] Sergio Escalera, Alicia Forns, Oriol Pujol, Petia Radeva, Gemma Snchez, and Josep Llads. Blurred shape model for binary and grey-level symbol recognition. *Pattern Recognition Letters*, 30(15):1424–1433, 2009.

[44] Sergio Escalera, Oriol Pujol, and Petia Radeva. Separability of ternary codes for sparse designs of error-correcting output codes. *Pattern Recognition Letters*, 30(3):285–297, 2009.

[45] Sergio Escalera, David Tax, Oriol Pujol, Petia Radeva, and Robert Duin. Subclass problem-dependent design of error-correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1041–1054, 2009.

[46] L.J. Eshelman. *The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination*. Morgan Kauffman, San Mateo, CA, 1990.

[47] Daniel Fagella. *The Rise of Neural Networks and Deep Learning in Our Everyday Lives A Conversation with Yoshua Bengio*, 2016 (accessed August 10, 2016).

[48] A. Ferreira. Survey on boosting algorithms for supervised and semi-supervised learning. Technical report, Technical Report, Instituto Superior de Engenharia de Lisboa, 2007.

[49] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[50] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The annals of statistics*, 28(2):337–374, 2000.

[51] Nicols Garca-Pedrajas. Constructing ensembles of classifiers by means of weighted instance selection. *Neural Networks, IEEE Transactions on*, 20(2):258–277, 2009.

[52] Nicols Garca-Pedrajas, Csar Garca-Osorio, and Colin Fyfe. Nonlinear boosting projections for ensemble construction. *J. Mach. Learn. Res.*, 8:1–33, 2007.

[53] Nicols Garca-Pedrajas and Domingo Ortiz-Boyer. Boosting random subspace method. *Neural Network*, 21(9):1344–1362, 2008.

[54] Nicols Garca-Pedrajas and Domingo Ortiz-Boyer. An empirical study of binary classifier fusion methods for multiclass classification. *Information Fusion*, 12(2):111–130, 2011.

[55] N. Garcia-Pedrajas and C. Fyfe. Evolving output codes for multiclass problems,. *IEEE Transactions of Evolutionary Computation*, 12(1):93–106, 2007.

[56] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Comput*, 4:1–58, 1995.

[57] Ross Girshick, Jamie Shotton, Pushmeet Kohli, Antonio Criminisi, and Andrew Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *ICCV*, pages 415–422, 2011.

[58] Simon Gnter and Horst Bunke. Feature selection algorithms for the generation of multiple classifier systems and their application to handwritten word recognition. *Pattern Recognition Letters*, 25(11):1323–1336, 2004.

[59] Tomasz Górecki. Using derivatives in a longest common subsequence dissimilarity measure for time series classification. *Pattern Recognition Letters*, 45:99–105, 2014.

[60] Thore Graepel, Ralf Herbrich, Peter Bollmann-Sdorra, and Klaus Obermayer. Classification on pairwise proximity data. *NIPS*, pages 438–444, 1999.

[61] Venkatesan Guruswami and Amit Sahai. Multiclass learning, boosting, and error-correcting codes, 1999.

[62] He Haibo and Cao Yuan. SSC: A classifier combination method based on signal strength. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1100–1117, 2012.

[63] J. Hampshire and A. Waibel. A novel objective function for improved phoneme recognition using time-delay neural networks. *IEEE Transactions on Neural Networks*, 1(2):216–228, 1990.

[64] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics*, 2013.

[65] Lei Han, Xinxiao Wu, Wei Liang, Guangming Hou, and Yunde Jia. Discriminative human action recognition in the learned hierarchical manifold space. *Image and Vision Computing*, 28(5):836–849, 2010.

[66] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471, 1998.

[67] Nima Hatami. Thinned-ECOC ensemble based on sequential code shrinking. *Expert Systems with Applications*, 39(1):936–947, 2012.

[68] Antonio Hernndez-Vela, Miguel Angel Bautista, Xavier Perez-Sala, Victor Ponce, Xavier Bar, Oriol Pujol, Cecilio Angulo, and S. Escalera. Bovdw: Bag-of-visual-and-depth-words for gesture recognition. In *ICPR*, pages 449–452. IEEE, 2012.

[69] Tin. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattem Analysis and Machine Intelligence*, 20:832–844, 1998.

[70] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[71] M.K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[72] Qinghua Hu, DarenYu, Zongxia Xie, and Xiaodong Li. EROS: Ensemble rough subspaces. *Pattern Recognition*, 40:3728 – 3739, 2007.

