

# CLASSIFICATION USING RESIDUAL VECTOR QUANTIZATION

A PhD Thesis  
Presented to  
The Academic Faculty

by

Syed Irteza Ali Khan

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy in the  
School of Electrical and Computer Engineering

Georgia Institute of Technology  
December, 2013

Copyright © 2013 by Syed Irteza Ali Khan

# CLASSIFICATION USING RESIDUAL VECTOR QUANTIZATION

Approved by:

Dr. Christopher F. Barnes, Advisor  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Dr. David V. Anderson, Co-Advisor  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Dr. Aaron D. Lanterman  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Dr. Ayanna MacCalla Howard  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Dr. Patricio Antonio Vela  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Dr. David M. Goldsman  
School of Industrial and Systems  
Engineering  
*Georgia Institute of Technology*

Date Approved: [07 October ,2013]

## ACKNOWLEDGEMENTS

It is an honor to present this thesis to my PhD committee to mark the completion of my PhD at Georgia Institute of Technology. It is a great pleasure to thank those who made this thesis possible. I owe my deepest gratitude to my PhD advisor Dr. Christopher Barnes and PhD co-advisor Dr. David Anderson for guiding and supporting me every step of the way in my PhD research. I am also grateful to my PhD proposal committee for giving me ample time to present my PhD proposal and shape my research to meet the requirements of PhD. I am thankful to my fellow students in ESP lab who gave me useful feedback to improve my thesis report. I am especially thankful to Nathan V. Parrish for his useful instructional discussions with me on some very key fundamental concepts in my research. This wonderful journey of learning would not have been possible without valuable support and guidance from the learned faculty in the School of Electrical and Computer Engineering, Georgia Institute of Technology. I also wish to thank my past teachers, especially my Kindergarten teacher Ms. Bokhari, my high school teachers Ms. Shakeela Jalali and Ms. Talat Aziz, and my physical-training instructor Mr. Gul Daraz, who inculcated the values in me that have helped lay a very solid foundation for my life. I am grateful to all my instructors in my undergraduate program at the College of Aeronautical Engineering, NUST Pakistan, who prepared me well to meet further academic challenges with success. I am especially thankful to Higher Education Commission of Pakistan for granting me the scholarship to support my PhD program at Georgia Institute of Technology.

I am indebted to my father, Syed Murtaza Ali Khan, and mother, Syeda Naseem Fatima, who have been a source of unwavering support, constant encouragement,

sincerest guidance, and inspiration to me. My father has been a special influence on me. He has always encouraged me to think outside the box and to keep an open mind. I am forever thankful to him for strengthening my resolve by always reposing a strong belief in my abilities. The period of my PhD program has been very testing on me. It has been especially hard on my wife, Fatima Ali Khan, and my children, Syeda Saher Fatima and Syed Hayyan Ali Zaidi. I feel very blessed to have a family that have stood steadfastly by me throughout my PhD program, making it a struggle of their own, and giving me the strength to persevere and complete my PhD. I am also grateful to my entire family, especially my elders and my brothers, Syed Mujtaba Ali Khan, Syed Ghazanfer Ali Khan, and Syed Muzaffer Ali Khan. I am also very thankful to my in-laws - the extended family members – especially, my father-in-law Dr. S. Iftikhar Ahmed for his calming counseling during the toughest of times in my PhD program. I dedicate my PhD to all my family members, who have all along prayed for my success and made it as a matter of family pride that I was in this PhD program. I am thankful to my friends, well wishers, and close family friends, especially Mr. and Mrs. Kirmani and Mr. and Mrs. Hashim Jafri, whose prayers were always with me.

Lastly, I offer my blessings to Georgia Institute of Technology. This great institution has provided me with a nurturing environment. Here, I have had enriching experiences and have learned values that have made a deep and positive impression on my personality. I will, forever, be obliged to this prestigious institution for playing such a positive and lasting role in my life.

# TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	iii
LIST OF TABLES	vii
LIST OF FIGURES	ix
SUMMARY	xii
<u>CHAPTER</u>	
1 CHAPTER 1 INTRODUCTION	1
2 CHAPTER 2 VECTOR QUANTIZATION	6
Introduction	6
Vector Quantization	7
Residual Vector Quantization (RVQ)	13
3 CHAPTER 3 VECTOR QUANTIZATION-BASED CLASSIFICATION	19
Introduction	19
Classified Vector Quantization	20
Learning Vector Quantization	23
Modified Tree-Search Vector Quantization for Classification	25
4 CHAPTER 4 RESIDUAL VECTOR QUANTIZATION	29
Introduction	29
Residual Vector Quantization-Based Classification	33
5 CHAPTER 5 MARKOV-BAYESIAN RESIDUAL VECTOR QUANTIZATION-BASED CLASSIFICATION: PRELIMINARY RESEARCH	35
Introduction	35
Markov-Bayesian RVQ Classification	36

Proof of Concept: Linearly Separable Synthetic Dataset	42
Proof of Concept: Linearly Non-Separable Synthetic Dataset	43
Proof of Concept: Image Dataset	53
RVQ Classification Performance Benchmark	58
Estimated Markov-Bayesian RVQ Costs	59
6 CHAPTER 6 MARKOV-BAYESIAN RESIDUAL VECTOR QUANTIZATION-BASED CLASSIFICATION: MAIN RESEARCH	61
Introduction	61
Effects of Varying Values of $M$ and $P$	62
RVQ Classification Schemes	67
Experiments and Results	69
7 CHAPTER 7 CONCLUSION AND FUTURE RESEARCH	105
Conclusion	105
Future Research	106
APPENDIX A	107
REFERENCES	123

## LIST OF TABLES

	Page
Table 1: (a) Class-conditional Transition Matrix with Markov order = 0. (b) Error matrix.	45
Table 2: (a) Class-conditional Transition Matrix with Markov order = 1. (b) Error matrix.	48
Table 3: Error matrices for (a) $M=3$ and $P=8$ , (b) $M=4$ and $P=8$ , (c) $M=5$ and $P=8$ ; Markov order = 0.	54
Table 4: Class-conditional Probability Matrices for (a) $M=3$ and $P=8$ , (b) $M=4$ and $P=8$ , (c) $M=5$ and $P=8$ ; Markov order = 0.	55
Table 5: RVQ-based classifiers-vs-SVM-based classifier.	57
Table 6: Implementation cost of RVQ classifier.	60
Table 7: Error matrices for $M = 2$ to $M = 11$ . RVQ has $P=8$ stages with the <i>zeroth</i> Markov order for classification.	64
Table 8: Classification performance of 1-NN based classifier versus Markov Bayesian RVQ classifier with $M = 7$ , $P = 8$ , and Markov Order = 4.	67
Table 9: Class-conditional Probability Matrix for RVQ with $M = 2$ and $P = 16$ . Class 1, 2, 3 are Plane, Car, Motorbike, respectively.	71
Table 10: Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has $M=2$ codevectors-per-stage and $P = 16$ stages.	72
Table 11: Class-conditional Probability Matrix for RVQ with $M = 4$ and $P = 8$ . Class 1, 2, 3 are Plane, Car, Motorbike, respectively.	74
Table 12: Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has $M = 4$ codevectors-per-stage and $P = 8$ stages.	74
Table 13: Error matrix for SVM classifier with SIFT feature and chi-squared distance kernel.	77
Table 14: Class-conditional Probability Matrix for RVQ with $M=2$ and $P=16$ . Class 1, 2, 3 are Heavy, Heavy-Sports, and Light-Sports motorbikes, respectively.	81

Table 15: Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has $M = 2$ codevectors-per-stage and $P = 16$ stages.	82
Table 16: Class-conditional Transition Probability Matrix for RVQ with $M = 4$ and $P = 8$ . Class 1, 2, 3 are Heavy, Heavy-Sports, and Light-Sports motorbikes, respectively.	84
Table 17: Error matrix for MBRVQ with RoE, Feature-cout Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has $M = 4$ codevectors-per-stage and $P = 8$ stages.	84
Table 18: Class-conditional Probability Matrix for RVQ with $M = 2$ and $P = 16$ . Class 1, 2 are Heavy, and Sports motorbikes, respectively.	87
Table 19: Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has $M = 2$ codevectors-per-stage and $P = 16$ stages.	87
Table 20: Class-conditional Probability Matrix for RVQ with $M = 4$ and $P = 8$ . Class 1, 2 are Heavy, and Sports motorbikes, respectively.	89
Table 21: Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has $M = 4$ codevectors-per-stage and $P = 8$ stages.	89
Table 22: Error matrix and classification performance measures for SVM on Graz dataset.	97
Table 23: Classification performance of MBRVQ classifier with CostERoE constraint on handwritten digits dataset RVQ has $M = 4$ and $P = 8$ , Markov order=4.	100
Table A.1. Class 1 = Heavy ; Class 2 = Sports, Training set size = 300 samples-per-class, total 600 samples. Test set size = 270 total samples, Input size = 150x250 grayscale pixels, $M = 4$ , $P = 8$ .	107
Table A.2. Class 1 = Heavy ; Class 2 = Sports, Training set size = 300 samples-per-class, total 600 samples, Test set size = 270 total samples, Input size = 150x250 grayscale pixels, $M = 2$ , $P = 16$ .	110
Table A.3 . Error matrix and classification performance measures for SVM on Graz dataset.	122



## LIST OF FIGURES

	Page
Figure 1: MSVQ block diagram ( <i>Courtesy Barnes, C.F et al. [1]</i> ).	12
Figure 2: Comparison of implementation costs of ESVQ, TSVQ, and RVQ.	16
Figure 3: Classified vector quantization.	21
Figure 4: Direct-sum codebook of RVQ with the implicit $\sigma$ -tree structure. $M$ is codevector-per-stage and $P$ is the number of RVQ stages.	29
Figure 5: Successive refinement of data and class boundaries using RVQ with $M = 3$ codevectors-per-stage and $P = 8$ stages.	32
Figure 6: $M^P = 4^8 = 65536$ Voronoi regions generated for $M=4, P=8$ RVQ.	36
Figure 7: Markov structure imposed on the stages of RVQ for classification.	37
Figure 8: Bayesian RVQ classifier.	41
Figure 9: Synthetic dataset of three classes, (a) Training set. (b) Test set. (c) Class-conditional Transition Probability Matrix of the three classes.	42
Figure 10: Synthetic Swiss roll training dataset of two classes to test and illustrate Bayesian RVQ classifier.	44
Figure 11: (a) Direct-sum codevector mapped by the training set shown in red, the remaining direct-sum codebook shown in blue. (b) Test data set. Class1 shown in blue, and Class 2 shown in red.	45
Figure 12: Training dataset: Class1 data is in blue, Class2 data is in red.	49
Figure 13: Classification performance curves for 2-category Swiss roll dataset.	50
Figure 14: Training dataset for 4-category classification using Bayesian RVQ. Class-1 data is in blue, Class-2 data is in red, Class-3 is in black, and Class-4 is in yellow.	52
Figure 15: Classification performance curves for 4-category classification using Bayesian RVQ.	52
Figure 16: Training dataset for classification using Markov Bayesian RVQ with $0^{th}$ Markov order.	53

Figure 17. RVQ codebook for $M = 4$ , $P = 8$ . 3-category training set comprises Plane, Car, and Motorbike classes.	63
Figure 18. RVQ classification performance for $M = \{2,3,4,5,6,7,8,9,10,11\}$ , and $P=8$ , and for Markov orders from 0 to $P-1 = 7$ .	65
Figure 19. Classification performance of 1-NN based classifier versus Markov Bayesian RVQ classifier with $M = 7$ , $P = 8$ , and Markov Order = 4.	66
Figure 20. RVQ codebook for RVQ with $M=2$ , $P=16$ . The dataset consists of Plane, Car, and Motorbike classes.	70
Figure 21. RVQ codebook for RVQ with $M=4$ , $P=8$ . The dataset consists of Plane, Car, and Motorbike classes.	73
Figure 22. Classification performance for different RVQ-based classifiers with $M = 2$ and $P = 16$ . The dataset consists of Plane, Car, and Motorbike classes from Caltech101.	75
Figure 23. Classification performance for different RVQ-based classifiers with $M = 4$ and $P = 8$ . The dataset consists of Plane, Car, and Motorbike classes from Caltech101.	76
Figure 24. Training dataset for 2-category classification. The classes are Heavy, and Sports motorbikes.	79
Figure 25. Training dataset for 3-category classification. The classes are Heavy, Heavy-Sports, and Light-Sports motorbikes.	79
Figure 26. RVQ codebook for RVQ with $M = 2$ , $P = 16$ . The dataset consists of Heavy, Heavy-Sports, and Light-Sports motorbike classes.	80
Figure 27. Classification performance for different RVQ-based classifiers with $M = 2$ and $P = 16$ . The dataset consists of Heavy, Heavy-Sports, and Light-Sports motorbike classes from Caltech101.	82
Figure 28. RVQ codebook for RVQ with $M = 2$ , $P = 16$ . The dataset consists of Heavy, Heavy-Sports, and Light-Sports motorbike classes.	83
Figure 29. Classification performance for different RVQ-based classifiers with $M = 4$ and $P = 8$ . The dataset consists of Heavy, Heavy-Sports, and Light-Sports motorbike classes from Caltech101.	85
Figure 30. Classification Performance for different RVQ-based classifiers with $M = 2$ and $P = 16$ . The dataset consists of Heavy and Sports motorbike classes from Caltech101.	88

Figure 31. Classification Performance for different RVQ-based classifiers with $M = 4$ and $P = 8$ . The dataset consists of Heavy and Sports motorbike classes from Caltech101.	90
Figure 32. RVQ-versus-1NN: (1 <sup>st</sup> Row) 2-category Swiss roll in Figure 13. (2 <sup>nd</sup> Row) 4-category Swiss roll in Figure 15. (3 <sup>rd</sup> Row) 3-category Caltech101 in Figure 16.	92
Figure 33. RVQ-versus-1NN: (1 <sup>st</sup> Row) 2-category Motorbike dataset from Caltech101 in Figure 23. (2 <sup>nd</sup> Row). 3-category Motorbike dataset from Caltech101 in Figure 24.	93
Figure 34. (Top): RVQ $M = 4$ and $P = 8$ . (Bottom): RVQ $M = 2$ and $P = 16$ .	95
Figure 35: Graz dataset with classes Bicycle, People, and Background.	97
Figure 36: The MNIST database of handwritten digits.	99
Figure 37. Handwritten digit database. RVQ codebook with $M = 4$ codevectors-per-stage and $P = 8$ stages.	100
Figure 38. RVQ classification performance for $M = \{2,3,4,5,6,7,8,9,10,11,12\}$ , and $P=8$ : Mean overall accuracy averaged over Markov orders 0 to 7.	101
Figure 39. RVQ classification performance for $M = 7$ and $P=8$ : overall accuracy for Markov order = $\{0, 1, 2, 3, 4, 5, 6, 7\}$ .	102
Figure 40. RVQ-versus-1NN: (Top) Computational cost. (Bottom) Memory cost.	103
Figure A.1. RVQ codebook with $M = 4$ and $P = 8$ for Graz dataset.	122

## SUMMARY

Residual vector quantization (RVQ) is a 1-nearest neighbor (1-NN) type of technique. RVQ is a multi-stage implementation of regular vector quantization. An input is successively quantized to the nearest codevector in each stage codebook. In classification, nearest neighbor techniques are very attractive since these techniques very accurately model the ideal Bayes class boundaries. However, nearest neighbor classification techniques require a large size of representative dataset. Since in such techniques a test input is assigned a class membership after an exhaustive search the entire training set, a reasonably large training set can make the implementation cost of the nearest neighbor classifier unfeasibly costly. Although, the k-d tree structure offers a far more efficient implementation of 1-NN search, however, the cost of storing the data points can become prohibitive, especially in higher dimensionality.

RVQ also offers a nice solution to a cost-effective implementation of 1-NN-based classification. Because of the direct-sum structure of the RVQ codebook, the memory and computational cost of 1-NN-based system is greatly reduced. For example, RVQ codebook with  $M = 4$  codevectors-per-stage and  $P = 8$  stages can potentially represent  $M^P = 65536$  training vectors with the cost of only  $MP = 32$  codevectors. Although, as compared to an equivalent 1-NN system, the multi-stage implementation of the RVQ codebook compromises the accuracy of the class boundaries, yet the classification error has been empirically shown to be within 3% to 4% of the performance of an equivalent 1-NN-based classifier.

# CHAPTER 1

## INTRODUCTION

Classification of images is perhaps the most important part of digital image analysis. Classification is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a *training set* of data containing observations whose category membership is known. In images, the intent of a classification process is to categorize all pixels in a digital image into one of several categories. Classification finds its application in a wide range of operations in computer vision. Computer vision [2] is a field that includes methods for acquiring, processing, analyzing, and understanding images and data from the real world to make decisions about the contents of images. Broadly speaking, the aim of computer vision is to duplicate the abilities of human vision by electronically perceiving and understanding images. The classification of images is a very important field of computer vision. The wide variety of applications of classification includes pattern recognition, object detection and recognition, and image understanding.

An important component of a classification of images is the choice of *features*. A feature, in general, is a piece of information that is relevant for solving the computational task related to a certain application. More specifically to images, a feature can refer to a simple intensity level in an image to a more complex structure like edge, line, texture, and an object. Even more complex features are manufactured to accomplish robust classification performances for object recognition. Few examples of such complex features that are very popular in computer vision applications are *Scale-invariant Feature Transform* (SIFT) [3], *Rotation-Invariant Feature Transform* (RIFT) [4], *Speeded-Up Robust Features* (SURF) [5], and *Gradient Location and Orientation Histogram* (GLOH)

[6]. The process of transforming data, such as text or images, into numerical features usable for various classification applications is called *feature selection* [7]. Feature selection has become the focus of much research, especially where huge amount of data is encountered. The objective of feature selection is three-fold: improving the classification performance of the classifiers, providing faster and more cost-effective classifiers, and providing a better understanding of the underlying process that generated the data.

An algorithm that implements classification is known as a classifier. The aim of a classifier is to separate the data in the feature space into regions belonging to each class. Broadly speaking, classifiers can be divided into two types: Separating *hyperplane*-based classifiers, or feature-template-based classifiers. In case of the former, the classifier is designed such that the data separated into class-specific regions by a plane. If the dimensionality of data is more than two, the separating plane is generically called a hyperplane. To achieve the separation between the data of different classes, it may be required to transform the data. This transformation of data required in the implementation of classification is called *feature extraction*. Artificial neural networks (ANN) [8] and support vector machines (SVM) [9] are one the widely used classifier of this type.

In feature-template-based classifiers, features are collected into a vector and then matched for nearness to the collection of exemplar features represented by training data. The criterion for nearness is called a distance function. All classifiers employ distance functions. Many types of distance functions are used, of which the most popular is Euclidean distance. Other well-known distance functions are quadratic, polynomial, chi-squared, and Earth Mover's Distance (EMD) distance functions [3]. The choice of a

distance depends on the type of features used by the classifier. Image-template matching-based classification is a special case of feature-template matching-based classification. In this case, the feature template is composed of the intensity levels of the image.

The scope of this research is the image-template matching-based classification. Various techniques based on features and image templates have been developed to achieve the various tasks of classification on a variety of images such as synthetic aperture radar, optical, satellite and infra-red images ranging from low to high resolution.

In feature-template matching-based techniques the original image is first transformed from a pixel-intensity value to another set of features. The classification is then performed on the transformed feature space. Common transformed features are edges [10], texture [11], invariant color cues [12], and invariant geometric features [13]. Stan Z. Li [14] used Markov random fields (MRF) for object recognition by employing Bayesian structural-based matching technique on linear features. Similarly, Zhang et al. [10] also used the Bayesian framework to match line features to perform object recognition.

Image-template matching-based classification is a very popular method and has certain advantages over other non-pixel features-based methods [15]. The former method uses the entire information in the image to make a class decision. Moreover, it does not involve further computations required for feature extraction. Within the area of classification based on image-template matching, the research community puts a great deal of emphasis on the techniques that decompose an image into a series of sub-images, also called basis vectors. Such decomposition also serves the purpose of image compression. Examples are principal component analysis (PCA) [16], [17] and

independent component analysis (ICA) [16], [18]. Residual Vector quantization (RVQ) [19] is also such a technique, which decomposes an image through a multi-stage, multiple *codevectors*-per-stage system that seeks to improve the image reconstruction through successive refinement of information. Other than these multi-stage frameworks, *K*-means [8] and nearest neighbor classifiers [8] are also widely used.

Classification using RVQ on image templates is the scope of this research. RVQ employs multi-stage codebooks which give it a significant advantage over the regular vector quantization VQ [20] and *k*-NN classifiers in terms of computation and memory storage requirements. Residual Vector quantization, as designed by Barnes, C.F [19], [21], [1], employs direct-sum codebook design to achieve a dense covering of the input space with low computational and memory costs. The direct-sum codebook design enables the RVQ to densely populate the input space with Voronoi regions at a relatively low cost.