[73] Yongzhen Huang, Zifeng Wu, Liang Wang, and Tieniu Tan. Feature coding in image classification: A comprehensive study. *PAMI*, 36(3):493–506, 2014.

[74] R.L. Iman and J.M. Davenport. Approximations of the critical regions of the friedman statistic. *Communications in Statistics*, 6:571–595, 1980.

[75] M. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neuralnetwork ensembles. *IEEE Transactions on Neural Networks*, 14(4):820–834, 2003.

[76] A.K. Jain and A. Vailaya. Shape-based retrieval: A case study with trademark image databases. *Pattern Recognition*, 31(9):1369–1390, 1998.

[77] Hui Jiang, Chong-Wah Ngo, and Hung-Khoon Tan. Gestalt-based feature similarity measure in trademark database. *Pattern Recognition*, 39(5):988–1001, 2006.

[78] YG Jiang, J Liu, A Roshan Zamir, G Toderici, I Laptev, M Shah, and R Sukthankar. Thumos challenge: Action recognition with a large number of classes. *http://crcv.ucf.edu/THUMOS14*, 2014.

[79] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & psychophysics*, 14(2):201–211, 1973.

[80] Eamonn J Keogh and Michael J Pazzani. Derivative dynamic time warping. In *SDM*, volume 1, pages 5–7, 2001.

[81] Microsft kinect [online]. http://www.xbox.com/kinect.

[82] J. Kittler, R. Ghaderi, T. Windeatt, and J. Matas. Face verification via error correcting output codes. *Image and Vision Computing*, 21(13-14):1163–1169, 2003.

[83] Alexander Klaser and Marcin Marszalek. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.

[84] J. F. Kolen and J. B. Pollack. Back propagation is sesitive to initial conditions. In Morgan Kaufmann, editor, *Advances in Neural Information Processing Systems*, volume 3, pages 860–867, San Francisco, CA., 1991.

[85] E.B. Kong and T.G. Dietterich. Error-correcting output coding corrects bias and variance. In A. Prieditis and J.F. Lemmer, editors, *Machine Learning: Proceedings of the Twelfth International Conference on Machine Learning*, pages 313–321. 1995.

[86] E.B. Kong and T.G. Dietterich. Why error-correcting output coding works with decision trees. Technical report, Technical Report, Department of Computer Science, Oregon State University, Corvallis, OR., 1995.

[87] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[88] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, New York, NY, 2004.

[89] L. I. Kuncheva and L. C. Jain. Designing classifier fusion systems by genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(4):327–336, 2000.

[90] Ludmila I. Kuncheva. Using diversity measures for generating error-correcting output codes in classifier ensembles. *Pattern Recognition Letters*, 26:83–90, 2005.

[91] Ivan Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005.

[92] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[93] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178, 2006.

[94] S. Z. Li and Zhang Zhenqiu. Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1112–1123, 2004.

[95] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *CVPR Workshop*, pages 9–14, 2010.

[96] Winston Li, Henry Leung, Chiman Kwan, and Bruce R. Linnell. E-nose vapor identification based on dempster-shafer fusion of multiple classifiers. *IEEE Transactions on Instrumentation and Measurement*, 57(10):2273–2282, 2008.

[97] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3337–3344. IEEE, 2011.

[98] Li Liu and Ling Shao. Learning discriminative representations from rgb-d video data. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2013.

[99] Lingqiao Liu, Lei Wang, and Xinwang Liu. In defense of soft-assignment coding. In *ICCV*, pages 2486–2493, 2011.

[100] Stuart P Lloyd. Least squares quantization in pcm. *IEEE Trans. on Information Theory*, 28(2):129–137, 1982.

[101] Fengjun Lv and Ramakant Nevatia. Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. In *ECCV*, pages 359–372. Springer, 2006.

[102] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *JMLR*, 11:19–60, 2010.

[103] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic, San Diego, CA, 2nd edition, 1999.

[104] Olvi L Mangasarian. Generalized support vector machines. *NIPS*, pages 135–146, 1999.

[105] James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *ICML*, pages 1033–1040, 2011.

[106] Prem Melville and Raymond J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99–111, 2005.

[107] S. H. Nabavi-Kerizi, M. Abadi, and E. Kabir. A pso-based weighting method for linear combination of neural networks. *Computers & Electrical Engineering*, 36(5):886–894, 2010.

[108] P.B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, 1963.