*K*-means clustering is also a similar technique that partitions the input space into *K* regions [16]. However, the RVQ holds an advantage over *K*-means clustering and *k*-nearest neighbor classifier in terms of computational and memory costs, especially, in a high-dimensional input space. The multi-stage RVQ can also be held similar to the techniques like principle component analysis (PCA) [16] and independent component analysis (ICA) [16] in the sense that RVQ also decomposes an input image into stage-wise residual images. However, RVQ is, relatively, more suited to the operations of segmentation and classification since RVQ partitions the input space into Voronoi regions. RVQ has been used with a great degree of success for image-driven data mining



to detect features and objects in digital images [22], [23], and [24]. In these applications, the class-conditional probabilities are calculated for each codevector in stage codebooks. A classification decision is then made on each stage using Naive Bayes on the stage codevectors. In other words, *maximum-a-posteriori-probability*-based (MAP) rule is locally applied on stage codebooks. Subsequently, the final-class decision is made by determining the highest class-conditional probability or equivalently the maximum local (stage) MAP and assigning the corresponding class membership to the input. However in this method, classification performance with optimal rejection of false alarm is not guaranteed.

The aim of this research is to explore the Bayesian framework to formulate a solution for robust RVQ-based classification, optimal in the *maximum-a-posteriori*-probabilistic (MAP) sense. Moreover, to exploit the efficient direct-sum multi-stage structure of RVQ, the Markov approach is also explored to make the RVQ-based classification cost effective.

This thesis report is organized into seven chapters. After the first chapter on introduction and a brief background of the research presented in this report, Chapter 2 covers a discussion on vector quantization (VQ) including residual vector quantization (RVQ). The third and fourth chapters discuss the application of VQ and RVQ, respectively, in the area of classification. The fifth chapter contains the preliminary research on the proposed RVQ-based classification. Chapter 6 discusses the main research that is built upon the preliminary findings reported in Chapter 5. The conclusion and suggestions on future works on RVQ-based classification are presented in Chapter 7.

## CHAPTER 2

# VECTOR QUANTIZATION

### Introduction

In vector quantization (VQ), samples of an input are grouped into a block or vector and are encoded altogether. On the contrary, the samples are encoded individually in scalar quantization (SQ). The idea that the encoding a group of samples of an input can be advantageous over the encoding of individual samples was first put forward by Shannon in his rate-distortion theory. Rate is the average number of bits per input sample, and the measures of distortion are generally mean-squared error and signal-to-noise ratio. Shannon shows that for a given rate, vector quantization results in a lower distortion than when scalar quantization is used at the same rate. A fundamental result of Shannon's rate-distortion theory is that VQ will always achieve better compression than SQ, even if the source is memoryless, i.e., the source emits a sequence of identically- and independently-distributed random variables [25]. The reason for the superior performance of VQ over SQ is that greater flexibility exists in partitioning the input space using VQ than using SQ.

Vector quantization is a generalization of scalar quantization from the quantization of a scalar to a vector. SQ is used primarily for analog-to-digital conversion. VQ is used with sophisticated digital signal processing. VQ is usually, but not exclusively, used for the purpose of data compression. In such cases, the input signal is already in some form of digital representation of the original signal and the desired output is a compressed version of the original signal. However, VQ has also become an

important technique in speech and image recognition, and its importance and applications are growing.

A vector can be used to describe almost any type of pattern. A pattern can be formed from a segment of a speech waveform or an image, simply by forming an ordered group of samples extracted from the speech waveform or image. In such settings, VQ can be viewed as a form of pattern recognition where an input pattern is quantized and approximated by one of the patterns of a predetermined set, called the codebook. VQ can also be viewed as a front end to a variety of digital signal processing tasks, including classification and linear transformation.

### **Vector Quantization**

In vector quantization, a quantizer  $Q$  (also referred to as a vector quantizer) of dimension  $k$ , codebook  $C$ , and size  $N$  maps each source symbol or vector  $\mathbf{x} = \{x_0, x_1, \dots, x_{k-1}\}$  in  $R^k$  to a finite set  $C$  containing  $N$  distinct codevectors, i.e.,  $Q : R^k \rightarrow C \in R^k$ . The number of bits required to represent each codevector, called the resolution, code rate, or simply the rate  $r$ , is  $\log_2 N/k$  bits per vector. This process describes the encoding stage of the VQ. The second and final stage, i.e., the decoding stage, maps each codevector obtained in the encoding stage to a vector that is an approximation of the input source vector. VQ can, therefore, be considered as a pattern-matching technique since each vector is encoded by comparing it to the codevectors using a suitable distance measure.

For a given set of input symbols or a training set, the principal goal in the implementation of the VQ is to design a codebook specifying the decoder, and a partition of the input space specifying the encoder [26], while trying to minimize the average distortion over the entire training set. Several distortion measures have been proposed in

the literature. In image coding, a commonly used distortion measure is the squared-error criterion,  $d(x, y) = (x - y)^2$ , even though it does not always correlate with the human perception of quality. For an input source that emits symbols  $x_i$  and a compression system that outputs symbols  $y_j$ , the average distortion is given by

$$D = \sum_i \sum_j p(y_j|x_i)p(x_i)d(x_i, y_j) .$$

The distortion measure  $d(x_i, y_j)$  in the above equation is a measure of closeness of input and output symbols, and is generally determined by the particular application [25]. The probability  $p(x_i)$  is the distribution of the source symbols, and the posterior probability  $p(y_j|x_i)$  determines the compression scheme used.

### **VQ Codebook Design Method**

The design of a VQ codebook is done by an iteration of two steps:

a) Encoder design: Given a decoder (i.e., codebook), source distribution, and distortion measure, the optimal encoder is designed such that the encoder satisfies the nearest-neighbor condition.

b) Decoder design: Given a partition (i.e., encoder), and a squared-error criterion, the optimal decoder is designed such that the constituent codevectors of the decoder are the centroids of every cell that are made out of the given partition [26].

Steps a) and b) are repeated until some design objective is met. This formulation is the basis of the widely used Generalized Lloyd Algorithm (GLA) for VQ implementation [20]. This algorithm is also called the Linde Buzo Gray (LBG) algorithm [20].

## Types of VQ Structures

### ESVQ

We now discuss the main types of implementations of VQ that appear in the literature. The goal of a VQ implementation is that for a given rate the output distortion is as close as possible to the optimal distortion given by Shannon in his rate-distortion theory. However, in general, optimal coding of source vectors is not possible unless an exhaustive search is carried out over all the codevectors, as is done in structurally unconstrained *exhaustive search vector quantizers* (ESVQs) [21]. For a rate  $r$  and dimension  $n$  of the input vector, there are  $2^{nr}$  codevectors. The computational costs of ESVQ,  $C_{ESVQ}$ , and memory requirements  $M_{ESVQ}$  are  $\approx 2^{nr}$ . A solution to this problem is to impose constraints on the VQ structure.

### TSVQ

One possible solution is *Tree-Structured VQ* (TSVQ) proposed in [27]. A  $P$ -level,  $m$ -ary TSVQ has a search complexity  $C_{TSVQ} \approx mP$ , but double the storage requirements as compared to ESVQ, i.e.,  $M_{TSVQ} \approx 2M_{ESVQ}$ . So, although the TSVQ addresses the search complexity problem, it aggravates the storage problem.

### Product Code VQ

A method for reducing both computational and storage complexity, especially for high-dimensional vectors, is to use product-code VQ. The basic idea in product-code VQ is to break a bigger problem into several smaller problems. Examples of product-code VQ are partitioned VQ, mean-residual VQ, gain-shape VQ, and mean-gain-shape VQ [19].

### Partitioned VQ

Partitioned VQ is the simplest and most direct way of to reduce the search and storage complexity in coding a high-dimensional vector. In partitioned VQ, a vector is partitioned into two or more smaller subvectors. The training set is also partitioned into sub-training sets, and separate optimal codebooks are designed for each partitioned sub-training set. An input vector is partitioned, and each partitioned subvector of the input vector is encoded by the corresponding codebook. A major disadvantage of the partitioned VQ is that the resulting codebook fails to capture the correlation between the subvectors in the training set [28].

### Mean-Removed VQ

In mean-removed VQ, the mean of an input vector is removed, followed by the quantization of the mean and the resultant vector, called mean-removed vector, separately. The technique is effective when source input vectors are similar to each and vary one another mainly in their mean values. For example, mean-removed VQ is used for a set of similar images with differing amounts of background illumination. The effect of the varying lighting conditions can be effectively reduced by removing the mean of each before quantization. The mean and the mean-removed images are, then, quantized separately, with one codebook for the mean values and the other codebook for the mean-removed images, respectively.

### Gain-Shape VQ

In applications, such as speech, where the dynamic range of the source input is quite large, gain-shape VQ is used. For such sources, a very large codebook is needed to represent the various vectors from the source. This requirement is reduced through gain-

shape VQ, in which the source input vectors are normalized by a suitable normalization factor. The normalized vector and the normalization factor are, then, quantized separately.

### Mean-Removed-Gain-Shape VQ

Often mean-removed and gain-shape techniques are combined together in one technique that is called mean-removed-gain-shape VQ. In this method, the mean of an input vector is removed and, then, the mean-removed input is normalized by its gain to obtain a vector that is effectively normalized to have zero mean and unit gain. Codebooks designed for such mean-removed-gain-shaped vectors tend to be very robust since their dependency on an accurate statistical model of the source reduces. Mean-removed-gain-shape VQ has been extensively used in image coding.

### **Multi-Stage VQ**

Another VQ technique that has proved valuable in a number of speech and image coding applications is multi-stage VQ (MSVQ) or cascaded VQ [29]. This technique is also referred to as residual VQ (RVQ) [29]. However, in this thesis, the terminology residual vector quantization (RVQ) is exclusively reserved for the VQ technique developed by Barnes [21], [1] (the details of RVQ will be given later). The other multi-stage VQ methods are referred to, simply, as MSVQ. Juang and Gray [29] first proposed the MSVQ structure, which is shown in Figure 1.

### MSVQ

The basic idea of MSVQ is to divide the encoding task into successive stages. Each stage has its own codebook. The first stage performs a relatively crude quantization

of the input vector using its stage-wise codebook. After the quantization step at the first stage, an error vector, also called residual, is generated by subtracting the codevector

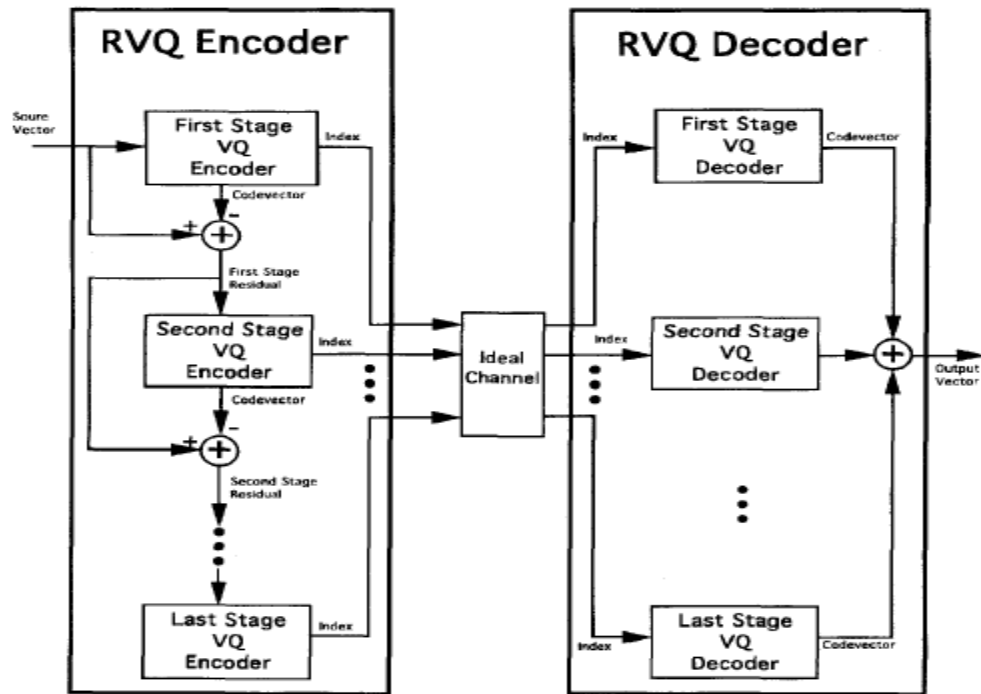


Figure 1. MSVQ block diagram (Courtesy Barnes, C.F et al. [1]).

used on the first stage from the input vector. Then, a second stage quantizer operates on the residual of the first stage and quantizes the error vector. Like the first stage, the second stage also generates the residual between the input vector of the second stage and the codevector used at the second stage. The residual after the second stage provides a second approximation to the original input vector thereby leading to a refined or more accurate representation of the original input. A third stage quantizer may be used to quantize the second stage residual to provide a further refinement and so on.

MSVQ design method constrains the parent codebook to be constructed from the direct sum of the smaller constituent stage-wise codebooks. Such codebook is called a direct-sum codebook. Direct-sum codebooks are memory efficient, in that if there are  $P$



stage-wise constituent codebooks and  $M$  codevectors per stage, then the parent codebook contains  $M^P$  code vectors, but requires the storage of only  $MP$  constituent code vectors.

Juang and Gray [29] suggested that MSVQ stages be designed by sequential application of the generalized Lloyd algorithm (GLA). Although sequential use of the GLA is nearly optimum for two-stage MSVQ with moderated to high output rates (i.e., large stage codebook sizes), this design method is increasingly unsatisfactory as the number of stages grows beyond two [19]. The main shortcoming of the sequential GLA design method is that each stage codebook is generated while considering only the error due to previous stages (the *causal* error); the error due to subsequent stages (the *anticausal* error) is ignored. This prior research demonstrated empirically that sequential nearest-neighbor encoding and suboptimal sequential design methods for direct sum codebooks generally produce rather poor results when more than two or three VQ stages are used. Subsequently, research interest in direct-sum codebooks with many stages waned, leaving the limitations of the direct-sum-codebook constraint poorly understood and the possibility of other encoding strategies unexplored. Barnes in [21], [1] proposed a novel design approach in which the codebook design technique takes into account both the causal and anticausal errors of the previous and subsequent stages to reduce the *overall* error. Barnes in [21], [1] refers to this design approach as a joint-optimal method, and the MSVQ design with this joint-optimal method is referred to as RVQ in this thesis.

RVQ is the focus of the research presented in this thesis. Therefore, the details on RVQ are given special attention and are presented as follows.

### **Residual Vector Quantization (RVQ)**

RVQ is a type of multi-stage vector quantization. RVQ is implemented with a

direct-sum codebook structure. Barnes in [21], [1] used RVQ with direct-sum, and causal-anti-causal codebook to demonstrate both low-level segmentation, and high-level object recognition.

Like MSVQ, RVQ decomposes an input vector stage-wise. This successive decomposition starts from the first stage, where an input vector is mapped to one of the codevectors in the codebook of the first stage. The mapping of the input is done according to some distance criterion. Barnes in [21], [1] used the mean-squared-error (MSE) distance measure. The mapped codevector of first stage is then subtracted from its input to yield a residual vector for the first stage. The residual is fed to the next stage as the input. The process continues for every subsequent  $p^{th}$  stage, and the respective residual vector is created by subtracting the mapped codevector  $y_p$  of the  $p^{th}$  stage from the input of that stage. This process stops if either the last stage  $P$  is reached, or when the MSE between the original input and the reconstructed input at a stage meets a pre-specified threshold. The reconstructed vector of the original input vector is obtained by summing up the corresponding  $y_p$  codevectors of all the used stages. For all the  $P$  stages of RVQ, the reconstructed image  $\hat{x}$  of the original image  $x$  is given as

$$\hat{x} = \sum_{p=1}^P y_p \quad , \quad p \in \{1,2,3, \dots, P\}.$$

The entire operation of RVQ, as mentioned above, can be summarized in the following three steps:

- a) A mapping to direct-sum codevectors: This function is a mapping from  $R^k$  to  $R^k$ , where  $k$  is the dimensionality of the codevectors and also the input space.
- b) A mapping to  $P$ -tuple representation of the direct-sum codevectors:  $P$ -tuple is a set,  $\{i_1, i_2, i_3, \dots, i_p, \dots, i_P\}$ , where  $i_p \in \{1,2, \dots, M\}$  is the index of one of

the  $M$  codevectors at the  $p^{th}$  stage of the RVQ. This mapping is a transformation from  $R^k$  to  $R^P$ .  $P$  is the total number of stages of the RVQ, and, generally,  $P \ll k$ .

- c) Mapping back from  $R^P$  to the input space  $R^k$ : The P-tuples are transformed back to in the input space to give the reconstructed image of the input image.

RVQ partitions the input space  $R^k$  into  $M^P$  Voronoi cells. The advantage of this approach is that in obtaining  $M^P$  partitions, the partitioning algorithm is run only  $P$  times and generates  $M$  partitions at each stage. In traditional VQ, the partitioning algorithm will run once but will create  $M^P$  partitions. For example, for the binary case (two code-vectors per stage,  $M = 2$ ) and a total of 8 stages ( $P=8$ ), RVQ only requires 16 searches. However, ESVQ will require 256 searches. As a result of the multi-stage implementation of RVQ, the exponential complexity in ESVQ is reduced to the linear complexity in RVQ. Moreover, even the distortion of ESVQ can be attained. In general, structurally constrained quantization cannot provide a performance as good as ESVQ. However, since they are able to more efficiently implement codes, larger vector sizes can be used, and if carefully designed, can achieve better performance than ESVQ, when compared on the basis of implementation costs [21]. The comparison between ESVQ, TSVQ, and RVQ is summarized in Figure 2.

Another question here concerns the optimality of RVQ. RVQ is said to be *jointly optimal*, also referred to as joint encoder-decoder optimal, if a local or global minimum value of the average distortion  $\mathcal{D}(E, D) = E[m(X_1, D(E(X_1)))]$  is achieved. Here,  $E$  is the encoder,  $D$  is the decoder,  $m(\cdot, \cdot)$  is a distortion metric, and  $E[\cdot]$  is the expectation operator. The necessary condition for the joint encoder-decoder optimality is that the

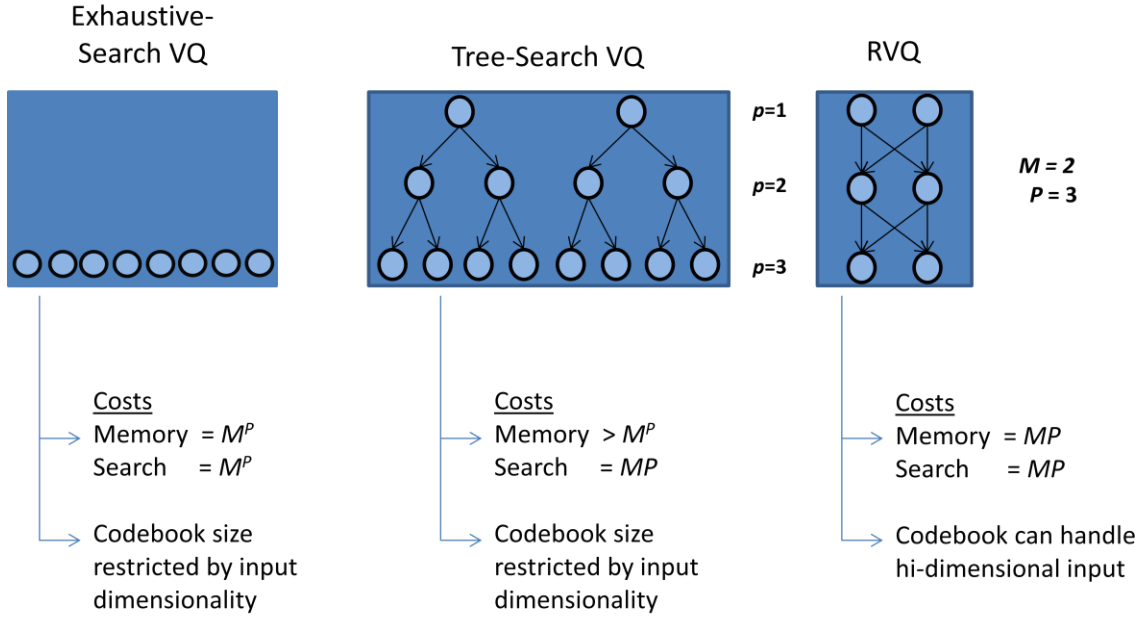


Figure 2. Comparison of implementation costs of ESVQ, TSVQ, and RVQ.

the codevectors  $y_p(i)$  of the  $p^{\text{th}}$  stage must satisfy the following condition:

$$\mathcal{D} = \frac{\partial D}{\partial y_p(i)} = 0..$$

This condition is satisfied when the stage codevectors are the centroids of residuals that are formed from the encoding decisions of both causal and anti-causal stages [21]. On the other hand, if only causal stages are considered, then satisfying the above condition will help achieve sequential optimality. For the encoder case, it is not possible to design optimal stages. Instead an overall global unconstrained encoder is designed, and then individual encoder codebooks are designed for each stage by using nearest-neighbor rules that try to match the performance of ESVQ with direct-sum codebook.

Because of the multiple-stage structure in RVQ, it is possible to implement RVQ with few codevectors per stage. This aspect of RVQ can be useful if the training data is limited

Having discussed the optimality conditions, and the general design and implementation guidelines of RVQ, we now turn our attention to the application of RVQ for classification. It is in this role that we seek to use RVQ in our research.

### **RVQ for Classification**

The use of RVQ in classification was reported in Barnes [22], [23], and in the context of image-driven data mining. In this work, a  $\sigma$ -tree is used as a data structure for RVQ stage-wise codevectors. The term sigma-tree is used to differentiate the tree structure of traditional TSVQs from the “summed” tree structure of RVQ. To better understand the  $\sigma$ -tree classifier, we relate it with other well-known areas of information theory, pattern recognition, and machine learning:

- a) The  $\sigma$ -tree classifier can be compared to dimensionality reduction methods such as Principal Components Analysis (PCA). PCA seeks to reorient the basis vectors in  $R^n$  and achieves compression by ignoring projected data components with least variances. A  $\sigma$ -tree RVQ achieves compression by encoding a source symbol with a lower dimensional tuple.
- b) The  $\sigma$ -tree classifier partitions the decision space  $R^n$  [21] like other well known classifiers such as neural networks, support vector machines and  $K$ -means clustering algorithm. Further, note that neural networks partition the decision space with hyperplanes or hypersurfaces, depending on whether or not hidden layers are used. Support vector machines also partition the decision space, but with maximum margin hyperplanes in a higher dimensional space. Like the  $K$ -means clustering scheme, the  $\sigma$ -tree classifier tessellates the decision space  $R^n$  with  $K$  Voronoi cells.

c) As already discussed, the Linde Buzo Gray (LBG) algorithm is widely used to design the encoder and decoder of a VQ, including RVQ and the  $\sigma$ -tree classifier. This algorithm is similar to the well-known  $K$ -means algorithm [21]. However, use of LBG design methods limit RVQs to typically only two stages. The greedy nature of sequential LBG design techniques prevent larger numbers of stages to be designed that give acceptable performance gains with additional stages.

Having introduced RVQ, we now turn to a more comprehensive discussion on the applications of VQ for classification and pattern recognition.

## CHAPTER 3

### VECTOR QUANTIZATION-BASED CLASSIFICATION

#### Introduction

As discussed in Chapter 2, vector quantization (VQ) is a signal compression technique. However, use of VQ for the representation of input vectors in terms of the codevectors provides a natural basis for segmentation and classification. VQ partitions the input space into Voronoi regions. The inputs are mapped to these Voronoi regions with calculable prior and conditional class-conditional probabilities. Therefore, a test input can be classified by applying the maximum-*a posteriori*-probability rule. As a result, VQ has become a relevant technique in speech and image recognition, and its importance and applications are growing [22], [29].

Vector quantization has been implemented in a variety of ways in the applications of classification and pattern recognition [30]. In one implementation, namely classified vector quantization (CVQ) [31], a VQ-based classifier is used as pre-processing step for improved compression and signal representation. In learning vector quantization (LVQ), the codevectors of the VQ are such placed in the input space that the classification performance is maximized [30]. In other words, the codebook is designed to approximate the Bayes decision boundaries in the input space, and the compression performance is not given the top priority. Vector quantization is also implemented by designing stage codebooks, called multistage VQ (MSVQ). This implementation significantly reduces the computation cost of the VQ. MSVQ is a general design methodology that can be used to implement both CVQ and LVQ [30]. The latter is also called modified tree search VQ

(MTSVQ). However, MTSVQ are designed to maximize both the compression and classification performances of the VQ.

After briefly describing the various types of VQ-based classifiers, a more detailed explanation of these implementations is discussed one-by-one in the following paragraphs.

### **Classified Vector Quantization**

Classified vector quantization (CVQ) [31] is a method for improved data compression performance, not for classification. The focus of the improved encoding for data compression is to reduce degradation of certain features in the signals. For example, in [31] CVQ is used encode the edges more efficiently than the other features in the images. In CVQ, the VQ codebook is composed of sub-codebooks. A sub-codebook or equivalently a class-specific VQ encoder is designed by training the VQ on the images from that particular class. In CVQ, an input is classified before being fed to the sub-codebook of that class for class-specific encoding. The encoder has two components: (1) classifier, and (2) encoder class-specific codebooks. In [31], CVQ is designed with the primary function of edge enhancement in the input images. Firstly, the classifier categorizes an input image into one of the pre-specified classes. The pre-classified image is, then, encoded by the VQ that is specifically designed for that class. The encoded image is decoded by a decoder. The decoder is simply a lookup table that decodes the encoded input image to produce the corresponding reconstructed images. Figure 3 illustrates the generic functional blocks of a CVQ encoder. There are  $M$  classes, and if the input belongs to class  $i$ , the  $i^{\text{th}}$  sub-codebook  $C_i$  of size  $N_i$  is employed to encode the input, using a distortion measure  $d(\cdot, \cdot)$ . A distortion measure is a measure of how close



an arbitrary input is to the codevectors in the codebook. In general, the distortion measures for different classes may be different. The total number of codevectors is  $\mathbb{N} = \sum_{i=1}^M N_i$ , with encoding indices ranging from 1 to  $\mathbb{N}$ .

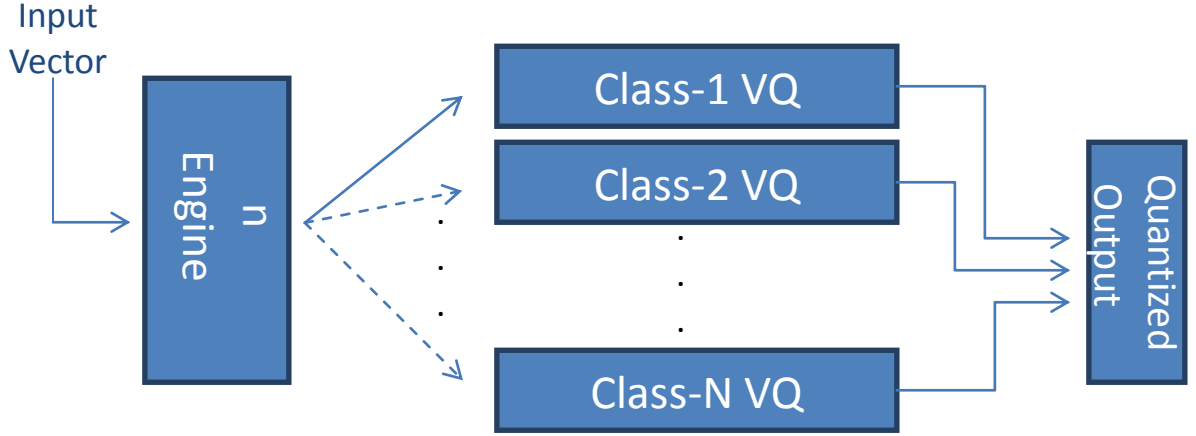


Figure 3. Classified vector quantization.

The encoding index of the nearest codevector is transmitted to the decoder. The decoder simply looks up the corresponding codevector from its codebook of size  $\mathbb{N}$  and generates the reconstructed image of the input. An important factor in the design of the individual codebooks is the optimal size of the codebooks for a given average distortion for the respective class. It is shown that the optimal average distortion  $D_i^*$  for a given class-specific codebook and the optimal codebook size  $N_i^*$  for that class, is given as

$$\frac{P_i D_i^*}{N_i^*} = \text{constant},$$

where  $P_i$  is the probability of occurrence of the images from the  $i^{\text{th}}$  class. So, by trial and error  $N_i^*$  and the corresponding  $D_i^*$  can be found that satisfy the above condition. The size of the codebook of a particular class implicitly assigns a corresponding weight to that

class. The higher the size of the codebook, the higher importance is given to the images from that class.

CVQ is based on a composite source model for images. This composite source model, especially for edge perception [31], has a firm basis in the psychophysics of human-vision system. In this model the image source is viewed as a bank of class-specific image sub-sources. It is assumed that each input image comes from one of these class-specific image sub-sources. Therefore, in the design of VQ codebook with this model, a separate VQ codebook is designed for each sub-source. In other words, separate VQ codebooks are designed for each class of the images using the LBG algorithm.

It should be noted that in [31] CVQ is a method for improved encoding performance, not for classification. The focus of the improved encoding is to reduce degradation of edges in images. The classification is performed once per input vector, and its complexity is negligible in comparison with that of the encoding. The encoding complexity of the CVQ is greatly reduced as compared with that of a regular VQ with the same average distortion measure. Similar applications of edge enhancement using CVQ are also reported in [32] and [33]. However, in [32], the sub-codebooks are designed with the Fuzzy C-Means method [34], and the CVQ sub-codebooks in [33] are designed the same way as in [32], but with a quad-tree pre-classifier. Moreover, in [33], the indices generated by CVQ are used for the application of image retrieval. In the image retrieval application, the images similar to a query image are retrieved based on a similarity measure.

Other than the application of edge enhancement, CVQ has also been integrated into a system designed for classification. In [35], CVQ sub-codebooks are designed on

pre-classified SIFT features [3] of edges and smooth regions in the images in the training set to generate bag-of-words (BOW) histogram features [36]. After the CVQ step, SVM [9] classifier is trained on the BOW histogram features of the two classes in the training set. In [37], CVQ sub-codebooks are designed for four classes: car, van, light truck, and bus. Multilayer perceptron (MLP) [38] is used as the pre-classifier before CVQ. For a given test image, the class-membership is assigned to that class for which the corresponding class-specific CVQ sub-codebook gives the least reconstruction error.

### **Learning Vector Quantization**

Learning vector quantization LVQ was invented by Teuvo Kohonen [30]. Is a *supervised* version of vector quantization in which the VQ codebook is designed with the training data having known class associations, also called labeled data. The learning technique for LVQ uses the class information of the labeled training data to position the codevectors such that the quality of the classifier decision boundary improves.

The training method of learning vector quantization is often called *competition learning*, because it works as follows: For each training data the codevector that is closest to it is determined. The direction of the change in the location of the codevector, also called *adaptation*, depends on whether the class of the training data and the class assigned to the codevector coincide or not. If they coincide, the codevector is moved closer to the training data; otherwise it is moved farther away. This movement of the codevector is controlled by a parameter called the *learning rate*, which is usually made to decrease monotonically with time. Since the learning rate is usually decreasing, the initial changes in the positions of the codevectors are larger than the changes made in later

epochs of the training process. Learning may be terminated when the positions of the codevectors hardly change anymore.

The earlier implementation of LVQ has been carried out with multiple algorithms, namely LVQ are LVQ1, LVQ2, and LVQ3 [30]. These three algorithms yield almost similar accuracies in most statistical pattern recognition tasks. LVQ1 and LVQ3 offer a more robust solution to the optimization of class boundaries. The learning rate can be optimized for quick convergence. However, LVQ3 differs from LVQ1 in the number codevectors involved in the update of the class boundaries. In LVQ1, only the nearest codevector is moved, but in LVQ2, two nearest codevectors, one belonging to the same class and the other belonging to a wrong class, are simultaneously updated. This process is also called *differential* shifting. LVQ2 also employs differential shifting. However, unlike LVQ3, the update of the class boundaries is not guaranteed to converge, if the LVQ2 algorithm is allowed to run over a long period of time.

### **Advanced LVQ**

The original methods for LVQ are based on the Euclidean distance in the optimization of class boundaries. The Euclidean distance assumes certain geometric properties in the training data, which, at times, may not be realistic. For this reason, extensions of the methods to more general distance functions have been proposed that are called generalized LVQ (GLVQ) [39], [40], and [41]. In these GLVQ algorithms distance function parameters are learned based on the given classification task such that a data-driven distance function is found. Consequently, the class boundaries are more accurately drawn that results in a significant improvement in classification accuracy. Another

critical advantage of GLVQ over the regular LVQ is that the increased accuracy in the formation of the class boundaries is achieved with lesser number of codevectors.

Learning vector quantization has been used for the classification of textual documents [42], [43], where LVQ network is used for classifying text documents. In general, LVQ require less training examples and are much faster than other classification methods. The experimental results show that the LVQ approach outperforms  $k$ -NN, and is comparable to SVM

### **Modified Tree-Searched Vector Quantization for Classification**

Because of the implicit connection between compression and classification, VQ can be considered as a framework which can be optimized to both compress and classify. Two applications of this concept are shown.

In [44], [45] compression and classification are combined in one single process of codebook design in vector quantization (VQ). The method is generally referred to as modified tree-searched vector quantization (MTSVQ). As the name of the method suggests, vector quantization is implemented with tree-searched vector quantization (TSVQ). TSVQ is a multi-stage, sigma-tree implementation of VQ. The reason to use TSVQ over the regular VQ is greatly reduced computational complexity in TSVQ. Furthermore, unbalanced TSVQ, which is a variable-rate coding method, can also be employed. Unbalanced TSVQ generally produces lower average distortion than balanced TSVQ. Trained data sets are used to design codebooks that achieve both small average distortion measure and good classification performance.

Purely for compression, the codebook of TSVQ is designed by splitting the node which reduces the average distortion until the average distortion reaches a pre-specified

value. However, in the design of MSTVQ, the optimization of TSVQ codebook is modified to achieve good classification along with good compression. As mentioned earlier, the efficacy of simultaneous compression and classification has been demonstrated through two applications: (1) Classification of the regions-of-interest in the compressed images, and (2) Enhancement of certain features in an image that are important in some sense.

In [44], binary classification was carried between man-made or natural structures. The training set consisted of five 512 x 512 grayscale images of aerial photography of the San Francisco Bay area. These images were provided by ESL, Inc. Each image was divided into 16 x 16 sub-blocks. The 16 x 16 sub-blocks were hand-labeled as either man-made or natural. However, to further reduce TSVQ complexity, the codebook construction based on the classified training set is done on a coarser resolution scale of 4 x 4 pixel blocks of the 16 x 16 sub-blocks. This coarse resolution could have affected the classification of this classifier.

For the design of the TSVQ codebook, primarily two separate splitting criteria are employed in growing the tree: 1) split the node that provides the largest ratio of decrease in distortion to increase in rate. The distortion measure is the mean squared error. 2) Split the node that has the greatest percentage of misclassified training vectors. This corresponds to measuring the distortion by the hamming distance between the node class and the hand-labeled class of the input training vector.

In general, the first splitting criterion provided the lowest distortion in the encoded images at the cost of comparatively poor classification performance. The second splitting criterion achieved better classification but poorer reconstruction. The choice

between the two splitting criteria involves a trade-off between rate and memory requirements. The classification of TSVQ is compared against Classification and Regression Trees (CART) [46]. From the results, it is concluded that the two classifiers are generally comparable. CART performed better on original images. On the other hand, CART performed worse on the compressed images.

In the application of image enhancement, regions of certain classes are enhanced. This is done by introducing weighted distortion measure. The basic idea is to achieve better encoding of the regions belonging to certain classes-of-interest assigning them higher weights. MSE is used as a distortion measure. Higher weights are assigned to the classes that are deemed more important. Similarly, the classes that are of lesser interest are assigned lower weights in the distortion measure. Assigning higher weights to certain classes contributes to higher distortion measure for those classes, and, as a result, the tree grows deeper for those classes, thus, achieving lower reconstruction errors for those classes.

MSTSVQ have been tested on MRI scans, and textured data with vector quantization done on non-overlapping 2x2 and 4x4 sub-blocks, respectively, in the images [45]. For MRI scans, bright training vectors are weighted more heavily than dark one. On comparison with non-weighted TSVQ, the weighted TSVQ encoded the bright regions in the images better.

For the textured data, the training and test images were taken from USC database [45]. Each image was divided into 4 x 4 non-overlapping sub-blocks. Highly textured data were assigned lower weights, and highly homogenous vectors were given higher weights. The test images showed clouds, a lake, and trees. A comparison of compressed

images encoded from the weighted and non-weighted tree showed that the cloud regions had less distortion, whereas, the tree areas had high texture and, therefore, had more distortion.

Another implementation of VQ that also uses multi-stage tree structure of TSVQ is called residual vector quantization (RVQ). RVQ has been used in the applications of classification and pattern recognition [22] [23]. The focus of this thesis is also RVQ-based classification. A discussion on RVQ-based classifiers is presented in the next chapter.



# CHAPTER 4

## RESIDUAL VECTOR QUANTIZATION

### Introduction

Residual vector quantization (RVQ) is a type of multi-stage vector quantization (MSVQ). RVQ is implemented with a direct-sum codebook structure [19]. Barnes in [22], [23], and [24] used RVQ with direct-sum, and causal-anti-causal codebook to demonstrate both low-level segmentation, and high-level object recognition. As mentioned earlier, RVQ is designed with stage codebooks with the direct-sum data structure that implicitly implements the RVQ book in a directly summed tree structure, also often called  $\sigma$ -tree. The direct-sum codebook of RVQ with its implicit tree structure is shown in Figure 4.

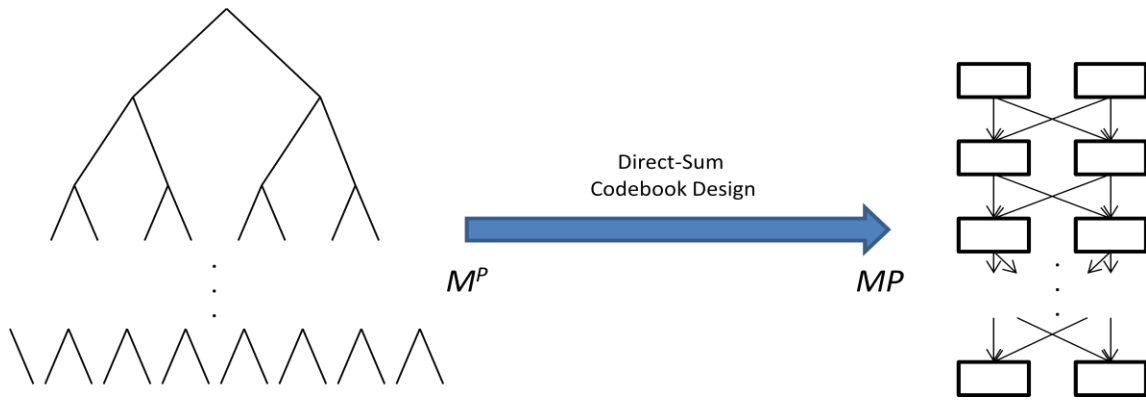


Figure 4. Direct-sum codebook of RVQ with the implicit  $\sigma$ -tree structure.  $M$  is codevector-per-stage and  $P$  is the number of RVQ stages.

Like MSVQ, RVQ decomposes an input vector stage-wise. This successive decomposition starts from the first stage, where an input vector is mapped to the nearest codevector in the codebook of the first stage. The mapping of the input is done according to some distance criterion. Barnes in [21] and [1] used the mean-squared-error (MSE)

distance measure. The mapped codevector of first stage is then subtracted from its input to yield a residual vector for the first stage. The residual is fed to the next stage as the input. The process continues for every subsequent  $p^{th}$  stage, and the respective residual vector is created by subtracting the mapped codevector  $y_p$  of the  $p^{th}$  stage from the input of that stage. This process stops if either the last stage  $P$  is reached, or when the MSE between the original input and the reconstructed input at a stage meets a pre-specified threshold [19]. The reconstructed vector of the original input vector is obtained by summing up the corresponding  $y_p$  codevectors of all the used stages. For all the  $P$  stages of RVQ, the reconstructed image  $\hat{x}$  of the original image  $x$  is given as

$$\hat{x} = \sum_{p=1}^P y_p \quad , \quad p \in \{1,2,3,\dots,P\}.$$

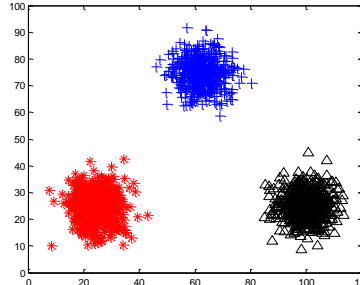
RVQ partitions the input space  $R^k$  into  $M^P$  Voronoi cells. The advantage of this approach is that in obtaining  $M^P$  partitions, the partitioning algorithm is run only  $P$  times and generates  $M$  partitions at each stage. In traditional VQ, the partitioning algorithm would run once but created  $M^P$  partitions. For the binary case (two code-vectors per stage,  $M = 2$ ) and a total of 8 stages ( $P=8$ ), RVQ only requires 16 searches. However, ESVQ will require 256 searches. As a result, the exponential complexity is reduced to the linear complexity. In general, structurally constrained quantization cannot provide a performance as good as ESVQ. However, since they are able to more efficiently implement codes, larger vector sizes can be used, and if carefully designed, can achieve better performance than ESVQ with higher RVQ dimensionality and fixed implementation costs [15].

One interesting consequence of RVQ's process of successive stage refinement of data is the progressive evolution of partitioning of the input space into Voronoi regions.

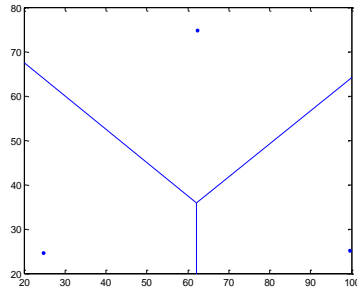
RVQ aims at refining the data with each added stage. This process is also equivalent to refining the partitioning of the input by adding Voronoi regions with each added stage. As a result, the class boundaries between the data of different categories are also refined, making the class-specific regions more and more compact, thus provide a potential for improved accuracy in the classification of data.