[109] Bingbing Ni, Pierre Moulin, and Shuicheng Yan. Order-preserving sparse coding for sequence classification. In *ECCV*, pages 173–187. Springer, 2012.

[110] Siqi Nie and Qiang Ji. Capturing global and local dynamics for human action recognition. In *ICPR*, pages 1946–1951, 2014.

[111] University of Maryland. Laboratory for language and media processing (lamp) , logo dataset, 2012.

[112] Eshed Ohn-Bar and Mohan Manubhai Trivedi. Joint angles similarities and hog2 for action recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pages 465–470. IEEE, 2013.

[113] D. Opitz and R. Maclin. Popular ensemble methods: an empirical study. *J. Artif. Res.*, 11:169–198, 1999.

[114] D Opitz and J Shavlik. Generating accurate and diverse members of a neural-network ensemble. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 535–541. The MIT Press, 1996.

[115] David W. Opitz. Feature selection for ensembles, 1999.

[116] Omar Oreifej, Zicheng Liu, and WA Redmond. HON4D: Histogram of oriented 4D normals for activity recognition from depth sequences. In *CVPR*, pages 716–723, 2013.

[117] M. Pardo, G. Sberveglieri, A. Taroni, F. Masulli, and G. Valentini. Decompositive classification models for electronic noses. *Analytica Chimica Acta*, 446(12):221–230, 2001.

[118] B. Parmanto, P.W. Munro, and H.R. Doyle. Improving committee diagnosis with resampling techniques. In Mozer M.C. Hesselmo M.E. Touretzky, D.S., editor, *Advances in Neural Information Processing Systems*, volume 8, pages 882–888. MIT Press, Cambridge, MA, 1996.

[119] A. Passerini, M. Pontil, and P. Frasconi. New results on error correcting output codes of kernel machines. *IEEE Transactions on Neural Networks*, 15(1):45–54, 2004.

[120] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *arXiv preprint arXiv:1405.4506*, 2014.

[121] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, pages 143–156. 2010.

[122] W.W. Peterson and J.R.Weldon. *Error-Correcting Codes*. MIT Press, Cambridge, MA, 1972.

[123] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[124] O. Pujol, P. Radeva, and J. Vitria. Discriminant ECOC: a heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1007–1012, 2006.

[125] Hossein Rahmani, Arif Mahmood, Du Q Huynh, and Ajmal Mian. Hopc: Histogram of oriented principal components of 3d pointclouds for action recognition. In *ECCV*, pages 742–757. 2014.

[126] Konstantinos Rapantzikos, Yannis Avrithis, and Stefanos Kollias. Dense saliency-based spatiotemporal feature points for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1454–1461, 2009.

[127] M. Re and G. Valentini. Ensemble methods: a review. In *Advances in Machine Learning and Data Mining for Astronomy*, pages 563–594. Chapman & Hall, 2012.

[128] Miguel Reyes, Gabriel Dominguez, and S. Escalera. Feature weighting in dynamic timewarping for gesture recognition in depth data. In *CVPRW*, pages 1182–1188, 2011.

[129] Roberto Rigamonti, Matthew Brown, Vincent Lepetit, et al. Are sparse representations really relevant for image classification? In *CVPR*, pages 1545–1552, 2011.

[130] A. Rocha and S. K. Goldenstein. Multiclass from binary: Expanding one-versus-all, one-versus-one and ECOC-based approaches. *IEEE Transactions on Neural Networks and Learning Systems*, 2013.

[131] Juan J. Rodrguez and Jess Maudes. Boosting recombined weak classifiers. *Pattern Recognition Letters*, 29(8):1049–1059, 2008.

[132] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[133] G. Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7:777781, 1994.

[134] Lior Rokach. Genetic algorithm-based feature set partitioning for classification problems. *Pattern Recognition*, 41(5):1676 – 1700, 2008.

[135] Lior Rokach. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis*, 53(12):4046–4072, 2009.

[136] Lior Rokach. *Pattern classification using ensemble methods*. Series in machine perception and artificial intelligence. World Scientific, Hackensack, N.J.; London, 2010.

[137] Maimon O. Arad O. Rokach, L. Improving supervised learning by sample decomposition. *International Journal of Computational Intelligence and Applications*, 5(1):37–54, 2005.

[138] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, 1978.

[139] L. E. Santana, A. M. P. Canuto, and L. Silva. Bio-inspired meta-heuristic as feature selector in ensemble systems: A comparative analysis. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1112–1119, 2011.