The effect of the stage-wise refinement of data by RVQ on the Voronoi regions in the input space is explained with the example illustrated in Figure 5a. In this example, three clusters of linearly separable 2D data points are considered, shown in Figure 5a in red, blue, and black. RVQ codebook is designed on the data points with  $M = 3$  codevectors-per-stage and  $P = 8$  stages. The Voronoi regions, with the direct-sum codevectors as the centroids of these regions, are shown for  $M = 3$  and the values of  $P$  starting 1 to 8. It is to be noted that the number of direct-sum codevectors at a given  $n^{\text{th}}$  stage is  $M^n$ , equivalently  $M^n$  Voronoi regions. It can be seen that for Figure 5b,  $M = 3$  and stage  $n = 1$ , the number of the corresponding direct-sum codevectors, shown as blue dots, and the Voronoi regions is  $M^n = 3^1 = 3$ . Similarly, for stage  $n = 2$ , there are  $M^n = 3^2 = 9$  direct-sum codevectors and Voronoi regions. Likewise, for the remaining values of  $n \in \{3, 4, \dots, P = 8\}$ , the number of the direct-sum codevectors and the Voronoi regions is  $M^n$  in Figure 5c. It can also be observed that with each added stage, the representation of the data improves. Furthermore, the class boundaries between the three classes are also progressively refined after each successive stage.

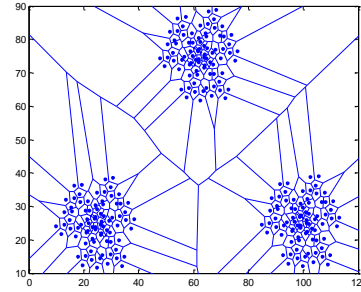
It has been shown that the successive refinement of data through the stage-wise implementation of RVQ codebook can also aid in progressive improvement in classification of the data. Therefore, the intrinsic structure of RVQ codebook holds



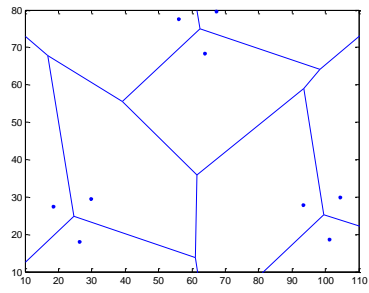
(a) Linearly separable 3-class dataset



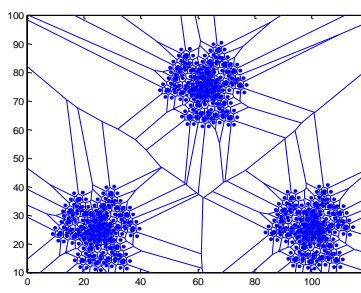
(b)  $M = 3, n = 1$



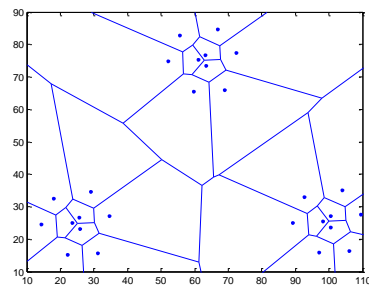
(f)  $M = 3, n = 5$



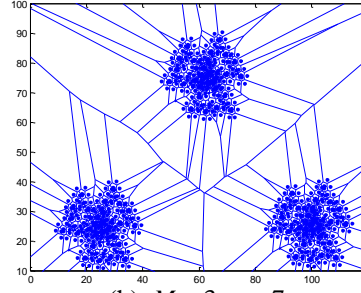
(c)  $M = 3, n = 2$



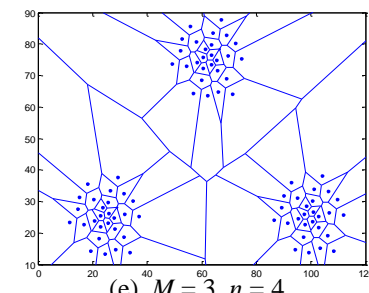
(g)  $M = 3, n = 6$



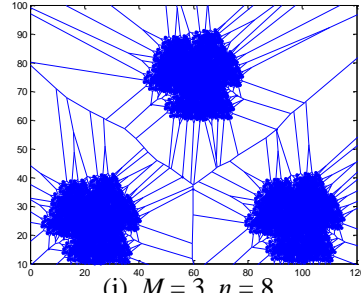
(d)  $M = 3, n = 3$



(h)  $M = 3, n = 7$



(e)  $M = 3, n = 4$



(i)  $M = 3, n = 8$

Figure 5. Successive refinement of data and class boundaries using RVQ with  $M = 3$  codevectors-per-stage and  $P = 8$  stages.

immense potential in its use as a classifier.

### **Residual Vector Quantization-Based Classification**

The use of RVQ for segmentation and classification was reported in Barnes [22], [23], and [24]. As mentioned earlier, RVQ is designed with stage codebooks with the direct-sum data structure that implicitly implement the RVQ book in a tree structure, also often called a  $\sigma$ -tree. To better understand the  $\sigma$ -tree classifier, we relate it with other well-known areas of information theory, pattern recognition, and machine learning:

- a) The  $\sigma$ -tree classifier can be compared to dimensionality reduction methods such as Principal Components Analysis (PCA). PCA seeks to reorient the basis vectors in  $R^k$  and achieves compression by ignoring projected data components with least variances. A  $\sigma$ -tree RVQ achieves compression by encoding a source symbol with a lower dimensional tuple.
- b) The  $\sigma$ -tree classifier partitions the decision space  $R^k$  like other well known classifiers such as neural networks, support vector machines and  $K$ -means clustering algorithm. Further, note that neural networks partition the decision space with hyperplanes or hypersurfaces, depending on whether or not hidden layers are used. Support vector machines also partition the decision space, but with maximum margin hyperplanes in a higher dimensional space. Like the  $K$ -means clustering scheme, the  $\sigma$ -tree classifier tessellates the decision space  $R^k$  with  $K$  Voronoi cells.
- c) As already discussed, the Linde Buzo Gray (LBG) algorithm is widely used to design the encoder and decoder of a VQ, including RVQ and the  $\sigma$ -tree classifier. This algorithm is similar to the well-known  $K$ -means algorithm [20].
- d) The stage indices,  $P$ -tuples, returned by the  $\sigma$ -tree classifier can be used to

probabilistically determine class membership. The size of code-book of the RVQ can be manipulated to implement effective multi-class classification. The multiple stages of the  $\sigma$ -tree classifier suggest a coarse to fine classification route, i.e., successive approximations.

So far the use of RVQ as a classifier has relied on heuristic methods [22], [23], and [24]. The class decisions are made stage-wise by applying maximum-*a posteriori*-probability (MAP) rule on the classification results at each stage, thus yielding stage-wise MAP class decisions. The final class decision is made by assigning the input to the class with the maximum MAP. This method, though heuristic, exhibits the stage-wise contributions of RVQ to classification.

However in this method, classification performance with optimal rejection of false alarm is not guaranteed. If the stage-wise decision making of RVQ is extended to include more than one stage considered together, RVQ-based classification can expectedly be made more robust with increased performance. The aim of this research is to explore the Bayesian framework to formulate a solution for robust RVQ-based classification, optimal in the maximum-*a posteriori*-probabilistic (MAP) sense over multiple RVQ stages. Moreover, to exploit the efficient direct-sum multi-stage structure of RVQ, the Markov approach is also explored to make the RVQ-based classification cost effective.

The next step in this research is to design a method that could integrate the stage-wise contributions of RVQ into a robust solution to classification. This is the topic of this research. Bayesian framework provides the capacity to deliver a structure that can model the combined stage-wise class decisions of RVQ to give an elegant RVQ-based classifier, optimal in the MAP sense in a way that spans all RVQ stages.

## CHAPTER 5

# MARKOV-BAYESIAN RESIDUAL VECTOR QUANTIZATION- BASED CLASSIFICATION: PRELIMINARY RESEARCH

### Introduction

The applications of Residual vector quantization (RVQ) in classification and segmentation were first reported in Barnes [22], [23], and [24] on image databases. These applications of RVQ have been explained in Chapter 4. The contribution of this research is the formulation of a framework for RVQ to combine the individual stage classification results together into a joint classification decision rule over all the stages of RVQ using multiple stage MAP decision rules. In this thesis, this classification framework is referred to as Markov residual vector quantization (MRVQ).

MRVQ is tested on a variety of standard image datasets such as Caltech101 [47], Graz [48], and handwriting [49], [50] datasets. After conducting a number of experiments on these datasets, it is empirically noted that the classification of MRVQ is upper bounded by 1-NN classifier, with MRVQ typically approaching within XX percent of 1-NN performance with MRVQ orders of XX. However, RVQ holds a clear advantage over 1-NN in the implementation cost, typically returning computational costs savings of XX orders of magnitude and XX orders of magnitude in memory costs

MRVQ is implemented with two different schemes: Feature-count rule, and Bayesian rule. The latter is also termed as Markov-Bayesian RVQ (MBRVQ). These two schemes for implementing RVQ-based classification and the preliminary research that led to the development of these two methods are explained next.

## Markov-Bayesian RVQ Classification

As explained in Chapter 4, the encoding process of RVQ is equivalent to partitioning the input space into Voronoi regions. The mapping of the training dataset can determine the class-conditional probabilities for each class of data in the dataset. In other words, for each class, a class-conditional probability can be assigned to each Voronoi region, and the Voronoi regions can be labeled with a class decision by applying maximum *a priori* probability (MAP) rule. As a result, any test input, which maps to any of the labeled Voronoi regions, or equivalently to a direct-sum RVQ codevector, can be classified into one of the classes by applying the MAP rule.

However, to apply the MAP rule to RVQ, it entails that all the Voronoi regions be labeled. For RVQ with  $M$  stages and  $P$  codevectors-per-stage,  $M^P$  Voronoi regions have to be labeled. If  $M=4$  and  $P=8$ ,  $4^8 = 65536$ , Voronoi regions tessellating over the input space have to be labeled for the MAP-based classification rule, as shown in Figure 6, below.

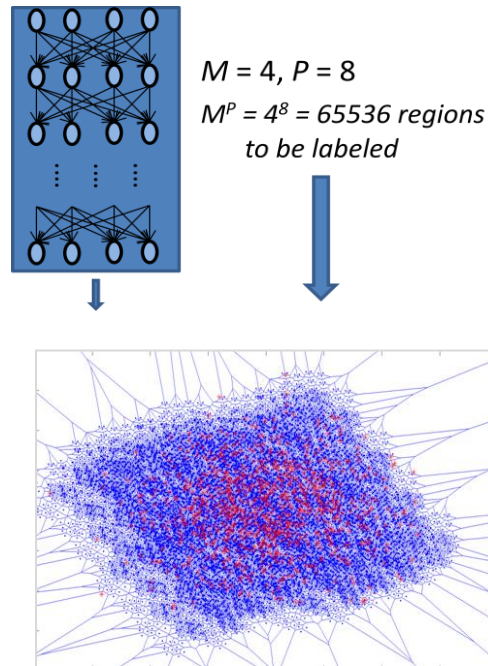


Figure 6.  $M^P = 4^8 = 65536$  Voronoi regions generated for  $M=4, P=8$  RVQ.

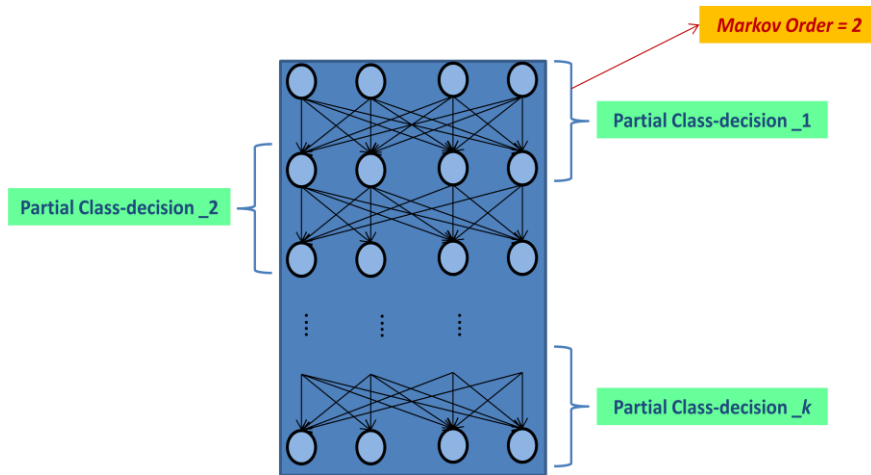


The number of Voronoi regions to be labeled exponentially increase for increasing values of  $M$  and  $P$ . Consequently, the memory required to store all the labels can become very large.

One way to reduce the cost associated with RVQ-based classification is to impose *Markovianity* on the stages of RVQ. By doing so, the total  $P$  number of stages of RVQ can be divided into groups of  $N$  independent stages, where  $N$  can be made to vary from 1 to  $P$ .  $N$  is called the Markov order of the RVQ classifier. With this formulation, a local class decision can be achieved for each group of  $N$  independent stages. The local class decisions can then be combined together to give the final class decision. The classification cost, because of the Markov structure on the RVQ, is expected to be as follows:

$$\text{Classification Cost} = \text{Order}(M^N), \text{ where } N < P, \text{ \& } M^N \ll M^P.$$

The Markov structure imposed on the RVQ is illustrated in Figure 7, below.



$$\text{Final Class-decision} = F(\text{Partial Class-decision}_1, \text{Partial Class-decision}_2, \dots, \text{Partial Class-decision}_k)$$

Figure 7. Markov structure imposed on the stages of RVQ for classification.

The mathematical formulation of RVQ-based classification is done using the Bayesian framework along with Markovian structure imposed on the stages of RVQ. The mathematical treatment to the classification using RVQ is given as follows.

According to the Bayes' theorem, for the random variables  $A$  and  $B$ , the posterior probability  $P(A|B)$  is

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Under the Bayesian framework, RVQ-based classification can be modeled as the posterior probability

$$P(c_i|CV_1, CV_2, CV_3, \dots, CV_P) = \frac{P(CV_1, CV_2, CV_3, \dots, CV_P|c_i)P(c_i)}{P(CV_1, CV_2, CV_3, \dots, CV_P)}. \quad (1)$$

$c_i$  is the class decision with  $c_i \in \{c_1, c_2, c_3, \dots, c_C\}$ , and  $i \in \{1, 2, 3, \dots, C\}$ . The total number of classes is  $C$ .  $CV_p$  is the stage codebook at the  $p^{\text{th}}$  stage, with  $p \in \{1, 2, 3, \dots, P\}$ , where  $P$  is the total number of RVQ stages.  $CV_p \in \{CV_p^1, CV_p^2, CV_p^3, \dots, CV_p^M\}$ , where  $CV_p^m$  is the  $m^{\text{th}}$  codevector in the stage codebook at the  $p^{\text{th}}$  stage, and  $m \in \{1, 2, 3, \dots, M\}$ , where  $M$  is the number of codevectors in a stage codebook of RVQ. It is assumed  $c_i$  is equiprobable, and

$$P(CV_1, CV_2, CV_3, \dots, CV_P) = \sum_{i=0}^C P(CV \cap c_i) = \text{CONSTANT},$$

where  $CV = \{CV_1, CV_2, CV_3, \dots, CV_P\}$ . Therefore, the posterior probability in Equation (1) can be reduced as follows:

$$P(c_i|CV_1, CV_2, CV_3, \dots, CV_P) \propto P(CV_1, CV_2, CV_3, \dots, CV_P|c_i).$$

As a result, the class decision  $c^*$  is given as

$$c^* = \text{argmax}_i P(CV_1, CV_2, CV_3, \dots, CV_P|c_i). \quad (2)$$

As mentioned before, Markov structure if imposed on the RVQ classifier, modeled with the Bayesian framework of Equation (2), can reduce the implementation cost of the RVQ classifier. When Markov condition is imposed on the stages of RVQ, it is based on an assumption that the classification at given RVQ stage is dependent on the classification results of a certain number of previous RVQ stages. The Markov order determines the number of dependent RVQ stages. If Markov order is zero, the classification results of every stage are independent of each other. Similarly, for Markov order equal to one, the classification at a given stage is dependent only on the classification at the previous stage. A useful consequence of the Markov condition is that a given dependent on the corresponding previous stages becomes independent of the higher stages of RVQ. Therefore, to get the final class decision, the independent classification results are simply multiplied together to get the solution.

To explain the formulation of Markov structure on Equation (2), the mathematical development is shown on  $P(CV_1, CV_2, CV_3, \dots, CV_P)$ , and it will be extended to the likelihood function  $P(CV_1, CV_2, CV_3, \dots, CV_P | c_i)$  in Equation (2). The Markov model is shown as follows:

According to the Markov chain rule,

$$P(CV_1, CV_2, CV_3, \dots, CV_P) = P(CV_1)P(CV_2|CV_1)P(CV_3|CV_2, CV_1)P(CV_4|CV_3, CV_2, CV_1) \dots P(CV_P|CV_{P-1}, CV_{P-2}, \dots, CV_1). \quad (3)$$

$0^{\text{th}}$  Markov order.

$$P(CV_1, CV_2, CV_3, \dots, CV_P) = P(CV_1)P(CV_2)P(CV_3)P(CV_4)P(CV_5)P(CV_6) \dots P(CV_P), \text{ and}$$

$$P(CV_1, CV_2, CV_3, \dots, CV_P | c_i) = P(CV_1 | c_i)P(CV_2 | c_i)P(CV_3 | c_i)P(CV_4 | c_i)P(CV_5 | c_i) \dots P(CV_6 | c_i) \dots P(CV_P | c_i).$$

The  $0^{\text{th}}$  Markov order in Equation (3) means that it is assumed that the class decisions made at each stage are independent of each other.

1<sup>st</sup> Markov Order.

$$P(CV_1, CV_2, CV_3, \dots, CV_p) = \frac{P(CV_1)P(CV_2|CV_1)P(CV_3|CV_2)P(CV_4|CV_3)P(CV_5|CV_4) \dots}{P(CV_p|CV_{p-1})}, \text{ and}$$

$$P(CV_1, CV_2, CV_3, \dots, CV_p|c_i) = \frac{P(CV_1|c_i)P(CV_2|CV_1, c_i)P(CV_3|CV_2, c_i)P(CV_4|CV_3, c_i) \dots}{P(CV_5|CV_4, c_i) \dots P(CV_p|CV_{p-1}, c_i)}.$$

By merging the first two shaded probabilities into the respective joint probability, the above equation can equivalently be written as

$$P(CV_1, CV_2, CV_3, \dots, CV_p|c_i) = \frac{P(CV_1, CV_2|c_i)P(CV_3|CV_2, c_i)P(CV_4|CV_3, c_i)P(CV_5|CV_4, c_i) \dots}{\dots P(CV_p|CV_{p-1}, c_i)}.$$

The 1<sup>st</sup> order Markov in equation (4) implies that each stage codevector is assumed to be dependent only on the previous stage.

2<sup>nd</sup> Markov Order

$$P(CV_1, CV_2, CV_3, \dots, CV_p) = \frac{P(CV_1)P(CV_2|CV_1)P(CV_3|CV_2, CV_1)P(CV_4|CV_3, CV_2) \dots}{P(CV_p|CV_{p-1}, CV_{p-2})}.$$

By merging the first three shaded probabilities into the respective joint probability, the above equation becomes

$$P(CV_1, CV_2, CV_3, \dots, CV_p) = \frac{P(CV_1, CV_2, CV_3)P(CV_4|CV_3, CV_2) \dots \dots P(CV_p|CV_{p-1}, CV_{p-2})}{\dots}$$

q<sup>th</sup> Markov order

The class-conditional probability  $P(CV_1, CV_2, CV_3, \dots, CV_p|c_i)$  can be generalized for an arbitrary Markov order  $q$  as follows:

$$P(CV_1, CV_2, CV_3, \dots, CV_p) = P(CV_1, CV_2, \dots, CV_{q+1}) \prod_{p=q+2}^P P(CV_p|CV_{p-1}, CV_{p-2}, \dots, CV_{p-q}),$$

and the class-conditional probability  $P(CV_1, CV_2, CV_3, \dots, CV_p|c_i)$  for the  $q^{\text{th}}$  Markov order is

$$P(CV_1, CV_2, CV_3, \dots, CV_p|c_i) = \frac{P(CV_1, CV_2, \dots, CV_{q+1}|c_i)}{\dots} \prod_{p=q+2}^P P(CV_p|CV_{p-1}, CV_{p-2}, \dots, CV_{p-q}, c_i). \quad (4)$$

For a given Markov order  $q$ , there are  $q+1$  consecutive RVQ stages involved in the calculation of independent class condition probabilities. These  $q+1$  stages are shaded in Equation (4), above. The direct-sum codevectors formed out of these  $q+1$  stage codevectors are termed as *Markov direct-sum sub-codevectors*, and the corresponding indices are called *Markov sub-tuples*. The Markov order cannot be greater than  $P-1$ . The Bayesian classification rule, in Equation (2), together with the Markov condition, in Equation (4), is termed as Markov-Bayesian RVQ classifier.

A functional block diagram of RVQ-based classifier using the MAP rule, also termed as Bayesian RVQ classifier, is shown in Figure 8, below.