[140] R.E. Schapire. Using output codes to boost multiclass learning problems. In *Proceedings of the 14th International Conference on Machine Learning*, pages 313–321, San Francisco, CA, Morgan Kauffman, Los Altos, CA,, 1997.

[141] RobertE Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[142] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[143] Wei Shen, Ke Deng, Xiang Bai, Tommer Leyvand, Baining Guo, and Zhuowen Tu. Exemplar-based human action pose correction and tagging. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1784–1791. IEEE, 2012.

[144] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[145] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, 2003.

[146] Dan Sun and Daoqiang Zhang. Bagging constraint score for feature selection with pairwise constraints. *Pattern Recognition*, 43(6):2106–2118, 2010.

[147] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Unstructured human activity detection from rgbd images. In *ICRA*, pages 842–849. IEEE, 2012.

[148] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Unstructured human activity detection from RGBD images. In *ICRA*, pages 842–849, 2012.

[149] J. Tanha, M. van Someren, and H. Afsarmanesh. An adaboost algorithm for multiclass semi-supervised learning. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1116–1121, 2012.

[150] Dacheng Tao, Xiaoou Tang, Xuelong Li, and Xindong Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1088–1099, 2006.

[151] K. Torkkola. Feature extraction by non-parametric mutual information maximization. *J. Machine Learning Research*, 3:1415–1438, 2003.

[152] Anne Treisman and Hilary Schmidt. Illusory conjunctions in the perception of objects. *Cognitive psychology*, 14(1):107–141, 1982.

[153] Alexey Tsymbal, Seppo Puuronen, and David W. Patterson. Ensemble feature selection with the simple bayesian classification. *Information Fusion*, 4(2):87–100, 2003.

[154] K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. Sharkey, editor, *Combining Artificial Neural Nets*, pages 127–162. Springer, Berlin, 1999.

[155] K. Tumer and N.C. Oza. Input decimated ensembles. *Pattern Anal. Appl.*, 6:65–77, 2003.

[156] Elif Deryas Ubeyli. Multiclass support vector machines for diagnosis of erythemato-squamous diseases. *Expert Systems with Applications*, 35(4):1733–1740, 2008.

[157] W. Utschick and W. Weichselberger. Stochastic organization of output codes in multiclass learning problems. *Neural Computation*, 13(5):1065–1102, 2001.

[158] Jan C Van Gemert, Cor J Veenman, Arnold WM Smeulders, and Jan-Mark Geusebroek. Visual word ambiguity. *PAMI*, 32(7):1271–1283, 2010.

[159] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[160] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *CVPR*, pages 588–595, 2014.

[161] Antonio W Vieira, Erickson R Nascimento, Gabriel L Oliveira, Zicheng Liu, and Mario FM Campos. Stop: Space-time occupancy patterns for 3d action recognition from depth map sequences. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 252–259. Springer, 2012.

[162] Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3169–3176. IEEE, 2011.

[163] Jiang Wang, Zicheng Liu, Jan Chorowski, Zhuoyuan Chen, and Ying Wu. Robust 3D action recognition with random occupancy patterns. In *ECCV*, pages 872–885. Springer, 2012.

[164] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1290–1297. IEEE, 2012.

[165] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Learning actionlet ensemble for 3d human action recognition. *PAMI*, 36(5):914–927, 2014.

[166] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, 2010.

[167] C.H. Wei, Y. Li, W.Y. Chau, and C.T. Li. Trademark image retrieval using synthetic features for describing global shape and interior structure. *Pattern Recognition*, 42(3):386–394, 2009.

[168] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, pages 650–663. Springer, 2008.

[169] T. Windeatt and G. Ardeshir. Boosted ECOC ensembles for face recognition. In *Internat. Conf. on Visual Information Engineering*, pages 165–168, 2003.

[170] Terry Windeatt. Accuracy/diversity and ensemble mlp classifier design. *IEEE Transactions on Neural Networks*, 17(5):1194–1211, 2006.

[171] Terry Windeatt and Reza Ghaderi. Binary labelling and decision-level fusion. *Information Fusion*, 2(2):103–112, 2001.

[172] Terry Windeatt and Reza Ghaderi. Coding and decoding strategies for multi-class learning problems. *Information Fusion*, 4(1):11–21, 2003.

[173] Lu Xia and JK Aggarwal. Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera. In *CVPR*, pages 2834–2841, 2013.

[174] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *CVPRW*, pages 20–27, 2012.

[175] Jianchao Yang, Kai Yu, Yihong Gong, and Tingwen Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801, 2009.

[176] Xiaodong Yang and YingLi Tian. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. In *CVPRW*, pages 14–19, 2012.

[177] Xiaodong Yang and YingLi Tian. Effective 3d action recognition using eigenjoints. *Journal of Visual Communication and Image Representation*, 2013.

[178] Xiaodong Yang, Chenyang Zhang, and YingLi Tian. Recognizing actions using depth motion maps-based histograms of oriented gradients. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1057–1060. ACM, 2012.

[179] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In *NIPS*, pages 2223–2231, 2009.

[180] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[181] Chun-Xia Zhang and Jiang-She Zhang. Rotboost: A technique for combining rotation forest and adaboost. *Pattern Recognition Letters*, 29(10):1524–1536, 2008.

[182] Zhou Zhi-Hua and Yu Yang. Ensembling local learners throughmultimodal perturbation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(4):725–735, 2005.

[183] Guoqiang Zhong, Kaizhu Huang, and Cheng-Lin Liu. Joint learning of error-correcting output codes and dichotomizers from data. *Neural Computing and Applications*, 21(4):715–724, 2012.

[184] Jie Zhou, Hanchuan Peng, and Ching Y. Suen. Data-driven decomposition for multi-class classification. *Pattern Recognition*, 41(1):67–76, 2008.

[185] Jin Deng Zhou, Xiao Dan Wang, Hong Jian Zhou, Jie Ming Zhang, and Ning Jia. Decoding design based on posterior probabilities in ternary error-correcting output codes. *Pattern Recognition*, 45(4):1802–1818, 2012.

[186] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall, Boca Raton, FL, 2012.

[187] J. Zhu, S. Rosset, H. Zou, and T. Hastie. Multi-class adaboost. Technical report, Technical report, Stanford Univ. Available at http://www-stat.stanford.edu/ hastie/Papers/samme.pdf., 2006.

[188] C. Zor, T. Windeatt, and B. Yanikoglu. Bias-variance analysis of ECOC and bagging using neural nets. *Studies in Computational Intelligence*, 373:59–73, 2011.

Appendix A

Proof of the maximum number of dichotomizers in dense and sparse ECOC

Before counting the maximum number of dichotomizers in dense and sparse ECOC, we need to consider two main points:
1. From the learning point of view, each dichotomizer is equivalent to its complementary. For example, in a 4-class problem, $[1, 0, -1, -1]^T$ is equal to $[-1, 0, 1, 1]^T$. Based on the first vector, the binary classifier is trained using the instances of class one against the instances of class 3 and class 4. So, we count only one of these vectors.
2. Clearly, each valid dichotomizer includes at least one cell with 1 value and one cell with -1 value. So, $[1, 0, 0]^T$ is not a valid vector and should not be considered in counting.

In the following, we show that the maximum number of different valid dichotomizers in sparse ECOC is $(3^{N_c} - 2^{N_c+1} + 1)/2$. For dense ECOC, the proof can be conducted in similar way.

Proof:
Let, the following appropriate sets be defined:
S: The set of all possible dichotomizers of $1, 0, -1$ values and size of N_c
A: The set of all valid dichotomizers of $1, 0, -1$ values and size of N_c
A_2: The set of all possible dichotomizers of $1, 0$ values and size of N_c
A_3: The set of all possible dichotomizers of $0, -1$ values and size of N_c

Clearly, $n(A) = n(S) - n(A^c)$; where A^c is the set of all non-valid dichotomizers. Obviously, $A^c = A_2 \cup A_3$. Therefore, we have

$$n(A) = n(S) - n(A_2 \cup A_3) = n(S) - n(A_2) + n(A_3) - n(A_2 \cap A_3) \tag{A.1}$$

It can be noted that: $N(S) = 3^{N_c}$, and $n(A_2) = n(A_3) = 2^{N_c}$. In addition, $A_2 \cap A_3$ has only one member, the zero vector of length N_c. Therefore, by using Eq.(A.1),

$$n(A) = 3^{N_c} - 2^{N_c} + 2^{N_c} - 1 = 3^{N_c} - 2^{N_c+1} + 1 \tag{A.2}$$

Here, each dichotomizer and its complement have been considered in n(A). Therefore, according to the first mentioned point, the maximum number of valid different dichotomizers is equal to $n(A)/2 = (3^{N_c} - 2^{N_c+1} + 1)/2$.