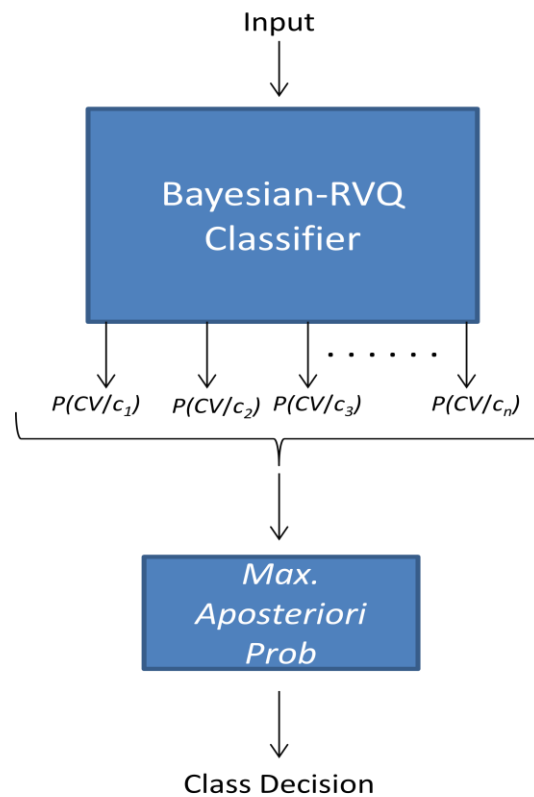
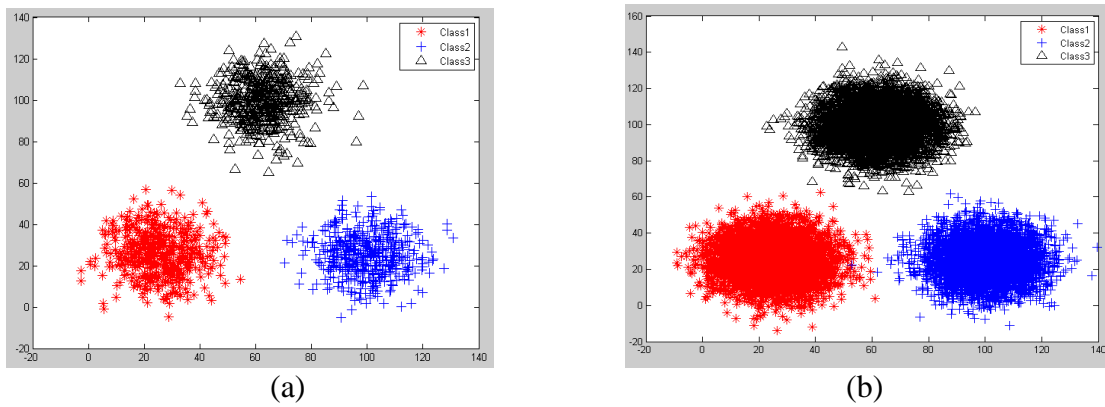


Figure 8. Bayesian RVQ classifier.

### Proof of Concept: Linearly Separable Synthetic Dataset

The algorithm for the Markov-Bayesian RVQ classification is tested and illustrated on a test dataset consisting of three classes; namely Class 1, Class 2, and Class 3, as shown in Figure 9. RVQ has  $M = 2$  codevectors-per-stage and  $P=8$  stages. Markov order is zero. In the zeroth-order Markov RVQ, the stage class decisions of RVQ are assumed to be independent of each other. The test and training datasets for each class are drawn from the same Gaussian distribution. The number of training data points for each class is 1000, and the number of test data points per class is 10000.



Stage index	Class 1		Class 2		Class 3		Test $P$ -tuple
	$CV^1$	$CV^2$	$CV^1$	$CV^2$	$CV^1$	$CV^2$	
$p=1$	<b>1</b>	0	<b>0.98</b>	0.02	<b>0.002</b>	0.998	1
$p=2$	0	<b>1</b>	1	<b>0</b>	0.006	<b>0.994</b>	2
$p=3$	<b>0.69</b>	0.31	<b>0.138</b>	0.862	<b>0.538</b>	0.462	1
$p=4$	<b>0.544</b>	0.456	<b>0.508</b>	0.492	<b>0.374</b>	0.626	1
$p=5$	0.46	<b>0.54</b>	0.494	<b>0.506</b>	0.496	<b>0.504</b>	2
$p=6$	0.524	<b>0.476</b>	0.556	<b>0.444</b>	0.542	<b>0.458</b>	2
$p=7$	<b>0.488</b>	0.512	<b>0.462</b>	0.538	<b>0.494</b>	0.506	1
$p=8$	0.532	<b>0.468</b>	0.532	<b>0.468</b>	0.51	<b>0.49</b>	2

$P(CV / \text{Class 1}) = 0.022$      
 $P(CV / \text{Class 2}) = 0.000$      
 $P(CV / \text{Class 3}) = 2.2 e^{-5}$

(c)

Figure 9. Synthetic dataset of three classes, (a) Training set. (b) Test set. (c) Class-conditional Transition Probability Matrix of the three classes.

It can be seen in Figure 10a that the training data points of the three classes are not overlapping. However, the test dataset of these classes is overlapping, as shown in Figure 10b. For each class, the class-conditional probabilities of the data points mapping to each codevector in the RVQ codebook are calculated by determining the frequency of mapping of the class-specific training data points to each codevector at every stage. It is to be noted that the RVQ stages are assumed to be independent; hence the Markov order is zero. The class-conditional probabilities are tabulated into a table called Class-conditional Transition Probability Matrix, as shown in Figure 10c, and in general the table is referred as transition probability matrix.

For the test  $P$ -tuple of an encoded test data point, shown on the right side of Figure 10c, the corresponding class-conditional probabilities in the transition probability matrix are highlighted for each class. As per the condition of the *zeroth* Markov order, the posterior probability for each class  $P(\mathbf{CV}_1, \mathbf{CV}_2, \mathbf{CV}_3, \dots, \mathbf{CV}_P | c_i)$  is calculated by multiplying together the highlighted individual class-conditional probabilities. As per Equation (2), the class decision is made by applying MAP on these class-conditional posterior probabilities. In this example, the MAP rule yields Class 1 as the class membership of the test input. The overall accuracy for this synthetic test dataset is 99%.

### **Proof of Concept: Linearly Non-Separable Synthetic Dataset**

In a series of experiments, Bayesian RVQ classifier is also tested on linearly non-separable synthetic data to see how it performs on a more complex dataset. Linearly non-separable *Swiss roll* dataset, as shown in Figure 10, 12, and 14, is formulated to test the classifier. In this dataset the two-dimensional data points belonging to different classes are arranged in concentric spirals. The experiments on these datasets are divided into two

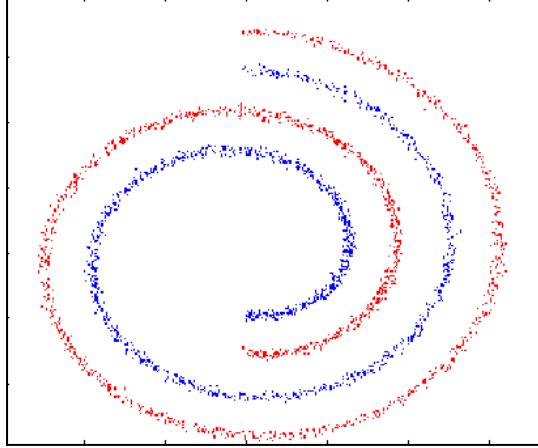


Figure 10. Synthetic Swiss roll training dataset of two classes to test and illustrate Bayesian RVQ classifier.

separate cases. In the first case, binary classification is performed on the dataset shown in Figure 10. The data of the two classes are marked Class 1 and Class 2 on the figure. It can be seen that the data from the two classes are spaced apart with no overlapping between each other. On the other hand in the second case, the Markov-Bayesian RVQ classifier is trained and tested on the Swiss roll dataset shown in Figure 12 and 14. The main difference between the datasets of the two cases is that in the second experiment the data from the multiple classes is not spaced apart as in the first case. Moreover, apart from the binary dataset (Figure 12) in the second case, the classifier is also tested on the Swiss roll dataset drawn from four classes, as shown in Figure 14. The value of these data points ranges between 0 and 255, and the number of training and test data points is 1600 data points for each class. These points are drawn from Gaussian distribution with means on the spirals of the Swiss roll. The two cases are discussed one-by-one as follows.

In the first case, RVQ codebook with  $M = 4$  and  $P = 8$  is designed and trained on the training data shown in Figure 10. The training is complete when all the class-



conditional probabilities in the Class-conditional Transition Probability Matrix are calculated for the *zereth* Markov order, as shown in Table 1. For the given training set, the distribution of the direct-sum RVQ codebook is shown in Figure 11a. The direct-sum codevectors the training points mapped to are shown in red and the rest of the direct-sum codevectors are marked in blue. The test dataset drawn for the two classes is shown in Figure 11b.

Table 1. (a) Class-conditional Transition Matrix with Markov order = 0. (b) Error matrix.

Stage	Class1	Class2
1	[0.25 ,0.27,0.20,0.28]	[0.28,0.28,0.19,0.26]
2	[0.12 ,0.31,0.28,0.29]	[0.16,0.31,0.26,0.27]
3	[0.28 ,0.06,0.38,0.29]	[0.23,0.29,0.23,0.25]
4	[0.18 ,0.28,0.24,0.30]	[0.22,0.21,0.31,0.26]
5	[0.31 ,0.18,0.24,0.26]	[0.24,0.25,0.27,0.23]
6	[0.001,0.35,0.29,0.4 ]	[0.02,0.34,0.31,0.34]
7	[0.39 ,0.23,0.00,0.38]	[0.36,0.29,0.00,0.36]
8	[0.002,0.34,0.26,0.4 ]	[0.04,0.29,0.36,0.31]

(a)

	Class 1	Class 2	
Class 1	1250	770	2020
Class 2	350	830	1180
	1600	1600	3200

(b)

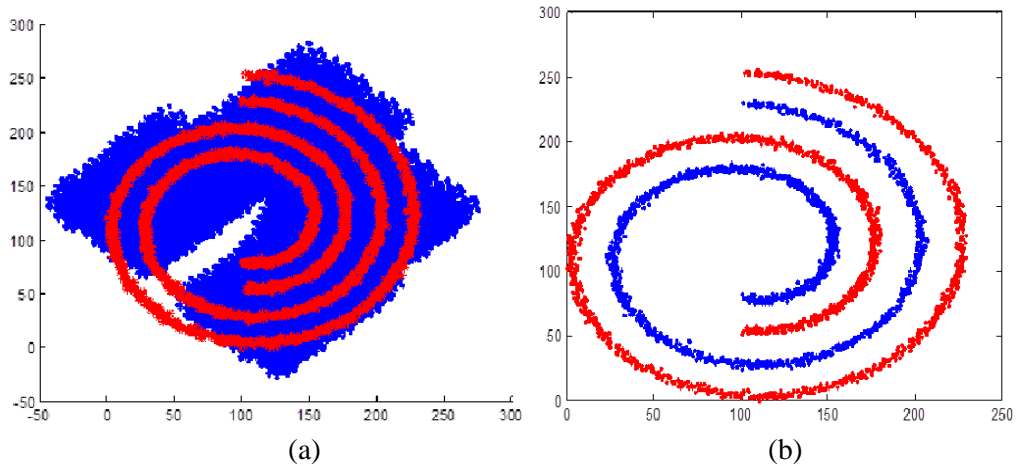


Figure 11. (a) Direct-sum codevector mapped by the training set shown in red, the remaining direct-sum codebook shown in blue. (b) Test data set. Class1 shown in blue, and Class 2 shown in red.

For the *zeroth* Markov order, the classification performance of the Bayesian RVQ classifier is shown in the error matrix [51] in Table 1b. Error matrix is also often called confusion matrix or a contingency table. To assess the accuracy of an image classification, it is common practice to create a confusion matrix. In a confusion matrix, the classification results are compared to additional ground truth information. The strength of a confusion matrix is that it identifies the nature of the classification errors, as well as their quantities. Performance of such systems is commonly evaluated using the data in the matrix. Commonly, the class names appear in the first column and the first in same order. The rest of the numerical entries in the matrix, except the entries in the last row and the last column are the classification results for each class. The diagonal classification results are the correctly classified test images, also called true-positive test images for the corresponding class. The true-positive entries in the matrix add up to the total number of test images. The performance measures that are calculated from the error matrix are user accuracy, producer's accuracy, and overall accuracy. User accuracy of the classification of a certain category is the ratio between the true-positive classification and the total number test images assigned to that class. Producer's accuracy for a certain class is the ratio between the true-positive for that class and the total number of actual test images from that class. Overall accuracy of classification is the total number or true-positive classification divided by the total number of test images. These three performance measures are commonly shown in percentages.

The user's, producer's, and overall accuracies for the test data shown in Figure 10 are calculated as follows:

$$\text{Class-1 User's Accuracy} = 1250/2020 = 61.88\%$$

Class-2 User's Accuracy	= 830/1180	= 70.34%
Class-1 Producer's Accuracy	= 1250/1600	= 78.12%
Class-2 Producer's Accuracy	= 830/1600	= 51.88%
Over all Accuracy	= (1250 + 830)/3200	= 65.00%

The same training and test datasets are also used to evaluate the classification performance of RVQ for Markov order equal to one. The corresponding Class-conditional Transition Probability Matrix and error matrix are shown in Table 2. The classification performance measures calculated from the error matrix are as follows:

Class-1 User's Accuracy	= 1243 /1833	= 67.81%
Class-2 User's Accuracy	= 1010 /1367	= 73.88%
Class-1 Producer's Accuracy	= 1243/1600	= 77.69%
Class-2 Producer's Accuracy	= 1010/1600	= 63.12%
Over all Accuracy	= (1243+1010)/3200	= 70.41%

From Table 2, it can be seen that at Stage-1 there is only one class-conditional probability associated to each codevector at this stage. Each probability is the probability of an input mapping to the respective codevector at Stage-1. However, from Stage-2 to Stage-8 four probabilities are associated to each codevector at a stage. The probability distribution in the table above is illustrated with the following example. For Class-1, Stage-1, Column-1 (shown in blue in Table 2); the first topmost probability is the probability of input mapping to Codevectors-1 and 1 of Stage-1 and Stage-2, respectively. Similarly, the second probability is the probability of the input that

Table 2. (a) Class-conditional Transition Matrix with Markov order = 1. (b) Error matrix.

Stages	Class1				Class2			
	[0.2533]	[0.2651]	[ 0.2034]	[ 0.2782]	[ 0.2777]	[ 0.2756]	[ 0.1863]	[0.2605]
1								
2	0.0486 0.0177 0.0039 0.0512	0.0505 0.0577 0.0919 0.1115	0.0735 0.0584 0.1037 0.0420	0.0807 0.1312 0.0039 0.0735	0 0.1017 0 0.0529	0.0660 0.0887 0.0825 0.0763	0.0639 0.0419 0.0330 0.1189	0.0880 0.0454 0.0976 0.0433
3	0.0696 0.1070 0.0413 0.0604	0 0.0433 0.0112 0.0046	0.0269 0.1024 0.1115 0.1365	0.0249 0.0591 0.1135 0.0879	0.0110 0.0529 0.0344 0.1326	0.0598 0.0770 0.0687 0.0866	0.0275 0.0570 0.1141 0.0302	0.0289 0.1010 0.0735 0.0447
4	0.0217 0.0020 0.0932 0.0636	0.0906 0.0157 0.1017 0.0735	0.0623 0.0203 0.0525 0.1063	0.1037 0.0210 0.1299 0.0420	0.0804 0.0076 0.0770 0.0515	0.0612 0.0131 0.0852 0.0495	0.0557 0.0137 0.1347 0.1072	0.0674 0.0275 0.0866 0.0818
5	0.0564 0.0873 0.0787 0.0912	0.0472 0.0289 0.0413 0.0617	0.0466 0.0781 0.0499 0.0696	0.0302 0.0873 0.0715 0.0741	0.0522 0.0515 0.0591 0.0790	0.0440 0.0866 0.0543 0.0687	0.0351 0.0859 0.0694 0.0818	0.0488 0.0460 0.0577 0.0797
6	0.001 0 0 0	0.0866 0.0308 0.1030 0.1299	0.1299 0.0571 0.0348 0.0715	0.0965 0.0912 0.1063 0.0617	0.0027 0.0096 0.0048 0.0007	0.0997 0.0619 0.0825 0.0921	0.1107 0.0515 0.0687 0.0742	0.1072 0.0605 0.0900 0.0832
7	0 0.1181 0.1483 0.1260	0.0007 0.1017 0.0374 0.0886	0 0 0 0	0 0.1306 0.1076 0.1411	0.0007 0.1395 0.0969 0.1216	0 0.0948 0.0845 0.1065	0 0 0 0	0 0.1175 0.1141 0.1237
8	0 0.0020 0 0	0.1529 0.0374 0 0.1437	0.1004 0.0735 0 0.0807	0.1391 0.1155 0 0.1549	0.0144 0.0076 0 0.0124	0.1230 0.0639 0 0.1065	0.1258 0.0955 0 0.1368	0.1271 0.0653 0 0.1216

(a)

	Class 1	Class 2	
Class 1	1243	590	1833
Class 2	357	1010	1367
	1600	1600	3200

(b)

maps to Codevectors-2 and 1 of Stage-1 and Stage-2, respectively.

Similarly, for Class-1, Stage-8, Column-1 (shown in green in Table 2); the first topmost probability is the probability of input mapping to Codevectors-1 and 1 of Stage-7 and Stage-8, respectively. The second probability is the probability of the input mapping to Codevectors-2 and 1 of Stage-7 and Stage-8, respectively.

For a given class, the probabilities across all the codevectors at a stage add to one. For examples, the probabilities shaded orange in the Transition Probability Matrix above add to one.

Moreover, the probabilities at stage 7 that are shaded red indicate that there is no mapping of any data, belonging to either of the two classes, from any codevector from Stage-6 to Codevector-3 of Stage-7.

As mentioned earlier, in the second case the Markov-Bayesian RVQ classifier is trained and tested on a Swiss roll dataset where the data drawn from multiple classes are not spaces apart. They overlap with each other to some degree, as shown in Figure 12 and Figure 14.

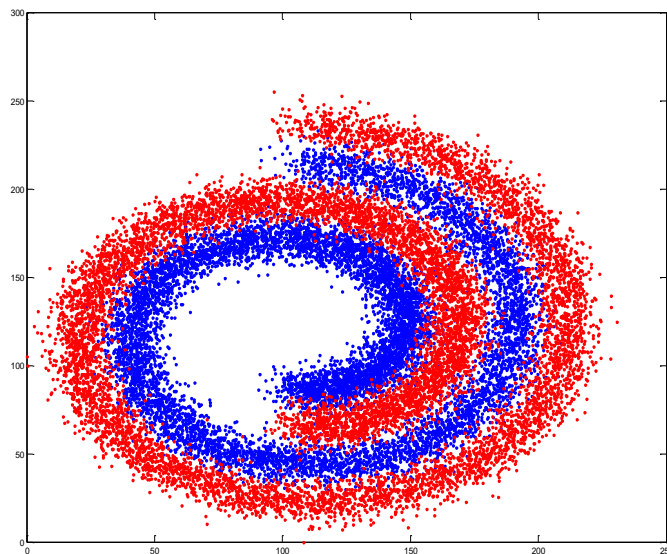


Figure 12. Training dataset: Class1 data is in blue, Class2 data is in red.

RVQ with  $M=4$  and  $P=8$  is trained on the dataset shown in Figure 12, and the corresponding Class-conditional Probability Matrix for Markov order ranging from 0 to  $P-1 = 7$  is also calculated. The data from the first class, namely Class-1, is shown in blue, the data from the second class, called Class-2, is shown red in the figure. The Markov-Bayesian RVQ classifier is tested on the binary data drawn for the same distribution as for the training data. The number of test data points is 1600 samples for each class. For this setup, the Overall and Producer's accuracies are plotted in Figure 13. The Markov order varies from 0 to  $P-1=7$ . The classification performance of this RVQ classifier is benchmarked against the performance of 1-Nearest-Neighbor (1-NN) classifier, also shown in Figure 13. In 1-NN classification, the class membership of a test data point is determined by the class membership of the nearest training data. Like in RVQ, Euclidean distance is the measure of nearness used in 1-NN. It can be seen that the performance of the Markov-Bayesian RVQ classifier begins to approach the 1-NN classifier performance from the *fourth* Markov order onwards.

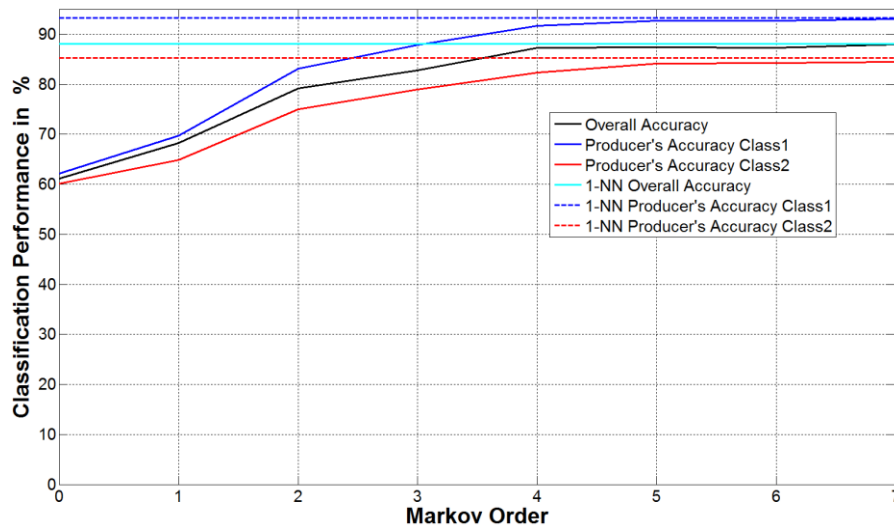


Figure 13. Classification performance curves for 2-category Swiss roll dataset.

In this experiment, it was noted that as the Markov order increased, test inputs started mapping to unused Markov direct-sum sub-codevectors. Equivalently, the test inputs were mapping to unused direct-sum codevectors. Figure 6, in Chapter 5, illustrates the distribution of a direct-sum RVQ codebook. The unused direct-sum codevectors after the training phase are shown in blue in the figure. An adverse implication of the test inputs mapping to unused is that the corresponding Voronoi regions are un-labeled. Therefore, MAP rule fails to yield a class-membership decision in such cases. To avoid this scenario, the encoding of a test input is subjected to the *realm-of-experience* (RoE) constraint. RoE constraint ensures that a test input is always assigned to the Markov direct-sum sub-codevector that is used by the training set. This assignment is made by searching over the nearest Markov direct-sum sub-codevector used by the training set. However, this RoE constraint adds an extra computational cost to the RVQ-based classification. If  $|T|$  is the size of the training set,  $k$  is the dimensionality of the input space, and  $O$  is the Markov order; then the computational cost by the RoE constraint is  $(P-O)k|T|$  in the worst case scenario, where the search for the nearest Markov direct-sum sub-codevector has to be carried out for all test Markov sub-tuples.

The Markov-Bayesian RVQ classifier is also tested on a Swiss-roll dataset comprising four classes. The training dataset is shown in Figure 14. Similar to the previous dataset, the data from the four classes, namely Class-1, Class-2, Class-3, and Class-4; are shown in blue, red, black, and yellow; respectively. The classification performance of the Markov-Bayesian RVQ classifier is plotted and benchmarked against 1-NN classifier in Figure 15. The classification performance is plotted for Markov order ranging from 1 to  $P-1 = 7$ . Again, it can be observed that the classification performance

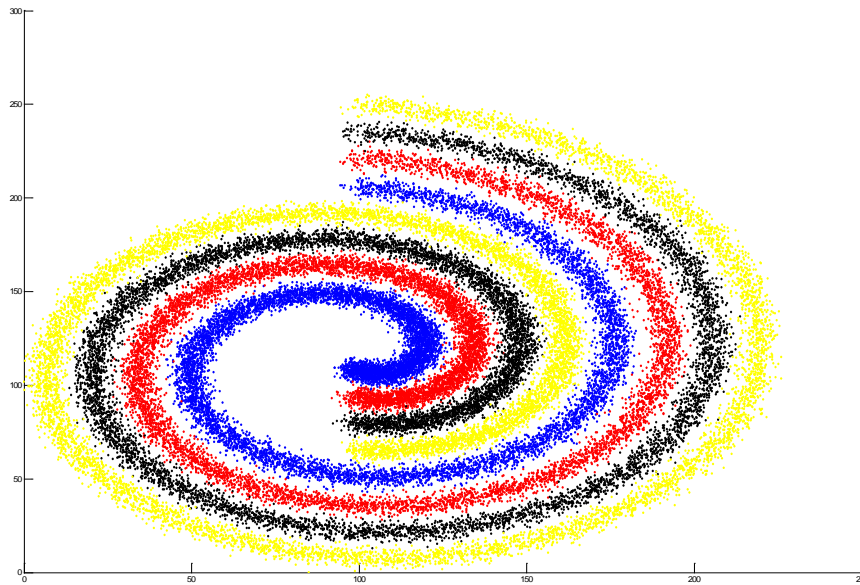


Figure 14. Training dataset for 4-category classification using Bayesian RVQ. Class-1 data is in blue, Class-2 data is in red, Class-3 is in black, and Class-4 is in yellow.

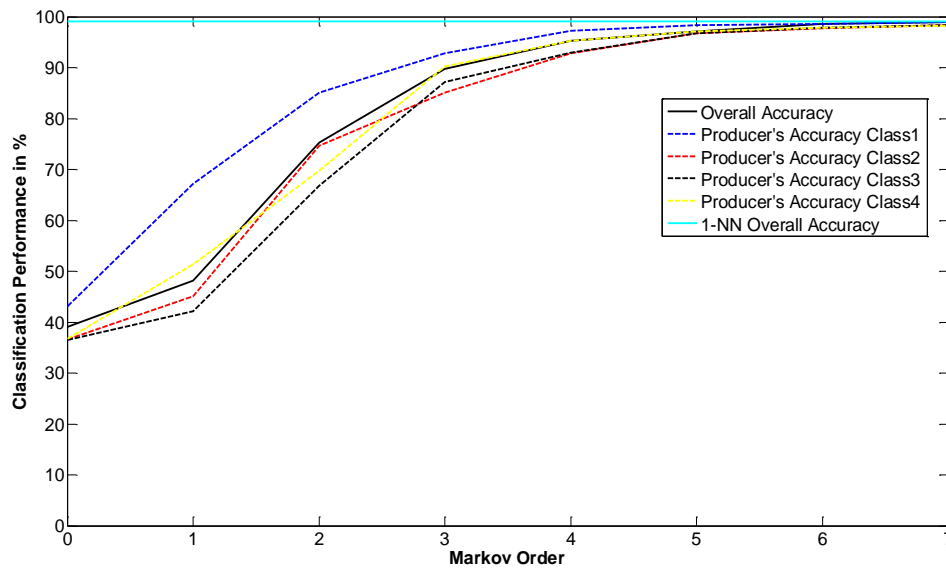


Figure 15. Classification performance curves for 4-category classification using Bayesian RVQ.

of the Bayesian RVQ classifier begins to converge to the performance of 1-NN classifier from *fourth* Markov order onwards.



### Proof of Concept: Image Dataset

After testing on synthetic data, image dataset is used to evaluate the performance of Markov-Bayesian RVQ classifier. Images of three different categories from Caltech101 image database [47] are used for the evaluation purpose. In this 3-category case, the classes are Plane, Car, and Motorbike. The size of the images in this database varies from approximately 150x350 pixels to 200x400 pixels. All the images are resized to 150x250 pixels for the RVQ classifier. The typical images from these data sets are shown in Figure 16. RVQ codebook is designed using the training dataset having 100 images for each class. In the test dataset, Plane, Car, and Motobike classes have 148, 87, and 256 images; respectively. To see the effect of varying number of codevectors-per-stage on the classification performance of Markov Bayesian RVQ classifier,  $M$  is varied

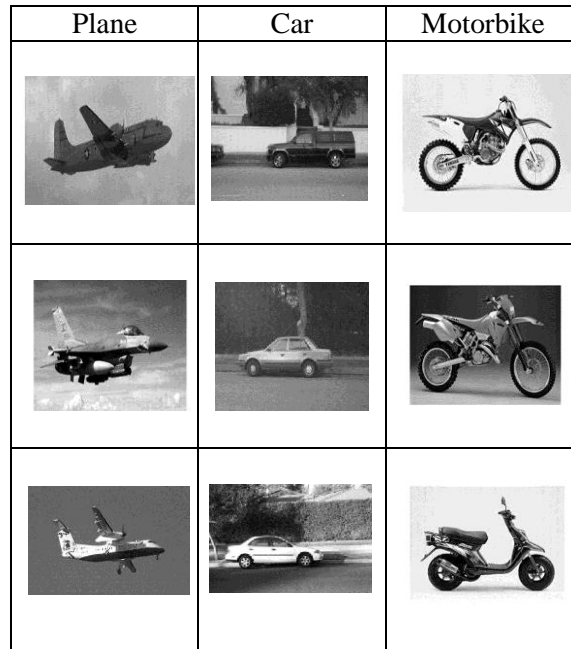


Figure 16. Training dataset for classification using Markov Bayesian RVQ with  $0^{th}$  Markov order.

from 3 to 5 codevectors-per-stage. The number of RVQ stages is  $P = 8$ , and the error matrices and Class-conditional Probability Transition Matrix for  $M = 3$ ,  $M = 4$ , and  $M = 5$  are shown in Table 3 and Table 4, respectively.

Table 3. Error matrices for (a)  $M=3$  and  $P=8$ , (b)  $M=4$  and  $P=8$ , (c)  $M=5$  and  $P=8$ ; Markov order = 0.

Class	Plane	Car	M'bike	
Plane	110	6	8	124
Car	34	81	1	116
M'bike	4	0	247	251
	148	87	256	491

(a)

Class	Plane	Car	M'bike	
Plane	126	15	4	145
Car	22	72	2	96
M'bike	0	0	250	250
	148	87	256	491

(b)

Class	Plane	Car	M'bike	
Plane	110	11	6	127
Car	30	76	0	106
M'bike	8	0	250	258
	148	87	256	491

(c)

The overall accuracy calculated from the error matrix for  $M = 3$  is 89.2%. It peaks to 91.2% for  $M=4$  before decreasing to 88.8% for  $M=5$ . It suggests that the RVQ classifier yields the best fit for  $M = 4$ , and begins to overfit for  $M = 5$  codevectors-per-stage.

More preliminary experiments are also conducted to gauge the classification performance of RVQ as a binary classifier. The Caltech 101 dataset, as shown in Figure 17, is used. For the purpose of binary classification, the following three separate RVQ classifiers were designed each with its own codebook:

Table 4. Class-conditional Probability Matrices for (a)  $M=3$  and  $P=8$ , (b)  $M=4$  and  $P=8$ , (c)  $M=5$  and  $P=8$ ; Markov order = 0.

Stage	Plane			Car			Motorbike		
	$CV^1$	$CV^2$	$CV^3$	$CV^1$	$CV^2$	$CV^3$	$CV^1$	$CV^2$	$CV^3$
$p=1$	0.06	0.5	0.44	0	1	0	0.91	0.03	0.06
$p=2$	0	0.74	0.26	0	0.361	0.639	0.22	0.38	0.4
$p=3$	0.46	0.53	0.01	0.639	0.028	0.33	0.32	0.55	0.13
$p=4$	0.26	0.29	0.45	0.33	0.472	0.194	0.04	0.35	0.61
$p=5$	0.3	0.18	0.52	0.139	0.417	0.44	0.35	0.38	0.27
$p=6$	0.3	0.18	0.52	0.33	0.417	0.25	0.29	0.38	0.33
$p=7$	0.3	0.24	0.46	0.33	0.278	0.389	0.27	0.27	0.46
$p=8$	0.29	0.37	0.34	0.167	0.25	0.583	0.09	0.49	0.42

(a)

Stage	Plane				Car				Motorbike			
	$CV^1$	$CV^2$	$CV^3$	$CV^4$	$CV^1$	$CV^2$	$CV^3$	$CV^4$	$CV^1$	$CV^2$	$CV^3$	$CV^4$
$p=1$	0.08	0.27	0	0.65	0	1	0	0	0.86	0.03	0.1	0.01
$p=2$	0.15	0.17	0.55	0.13	0	0.44	0.361	0.194	0.16	0.36	0.18	0.3
$p=3$	0	0.3	0.3	0.4	0.139	0.167	0.22	0.472	0	0.4	0.29	0.31
$p=4$	0.23	0.02	0.31	0.44	0.139	0.22	0.33	0.306	0.17	0.18	0.34	0.31
$p=5$	0.19	0.11	0.42	0.28	0.389	0.361	0.083	0.167	0.08	0.31	0.12	0.49
$p=6$	0.24	0.26	0.29	0.21	0.22	0.25	0.167	0.361	0.17	0.13	0.27	0.43
$p=7$	0.16	0.26	0.37	0.21	0.22	0.167	0.194	0.417	0.26	0.18	0.25	0.31
$p=8$	0.14	0.29	0.16	0.41	0.167	0.25	0.278	0.306	0.31	0.17	0.24	0.28

(b)

Stage	Plane					Car					Motorbike				
	$CV^1$	$CV^2$	$CV^3$	$CV^4$	$CV^5$	$CV^1$	$CV^2$	$CV^3$	$CV^4$	$CV^5$	$CV^1$	$CV^2$	$CV^3$	$CV^4$	$CV^5$
$p=1$	0.08	0.31	0	0	0.61	0	0.583	0	0.417	0	0.86	0.02	0.1	0	0.02
$p=2$	0.06	0.06	0.06	0.49	0.33	0	0.33	0.22	0.056	0.389	0.16	0.17	0.19	0.05	0.43
$p=3$	0.16	0.14	0.25	0.16	0.29	0.028	0.167	0.25	0.33	0.22	0.04	0.2	0.22	0.23	0.31
$p=4$	0.18	0.12	0.34	0.14	0.22	0.278	0.25	0.167	0.167	0.139	0.12	0.25	0.34	0.18	0.11
$p=5$	0.24	0.06	0.16	0.22	0.32	0.25	0.139	0.194	0.194	0.22	0.25	0.18	0.15	0.24	0.18
$p=6$	0.17	0.14	0.11	0.35	0.23	0.22	0.25	0.194	0.22	0.11	0.21	0.19	0.09	0.23	0.28
$p=7$	0.06	0.18	0.15	0.31	0.3	0.083	0.22	0.11	0.25	0.33	0.02	0.19	0.22	0.27	0.3
$p=8$	0.19	0.25	0.18	0.13	0.25	0.25	0.083	0.139	0.306	0.22	0.15	0.26	0.12	0.18	0.29

(c)

- (a) Plane-vs-Rest.
- (b) Car-vs-Rest.
- (c) Motorbike-vs-Rest.

Support vector machines (SVM) based classification was also done using the same training and test datasets to compare its classification performance against the binary and multi-class Markov-Bayesian RVQ classifiers. 1-vs-rest scheme was used to implement the SVM classifier with the quadratic kernel. In Table 5, the producer accuracies for each classifier are shown as their classification performance measures.

It is observed in the preliminary results, shown in Table 5, that overall multi-class Markov-Bayesian RVQ classifier performs better than its binary version. On the average, the performance of multi-class MBRVQ is approximately 5% better than the binary-class MBRVQ. Furthermore, when compared with the SVM classifier, the proposed multi-class RVQ classifier performance is promising, even at a Markov order as low as zero. The performance of SVM classifier is better than the multi-class MBRVQ by only 4%, approximately. It is expected that by increasing the Markov order, the proposed multi-class RVQ classifier will improve in its performance.

### **Conclusion of Preliminary Research**

For effective RVQ classification it is imperative that the training dataset is chosen so that it is a good representation of the source, and the direct-sum codevectors of the RVQ provide a dense covering, which will ensure that a test input is quantized with a low reconstruction error. However, with a dense covering we are faced with the challenges of high memory and computation costs for performing the classification task. We will discuss these challenges in more detail in the following paragraphs.

Table 5. RVQ-based classifiers-vs-SVM-based classifier.

RVQ Settings	Classes	Binary Classification		Multi-Class RVQ
		Binary RVQ	SVM Quadratic	
M= 3, P=8	Plane	76.87	70.19	88.71
	Car	65.87	100	70
	M'bike	98.43	99.59	98.41
M= 4, P=8	Plane	81.2	70.19	86.9
	Car	66.67	100	75
	M'bike	99.6	99.59	100
M= 5, P=8	Plane	74.14	70.19	86.6
	Car	70.25	100	71.7
	M'bike	98.43	99.59	96.9

With RVQ providing a dense covering of the input space, a classification operation will require to label all the regions associated with dense covering of the RVQ. This scenario can pose serious problems when for the values of  $M$  (number of codevectors per stage) and  $P$  (number of stages) the number of regions-to-be-labeled is very high i.e.,  $M^P$ . As discussed earlier, to address the cost issue, a Markov Bayesian structure on the stages of the RVQ can be imposed, where the aim is to combine class-membership decisions stage-wise. As a result, the cost of the classification is expected to reduce from  $M^P$  to the order of  $MP$ . The Markovian order determines the number of stages in the RVQ that form a dependent neighborhood. The higher the Markovian order, the higher is the cost of the classification. The results of the preliminary experiments on the Swiss roll datasets, Figure 10, 12, 14; and Caltech101 dataset, Figure 16, also indicate the efficacy of the Markov structure on the Bayesian RVQ classifier.

However, before further experiments are carried to explore the feasibility of Markov Bayesian RVQ classifier, an analysis is made to explore whether the similar  $P$ -tuples of the encoded data points also correspond to similar data points. If this is so, then RVQ-based classification can become feasible in the  $P$ -tuple space, which will result in reducing significantly the overall operational cost of the proposed RVQ-based classifier.

### **RVQ Classification Performance Benchmark**

The classification performed by jointly considering all the  $P$  stages of the RVQ is the benchmark classification performance of the RVQ classifier. Therefore, a simple joint  $P$ -stage RVQ classifier is implemented. For all the classification experiments, the 2D Swiss Roll dataset is used in which two datasets corresponding to two different classes are mixed together in the Swill Roll format, as shown in Figure 12.

#### **Joint $P$ -Stage RVQ Classifier**

The joint  $P$ -stage, also referred to as full  $P$ -tuple, classification is implemented in the following ways.

##### Full $P$ -tuple matching-based Classification

In this method, the full  $P$ -tuples of test inputs are matched to the full  $P$ -tuples of the training inputs using two different distance criteria: Euclidean and Hamming distance criteria. Following are the results for the two criteria:

##### Euclidean Distance Criterion

	Class 1	Class 2	
Class 1	1542	311	1853
Class 2	58	1289	1347
	1600	1600	3200

$$\text{Over all Accuracy} = (1542+1289)/3200 = 88.5\%$$

### Hamming Distance Criterion

	Class 1	Class 2	
Class 1	1581	388	1969
Class 2	19	1212	1231
	1600	1600	3200

$$\text{Over all Accuracy} = (1542+1289)/3200 = 87.3\%$$

### Direct-sum Codevector matching-based Classification

In this method, the decoded test inputs are matched according to 1-NN rule to the decoded training set. The matching between the decoded test input and the training set is done using the Euclidean distance criterion. Following is the result for this method:

### Euclidean Distance Criterion

	Class 1	Class 2	
Class 1	1600	0	1600
Class 2	0	1600	1600
	1600	1600	3200

$$\text{Over all Accuracy} = (1600+1600)/3200 = 100\%$$

### **Conclusion**

From the results above, it is concluded that RVQ  $P$ -tuple labeling using a Euclidean or Hamming distance metric do not correspond to Euclidean distance using values of the decoded inputs. It can be generalized that in images similar  $P$ -tuples are not guaranteed to map to similar images.

### **Estimated Markov-Bayesian RVQ Costs**


The cost of implementing Markov Bayesian RVQ classifier is dependent on the Markov order  $O$ , number of RVQ stages  $P$ , number of codevectors-per-stage  $M$ , and the number of classes  $C$  in the training dataset. It is desired that the Markov order is as low as

possible, somewhere in the middle of  $O^{th}$  and  $P^{th}$  Markov order, so that the associated costs are also kept low.

For given values of  $M$ ,  $P$ , and  $C$ , the costs of implementation for Markov order  $O$  from zero to  $P$  are given in Table 6. It can be seen in the table that the memory for storing the codebook and the cost of the search through the codebook remain  $kMP$  for any value of Markov order. However, the memory cost of storing the Class-conditional Transition Probability Matrix increases in the order of  $M^{O+1}C$ . The equation that expresses the relationship between Markov order and the memory cost of storing the probabilities is  $(P-O) M^{O+1}C$ . It is to be noted that the search cost of 1-NN classifier is  $|T|$ , where  $|T|$  is the size of the training set. Moreover, the memory cost associated with 1-NN classification is in the order of  $|T|$ . It is expected that the RVQ-based classifier will offer cost savings over the 1-NN classifier.

Table 6. Implementation cost of RVQ classifier.

Markov Order	Memory Cost		Search Cost
	Codebook	Probabilities	
$0^{th}$	$MP$	$8MC$	$MP$
$1^{st}$	$MP$	$7M^2C$	$MP$
$2^{nd}$	$MP$	$6M^3C$	$MP$
$3^{rd}$	$MP$	$5M^4C$	$MP$
$4^{th}$	$MP$	$4M^5C$	$MP$
$5^{th}$	$MP$	$3M^6C$	$MP$
$6^{th}$	$MP$	$2M^7C$	$MP$
$7^{th}$	$MP$	$M^8C$	$MP$

  
 $(P-O) M^{O+1}C$



## CHAPTER 6

# MARKOV-BAYESIAN RESIDUAL VECTOR QUANTIZATION- BASED CLASSIFICATION: MAIN RESEARCH

### Introduction

The number of codevectors  $M$  and stages  $P$  of RVQ are essential RVQ parameters that control the density of the covering of the input space. Equivalently,  $M$  and  $P$  control the size of the codebook of RVQ. Since the number of Voronoi regions generated in the input space is directly related to these two RVQ parameters, the classification performance of RVQ will be investigated for different values of  $M$  and  $P$ .

The focus of the preliminary research was to propose a method to make RVQ-based classification feasible by imposing a Markov structure on the stages of RVQ. It was also noted that classification performance showed improvement when Markov order was increased from zero to one. With the Markov structure, the operational cost of RVQ-based classifier can be reduced from  $M^P$  to the order of  $MP$ . However, with increase the Markov order, the cost of RVQ classification also increases. Therefore, the effect of different Markov orders on RVQ-based classification is also explored in-depth to analyze the underlying issues.

Lastly, since RVQ is a template-matching-based technique, the characteristics of a dataset will heavily bear on the performance of the RVQ-based classifier. Therefore, datasets with differing characteristics will be investigated for RVQ classification. Caltech101 [47], Graz [48], and the MNIST database of handwritten digits [50] used to test Markov RVQ classifier for Markov order ranging from zero to  $P-1$ . The *zeroth*

Markov order means that all the RVQ stages are assumed independent of each other. Whereas, Markov order  $P-1$  implies that all the  $P$  stages of RVQ are assumed dependent on each other. The datasets vary from each other in the amount of variability in the images.

In short, RVQ –based classification is investigated for the following three factors:

- (a) Different values of  $M$  codevectors-per-stage of RVQ.
- (b) Varying Markov order on the stages of RVQ.
- (c) Image datasets with different characteristics.

The classification results obtained from this investigation provides insightful analysis into the working of RVQ as a classifier, and it guides the research to understand the parameters needed to extract improved classification results out of RVQ.

The proposed RVQ classification is also compared to SVM-based classification involving feature vectors consisting of image intensity levels, and scale invariant feature transform (SIFT) [3]. It is shown how the proposed RVQ classifier fares with SIFT feature vectors.

### **Effects of Varying values of $M$**

For the Caltech101 dataset used in the preliminary research, Markov Bayesian RVQ classifier is investigated for varying number of codevectors-per-stage  $M$ . It is reminded that the training and test data are the same as used in the preliminary research, as shown earlier in Figure 17. The categories are Plane, Car, and Motorbike, and Markov order for classification varies from 0 to  $P-1 = 7$ . The number of RVQ stages  $P$  is 8, and the number of codevectors-per-stage is varied from  $M = 2$  to  $M = 11$ . The chosen range of  $M$  is enough to see the trend in the performance of RVQ classification. The RVQ

codebook with  $M=4$  and  $P = 8$ , and trained on the dataset shown in Figure 16, is shown in Figure 17. The error matrices for the different values of  $M$  and a Markov order of 0 are shown in Table 7. The overall classification accuracy for  $M = \{2,3,4,5,\dots,11\}$  and the values of Markov order ranging from 0 to  $P-1 = 7$  is plotted in Figure 18.

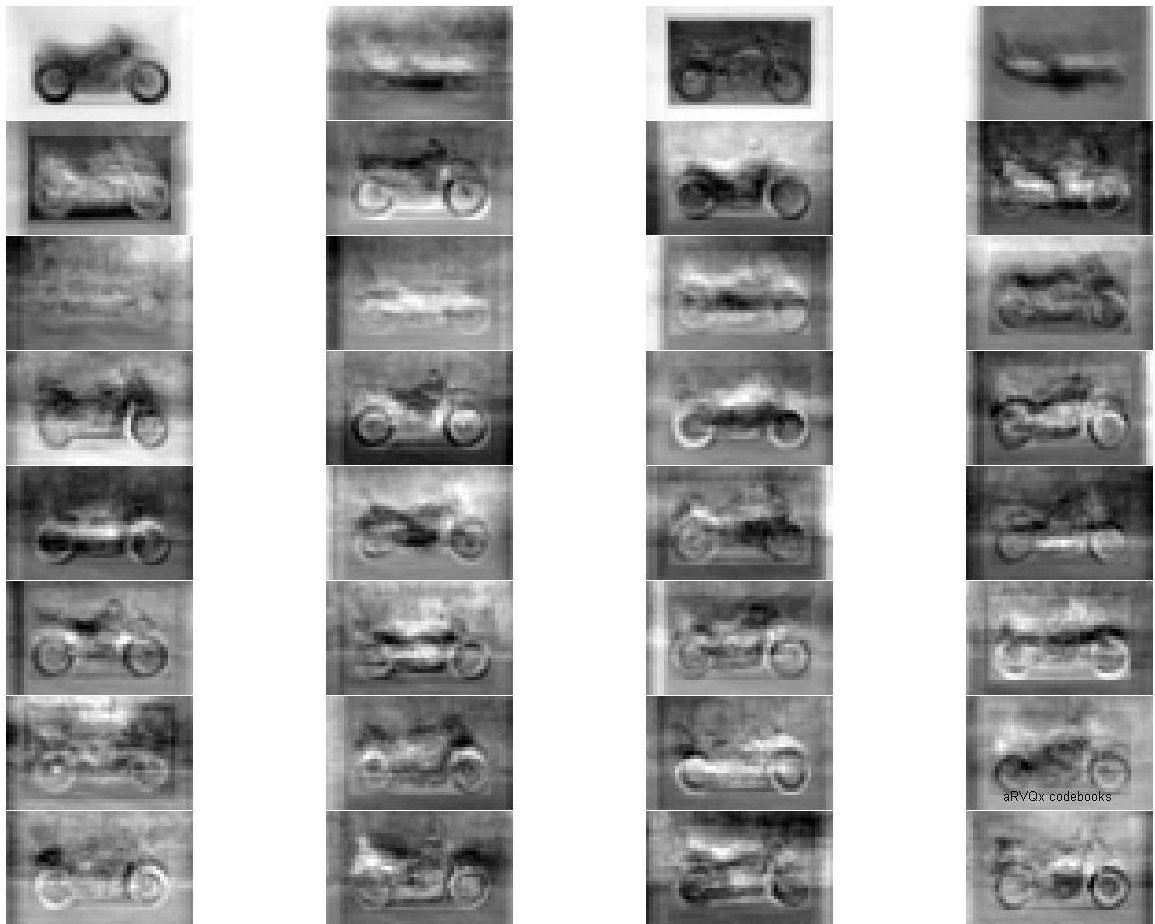


Figure 17. RVQ codebook for  $M = 4$ ,  $P = 8$ . 3-category training set comprises Plane, Car, and Motorbike classes.

Table 7. Error matrices for  $M = 2$  to  $M = 11$ . RVQ has  $P=8$  stages with zeroth Markov order for classification.

$M = 2$					$M = 3$					$M = 4$				
Classes	Plane	Car	M'bike		Classes	Plane	Car	M'bike		Classes	Plane	Car	M'bike	
Plane	110	16	6	132	Plane	132	9	3	144	Plane	122	3	30	155
Car	18	61	2	81	Car	12	78	2	92	Car	21	84	1	106
M'bike	20	10	248	278	M'bike	4	0	251	255	M'bike	5	0	225	230
	148	87	256	491		148	87	256	491		148	87	256	491
$M = 5$					$M = 6$					$M = 7$				
Classes	Plane	Car	M'bike		Classes	Plane	Car	M'bike		Classes	Plane	Car	M'bike	
Plane	131	5	31	167	Plane	124	1	8	133	Plane	139	5	4	148
Car	8	82	2	92	Car	19	85	1	105	Car	7	82	2	91
M'bike	9	0	223	232	M'bike	5	1	247	253	M'bike	2	0	250	252
	148	87	256	491		148	87	256	491		148	87	256	491
$M = 8$					$M = 9$					$M = 10$				
Classes	Plane	Car	M'bike		Classes	Plane	Car	M'bike		Classes	Plane	Car	M'bike	
Plane	136	3	10	149	Plane	128	3	4	135	Plane	134	0	18	152
Car	10	84	1	95	Car	16	84	1	101	Car	9	87	1	97
M'bike	2	0	245	247	M'bike	4	0	251	255	M'bike	5	0	237	242
	148	87	256	491		148	87	256	491		148	87	256	491

It can be seen in Figure 18 that for the given 3-category dataset, the classification performance tends to improve until  $M = 7$ , after which it decreases. This trend suggests that Markov Bayesian RVQ classifier begins to over-fit after  $M = 7$ . It is also noted that for all the values of  $M$  the classification performance begins to converge to the best classification performance of MBRVQ classifier from the 3<sup>rd</sup> Markov order onwards. It is reminded that the value of Markov order is also very critical to the success of the RVQ-based classifier. The higher the Markov order, the higher the memory cost of the RVQ classifier. In this experiment, it is further observed that as the Markov order increased, test inputs started mapping to unused Markov direct-sum sub-codevectors. Equivalently, the test inputs were mapping to unused direct-sum codevecotors. Figure 6, in Chapter 5, illustrates the distribution of a direct-sum RVQ codebook. The unused direct-sum codevectors after the training phase are shown in blue in the figure. An adverse

implication of the test inputs mapping to unused is that the corresponding Voronoi regions are un-labeled. Therefore, MAP rule fails to yield a class-membership decision in such cases. To avoid this scenario, the encoding of a test input is subjected to the *realm-of-experience* (RoE) constraint. RoE constraint ensures that a test input is always assigned to the Markov direct-sum sub-codevectors that are used by the training set. This assignment is made by searching over the nearest Markov direct-sum sub-codevector used by the training set. However, this RoE constraint adds an extra search cost to the RVQ-based classification.

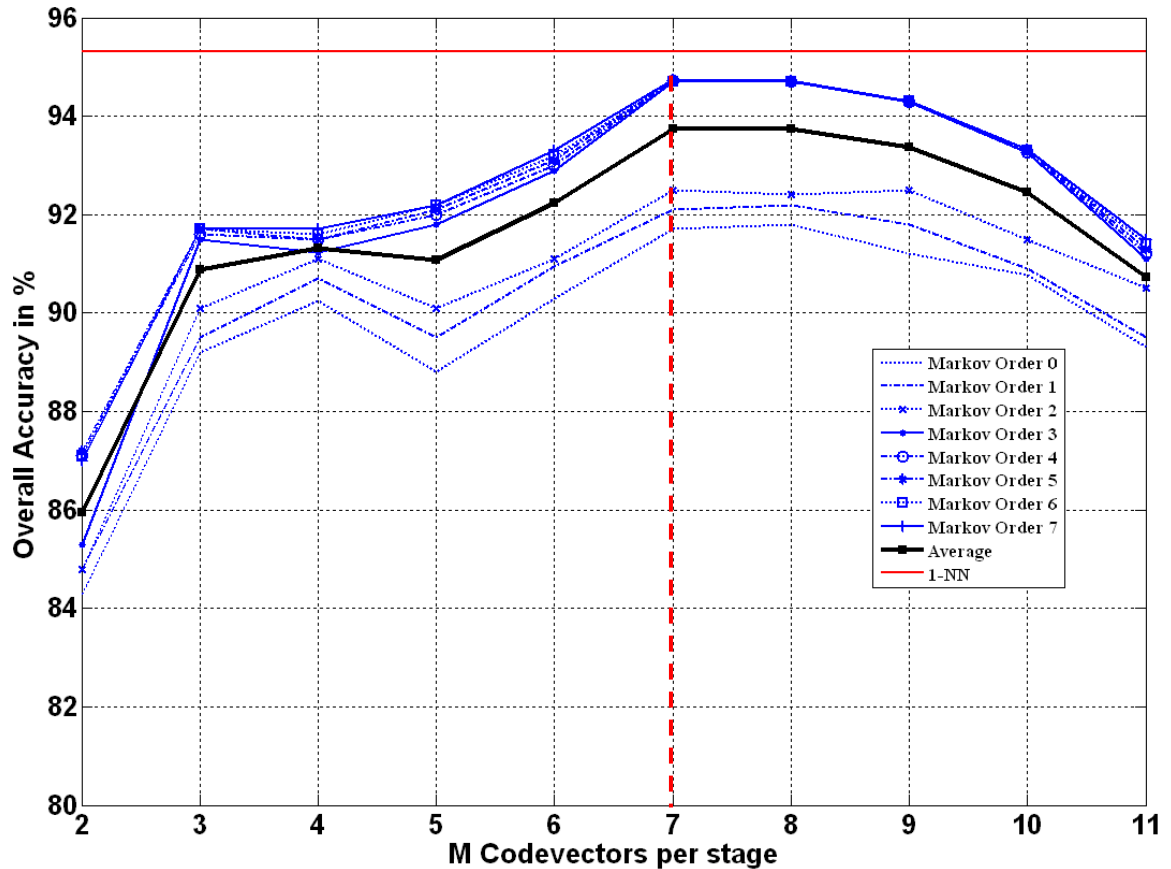


Figure 18. RVQ classification performance for  $M = \{2,3,4,5,6,7,8,9,10,11\}$ , and  $P=8$ , and for Markov orders from 0 to  $P-1 = 7$ .

Since in the figure above the performance of the RVQ-based classifier peaks at  $M=7$ , its classification performance at  $M = 7$  is also calculated over a range of Markov

order from 0 to  $P-1 = 7$ , where  $P = 8$  stages. The classification performance in terms of the overall accuracy is shown in Figure 19, along with the error matrix and the related classification performance measures derived from the error matrix shown in Table 8. The performance of the RVQ-based classifier is also compared with 1-NN classifier, also shown in Figure 19. It can also be seen that the RVQ classifier converges to 1-NN from the *third* Markov order onwards. Expectedly, the classification performance of MBRVQ classifier improves with the increase in the Markov order. It is desired that the Markov order is as low as possible.

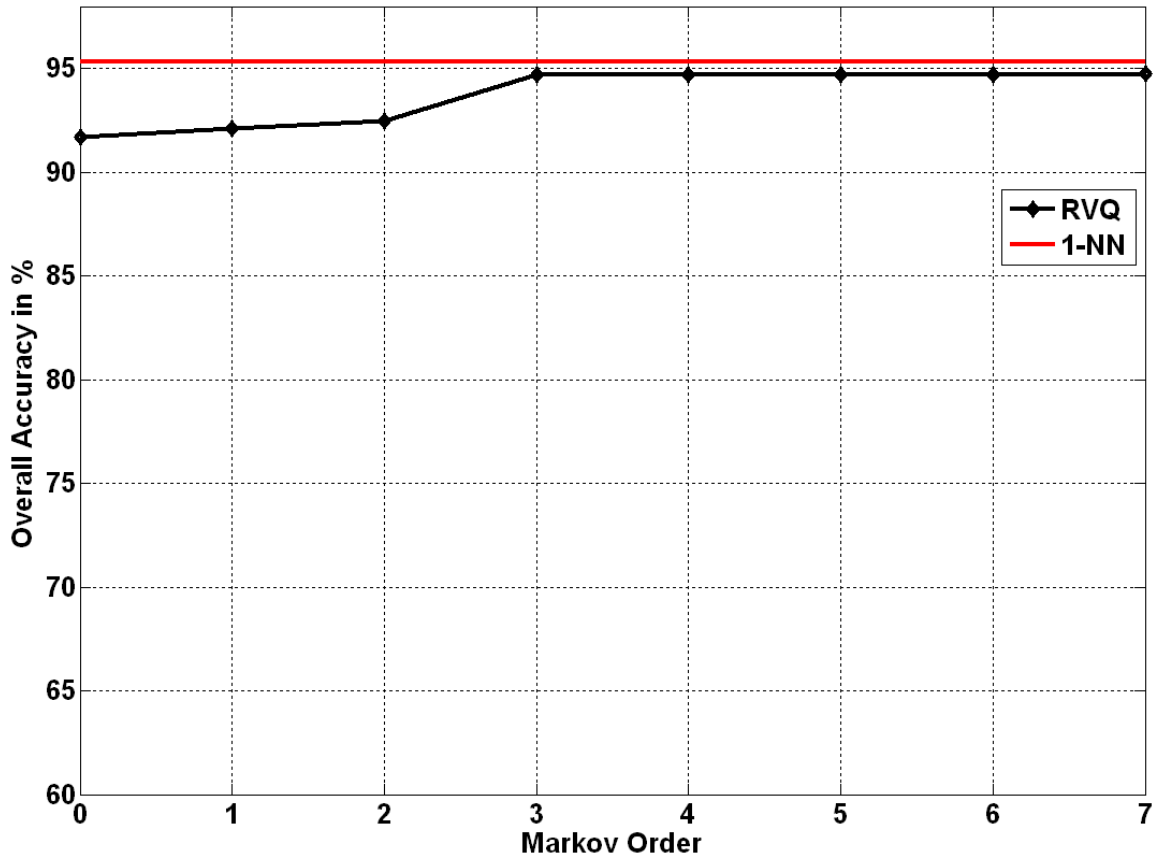


Figure 19. Classification performance of 1-NN based classifier versus Markov Bayesian RVQ classifier with  $M = 7$ ,  $P = 8$ , and Markov Order = 4.

Table 8. Classification performance of 1-NN based classifier versus Markov Bayesian RVQ classifier with  $M = 7$ ,  $P = 8$ , and Markov Order = 4.

<b>Error Matrix</b>					<b>1-NN Classifier</b>				
<b>RVQ, <math>M=7</math>, <math>P=8</math>, and Markov Order = 4</b>									
<b>Classes</b>	Plane	Car	M'bike		<b>Classes</b>	Plane	Car	M'bike	
Plane	136	3	10	149	Plane	148	11	12	171
Car	10	84	1	95	Car	0	76	0	76
M'bike	2	0	245	247	M'bike	0	0	244	244
	148	87	256	491		148	87	256	491
	<b>Producer's Accuracy</b>		<b>User's Accuracy</b>			<b>Producer's Accuracy</b>		<b>User's Accuracy</b>	
Plane	136/149 = 91.3%		136/148 = 91.9%		Plane	148/171 = 86.6%		148/148 = 100 %	
Car	84/95 = 88.4%		84/87 = 96.6%		Car	76/76 = 100 %		76/87 = 87.4%	
M'bike	245/247 = 99.2%		245/256 = 95.7%		M'bike	244/244 = 100 %		244/256 = 95.3%	
<b>Overall Accuracy</b>					<b>Overall Accuracy</b>				
(136+84+245)/491					(148+76+244)/491				
=					=				
94.71%					95.32%				

### RVQ Classification Schemes

As mentioned earlier, the Markov order plays a critical role in the RVQ-based classification. The higher the Markov order, the higher the overall cost of implementing the RVQ-based classifier. Up to now, RVQ-based classification with the Bayesian framework has been discussed. At this point, another framework, called *feature-count* rule is introduced as a different method to implement the RVQ-based classification. It is emphasized for clarity, that feature-count rule is different from the earlier proposed Markov Bayesian RVQ (MBRVQ) classification. While both the methods impose the Markov structure on its RVQ stages, the Bayesian framework is not employed in feature-count rule. The implementation of these two methods is explained next.

The description of the two different schemes along with their implementation details are explained as follows:

**Feature-count Rule:** The Markov direct-sum sub-codevectors of the RVQ can be viewed as clustering the data in a causal-anti-causal (CAC) residual sub-space. Intuitively, a cluster in a CAC residual sub-space can be thought of as a feature of the class. For a given test input, these features can be counted and attributed to a class according to the class-conditional probability of the CAC cluster or the corresponding Markov sub-tuple.

**Markov Bayesian RVQ (MBRVQ):** This method is not different from the Bayesian RVQ classification method as described before.

**CostERoE:** As mentioned earlier, the RoE constraint incurs an additional search cost. Therefore, a cost-effective method for implementing the RoE constraint is tested. This new method is termed as CostERoE, whereas, the earlier method will be referred to as RoE in this report. CostERoE method of implanting the realm-of-experience constraint is used by Barnes in [22]. CostERoE ensures that the encoding of a test input stops at the stage when the input maps to the direct-sum codevector not used by the training set. Consequently, the situation where a test input maps to an un-labeled Voronoi region is avoided. The advantage of implementing the CostERoE constraint over the RoE constraint is that the former adds no further computational cost to the RVQ-based classification. However, it adds  $|T|$  bytes to the overall memory cost of the RVQ-based classification method, where  $|T|$  is the size of the training set.

**Thresholds:** Different thresholds  $T_h$  on the class-conditional probabilities associated with the Markov sub-tuples will be applied to see their effects on the classification performance of the two schemes. Intuitively, the threshold  $T_h$  can be thought as a means to weight CAC-clusters in reaching a classification decision. Therefore, with a suitable



$T_h$ , only those CAC clusters can be isolated those contribute most significantly towards reaching the class-membership decision.

The two schemes for RVQ-based classification i.e., Feature-count Rule and MBRVQ, will be tested on Caltech101 [47] and Graz [48] image databases. These experiments will help to understand the dynamics of RVQ-based classifier. Two different settings of RVQ are used:  $M=4$  and  $P=8$ , and  $M=2$  and  $P=16$ . The two RVQ settings generate the same size of the direct sum codebook, thus have the same number of degrees of freedom (DoF). However, the degrees of freedom at the RVQ stage level is different i.e., stage DoF for  $M = 2$  is two, and stage DoF for  $M = 4$  is four.

These tests serve as a guide to understanding the RVQ-based classification to determine how best to use RVQ for classification, and what are the suitable conditions and datasets to use RVQ as a classifier.

## **Experiments and Results**

### **Experiments: Set 1**

In this first set of experiments, the Markov order is kept to zero to establish a performance base-line. The first data set that is used is a 3-category dataset consisting of the classes Plane, Car, and Motorbike from Caltech101 database [47]. Each class has hundred training images, the training set consists of 148, 87, and 256 images from the classes Plane, Car, and Motorbike ; respectively. They typical images of this dataset are shown in Figure 16. The RVQ codebook, Class-conditional Probability Matrix, and the error matrix for RVQ with  $M=2$  and  $P=16$  are shown in Figure 20, Table 9, and Table 10, respectively. Whereas, Figure 21, Table 11, and Table 12 show the Class-conditional

Transition Probability Matrix, and the error matrix for RVQ with  $M=4$  and  $P=8$ , respectively.

The RVQ-based classification is carried on this dataset with the following four techniques:

- (a) MBRVQ with RoE constraint.
- (b) Feature-count Rule with RoE constraint.
- (c) MBRVQ with CostERoE constraint.
- (d) Feature-count Rule with CostERoE constraint.

The classification performance of the above four methods for RVQ with  $M = 2$  and  $P = 16$  are tabulate in the error matrix, shown in Table 10, It can be seen that the MBRVQ-based methods are far superior to Feature-count Rule-bases methods.



Figure 20. RVQ codebook for RVQ with  $M=2$ ,  $P=16$ . The dataset consists of Plane, Car, and Motorbike classes.

Table 9. Class-conditional Probability Matrix for RVQ with  $M = 2$  and  $P = 16$ . Class 1, 2, 3 are Plane, Car, Motorbike, respectively.

Stage	Class	$CV_1$	$CV_2$	Stage	Class	$CV_1$	$CV_2$
<b>1</b>	1	0.10	0.45	<b>9</b>	1	0.36	0.33
	2	0.00	0.50		2	0.34	0.33
	3	0.90	0.05		3	0.30	0.34
<b>2</b>	1	0.91	0.16	<b>10</b>	1	0.33	0.33
	2	0.00	0.43		2	0.28	0.37
	3	0.09	0.41		3	0.39	0.29
<b>3</b>	1	0.00	0.36	<b>11</b>	1	0.38	0.30
	2	0.12	0.35		2	0.33	0.34
	3	0.88	0.28		3	0.29	0.36
<b>4</b>	1	0.46	0.19	<b>12</b>	1	0.41	0.29
	2	0.36	0.31		2	0.28	0.37
	3	0.19	0.5		3	0.31	0.35
<b>5</b>	1	0.46	0.13	<b>13</b>	1	0.32	0.35
	2	0.26	0.45		2	0.35	0.32
	3	0.28	0.42		3	0.33	0.34
<b>6</b>	1	0.31	0.35	<b>14</b>	1	0.28	0.38
	2	0.38	0.31		2	0.35	0.32
	3	0.32	0.34		3	0.38	0.30
<b>7</b>	1	0.26	0.38	<b>15</b>	1	0.24	0.41
	2	0.44	0.27		2	0.38	0.30
	3	0.30	0.35		3	0.38	0.29
<b>8</b>	1	0.29	0.37	<b>16</b>	1	0.35	0.32
	2	0.35	0.32		2	0.35	0.32
	3	0.36	0.31		3	0.31	0.35

Table 10. Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has  $M=2$  codevectors-per-stage and  $P = 16$  stages.

Classes	MBRVQ, RoE				Feature-count Rule, RoE				MBRVQ, CostERoE				Feature-count, CostERoE			
	1	2	3		1	2	3		1	2	3		1	2	3	
1	97	4	1	102	142	65	154	361	97	4	1	102	134	55	141	330
2	30	83	3	116	3	21	25	49	30	83	3	116	11	31	46	88
3	21	0	252	273	3	1	77	81	21	0	252	273	3	1	69	73
	148	87	256	491	148	87	256	491	148	87	256	491	148	87	256	491
Prod Acc %	95.1	71.6	92.3		39.3	42.9	95.1		95.1	71.6	92.3		40.6	35.2	94.5	
User Acc %	65.5	95.4	98.4		95.9	24.2	30.1		65.5	95.4	98.4		90.5	35.6	27.0	
Overall Acc %	87.98				48.88				87.98				47.7			

The same experiment on the 3-category dataset is repeated for RVQ with  $M = 4$  codevectors-per-stage and  $P = 8$  stages. The corresponding RVQ codebook, Class-conditional Probability Matrix, and the Error matrix for MBRVQ-based method and Feature-count Rule-based methods are shown in Figure 21, Table 11, and Table 12; respectively.

It can be seen in Table 12 that the MBRVQ-based methods are far superior to Feature-count Rule-based methods. Moreover, as compared to the RVQ with  $M = 2$  and  $P = 16$ , the classification performance of the RVQ-based classifier for  $M = 4$  and  $P = 4$  is superior. For example, in case the latter the MBRVQ-based classifiers have an overall accuracy of over 90 %, as shown in Table 12; whereas, it can be seen in Table 10 that the overall accuracy of the MBRVQ-based classifiers for  $M = 2$  and  $P = 16$  is 88%, approximately. The same trend can be seen for the Feature-count Rule-based classification.

The classification performances of the four RVQ-based classifiers are also calculated for all the values of Markov order from 0 to  $P-1$ . For  $M=2$  and  $P=16$ , the over-

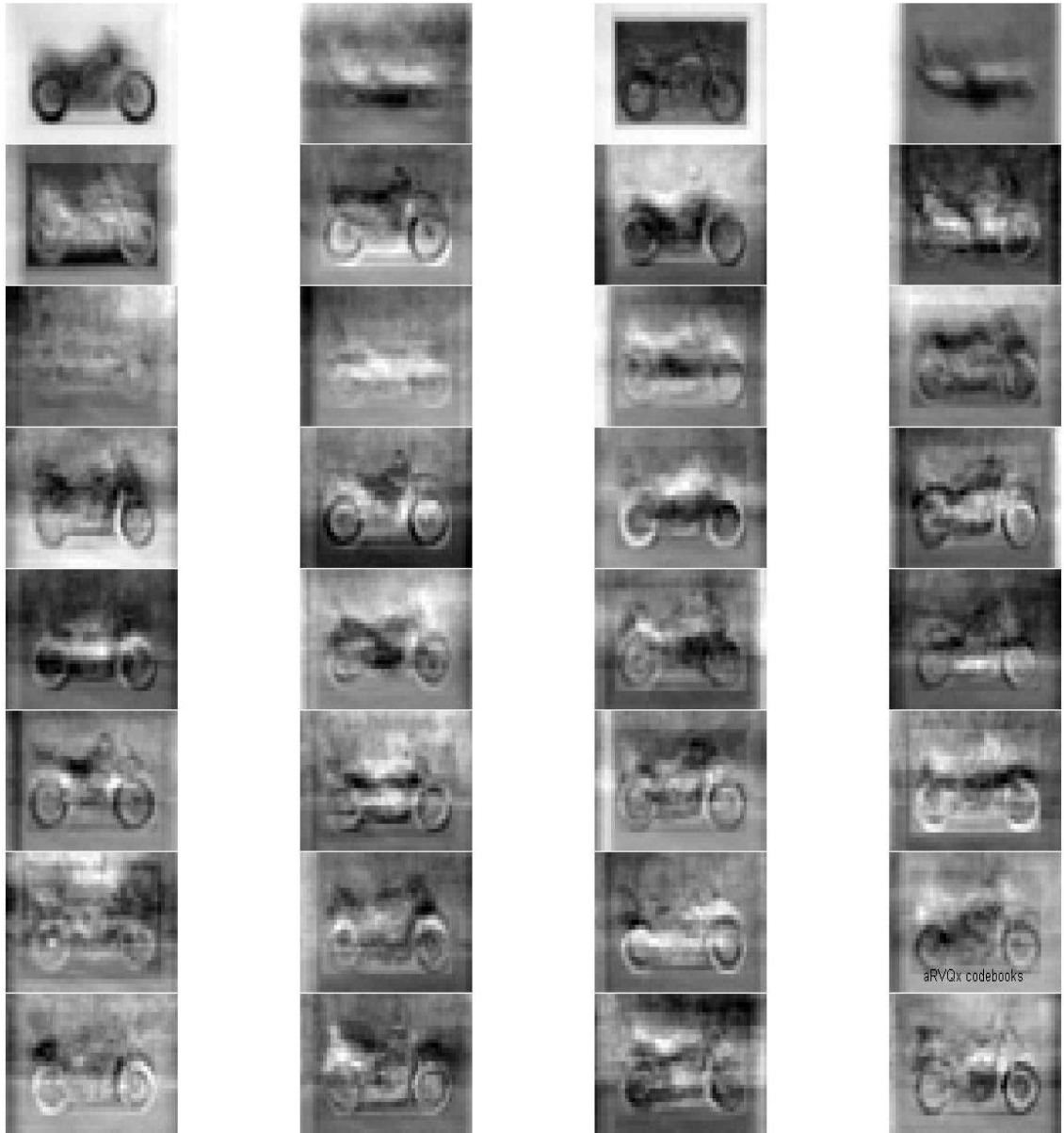


Figure 21. RVQ codebook for RVQ with  $M=4$ ,  $P=8$ . The dataset consists of Plane, Car, and Motorbike classes.

Table 11. Class-conditional Probability Matrix for RVQ with  $M = 4$  and  $P = 8$ . Class 1, 2, 3 are Plane, Car, Motorbike, respectively.

Stage	Plane				Car				Motorbike			
	$CV^1$	$CV^2$	$CV^3$	$CV^4$	$CV^1$	$CV^2$	$CV^3$	$CV^4$	$CV^1$	$CV^2$	$CV^3$	$CV^4$
$p=1$	0.08	0.27	0	0.65	0	1	0	0	0.86	0.03	0.1	0.01
$p=2$	0.15	0.17	0.55	0.13	0	0.44	0.361	0.194	0.16	0.36	0.18	0.3
$p=3$	0	0.3	0.3	0.4	0.139	0.167	0.22	0.472	0	0.4	0.29	0.31
$p=4$	0.23	0.02	0.31	0.44	0.139	0.22	0.33	0.306	0.17	0.18	0.34	0.31
$p=5$	0.19	0.11	0.42	0.28	0.389	0.361	0.083	0.167	0.08	0.31	0.12	0.49
$p=6$	0.24	0.26	0.29	0.21	0.22	0.25	0.167	0.361	0.17	0.13	0.27	0.43
$p=7$	0.16	0.26	0.37	0.21	0.22	0.167	0.194	0.417	0.26	0.18	0.25	0.31
$p=8$	0.14	0.29	0.16	0.41	0.167	0.25	0.278	0.306	0.31	0.17	0.24	0.28

Table 12. Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has  $M = 4$  codevectors-per-stage and  $P = 8$  stages.

Classes	MBRVQ, RoE				Feature-count Rule, RoE				MBRVQ, CostERoE				Feature-count, CostERoE			
	1	2	3		1	2	3		1	2	3		1	2	3	
1	126	15	4	145	146	61	151	358	127	13	4	144	147	58	134	339
2	22	72	2	96	0	25	25	50	21	74	1	96	0	28	40	68
3	0	0	250	250	2	1	80	83	0	0	251	251	1	1	82	84
	148	87	256	491	148	87	256	491	148	87	256	491	148	87	256	491
Prod Acc %	85.1	82.8	97.7		98.6	28.7	31.3		85.8	85.1	98		99.3	32.2	32	
User Acc %	86.9	75	100		40.8	50	96.4		88.2	77.1	100		43.4	41.2	97.6	
Overall Acc %	91.24				51.12				92.06				52.34			

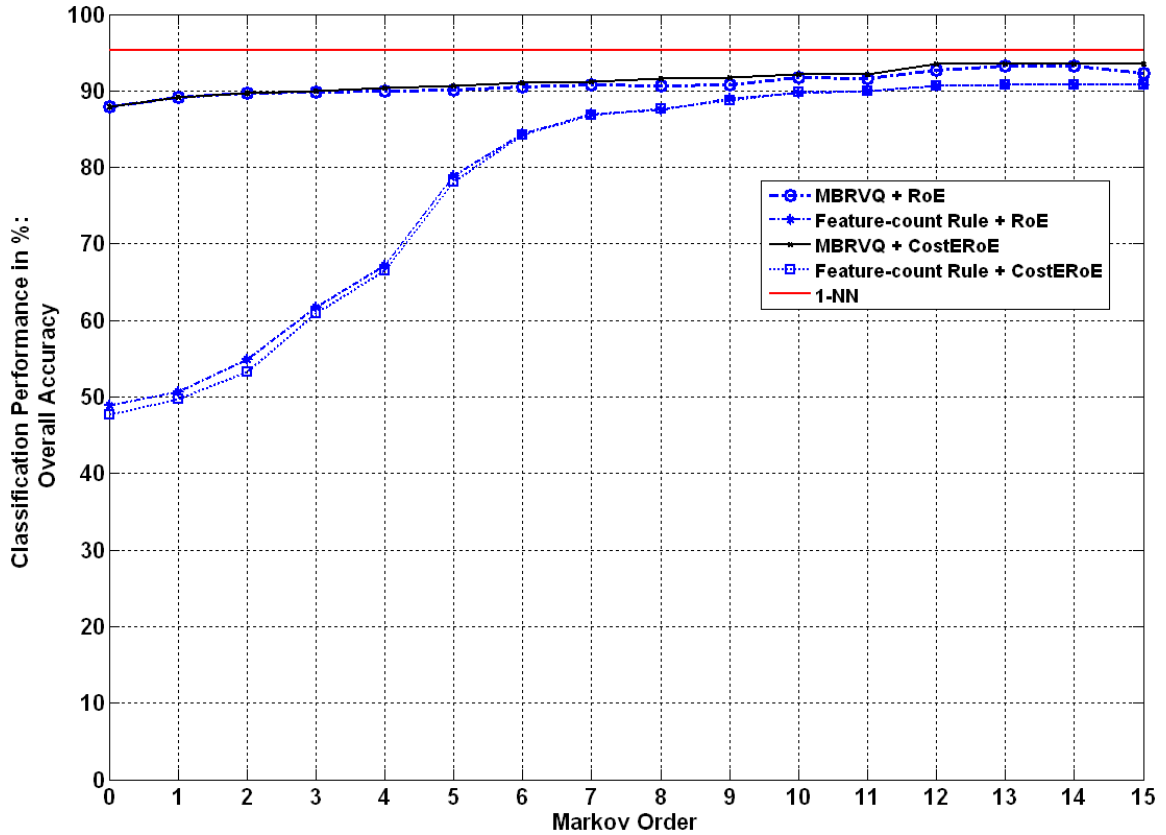


Figure 22. Classification performance for different RVQ-based classifiers with  $M = 2$  and  $P = 16$ . The dataset consists of Plane, Car, and Motorbike classes from Caltech101.

all accuracies are plotted for these four methods in Figure 22. Similarly, the classification performances of the RVQ-based classifiers are also plotted in Figure 23 for RVQ with  $M=4$  codevectors-per-stage and  $P = 8$  stages. It can be seen in Figure 22 and Figure 23 that MBRVQ classifier with CostERoE constraint outperforms the other RVQ-based schemes tested on this dataset.

### Support Vector Machine (SVM) Classifier with the SIFT Feature

A support vector machine (SVM) [9] constructs a hyperplane or set of hyperplanes in a high or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane

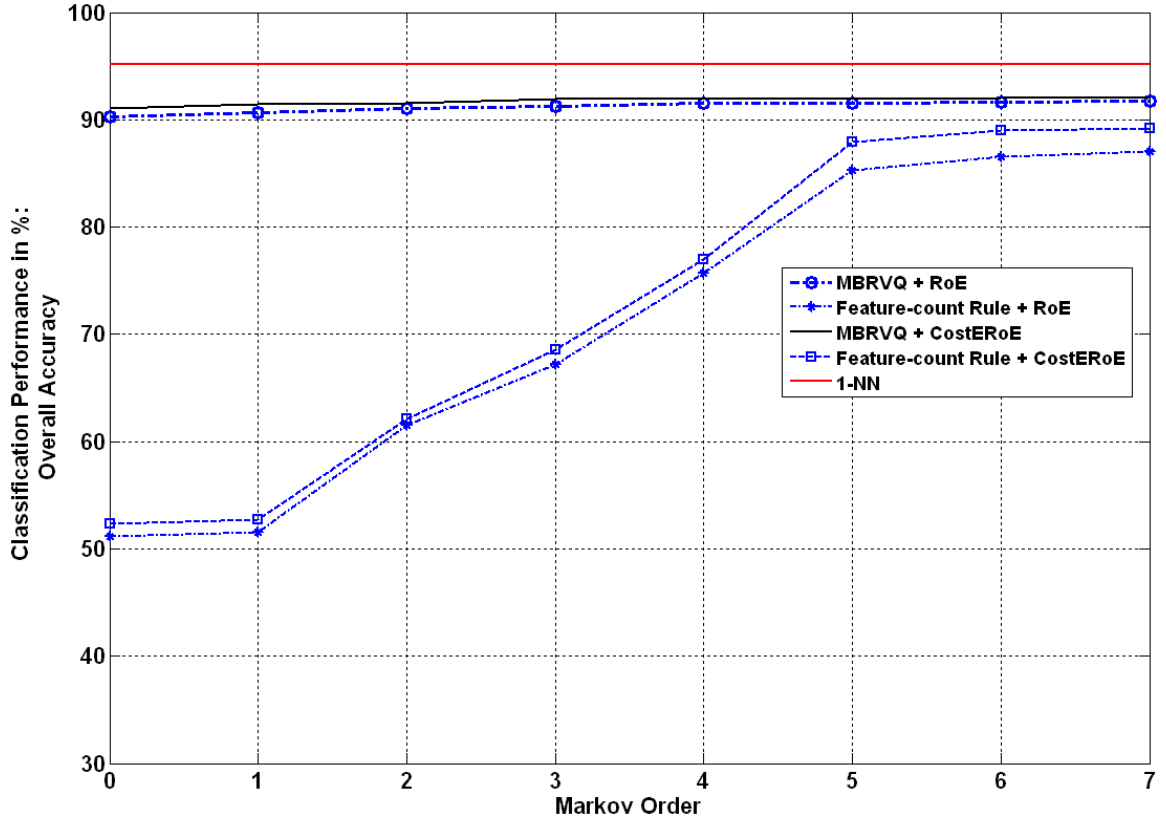


Figure 23. Classification performance for different RVQ-based classifiers with  $M = 4$  and  $P = 8$ . The dataset consists of Plane, Car, and Motorbike classes from Caltech101.

that has the largest distance to the nearest training data point of any class, since in general the larger the margin the lower the generalization error of the classifier. The training points are called support vectors.

As a classifier, SVM is a binary classifier. For the applications of multi-classification, SVM has popularly been used in two ways: one-versus-one, and one-versus-all. In one-versus-one, binary SVM classifiers are built that distinguish between every pair of classes. In this method, classification is done by a *max-wins* voting strategy, in which every classifier assigns a test input to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the most votes determines the instance classification. In one-versus-all strategy, binary SVM classifiers



are constructed to classify between one of the classes and the rest of the classes. Classification of a test input for the one-versus-all case is done by a *winner-takes-all* strategy, in which the classifier with the highest output function assigns the class to the test input.

In this experiment, one-versus-one strategy has been used to perform the SVM-based classification. The choice of the features for the SVM classification is SIFT with chi-squared distance kernel. SIFT stands of *scale-invariant feature transform*. SIFT were first designed and used for SVM-based classification by David Lowe [3]. SIFT feature is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes. Because of the invariance properties of the SIFT feature, it is widely used with SVM in the applications of high-level classification, such as recognition of objects-of-interest in images and videos.

The classification results of the SVM classifier are shown in Table 13. The SIFT feature extracted from each image of the 3-category dataset is a 128x20 vector. It can be seen from Table 12 and Table 13, that SVM classification performed with SIFT feature is almost as good as the MBRVQ-based classifiers.

Table 13. Error matrix for SVM classifier with SIFT feature and chi-squared distance kernel.

<b>Classes</b>	<b>1</b>	<b>2</b>	<b>3</b>	
<b>1</b>	126	11	3	140
<b>2</b>	21	75	1	97
<b>3</b>	1	1	252	254
	148	87	256	491
<b>Prod Acc %</b>	<b>85.1</b>	<b>86.2</b>	<b>98.4</b>	
<b>User Acc %</b>	<b>90</b>	<b>77.3</b>	<b>99.2</b>	
<b>Overall Acc %</b>	<b>92.26</b>			

## Experiments: Set 2

In this second set of experiments, the RVQ settings remain the same as in the first set of experiments. The only difference is the dataset. In this case, the dataset solely consists of the images of motorbikes from Caltech101. The Motorbike dataset is sub-classified into two to three distinct categories. The two-category Motorbike dataset comprises the classes Heavy, and Sports. Class Heavy refers to heavy motorbikes, and the class Sports refers to sports motorbikes. This 2-category dataset is termed as HS-Motorbike. The typical images of the two sub-classes are shown in Figure 24. Similarly, the 3-category Motorbike dataset contains the classes Heavy, Heavy-Sports, and Light-Sports. This dataset is termed HHSL-Motorbike. The typical images of the 3-category Motorbike dataset are shown in Figure 25.

The results for the 3-category HHSL-Motorbike dataset are presented first. The RVQ codebook for  $M = 2$  and  $P = 16$ , the corresponding Class-conditional Transition Probability Matrix, and the classification results are shown in Figure 26, Table 14, and Table 15; respectively. Similarly, for RVQ with  $M = 2$  and  $P = 16$ , the classification performance for Markov order ranging from 0 to  $P-1 = 15$  is shown in Figure 27.

The same experiment on the 3-category HHSL-Motorbike dataset is repeated for RVQ with  $M = 4$  codevectors-per-stage and  $P = 8$  stages. The corresponding RVQ codebook, Class-conditional Transitional Probability Matrix, and the error matrix for MBRVQ-based method and Feature-count Rule-based methods are shown in Figure 28, Table 16, and Table 17; respectively. Similarly, for RVQ with  $M = 4$  and  $P = 8$ , the classification performance for Markov order ranging from 0 to  $P-1 = 7$  is shown in Figure 29.

Heavy	Sports
	
	
	

Figure 24. Training dataset for 2-category classification. The classes are Heavy, and Sports motorbikes.

Heavy	Heavy-Sports	Light-Sports
		
		
		

Figure 25. Training dataset for 3-category classification. The classes are Heavy, Heavy-Sports, and Light-Sports motorbikes.



Figure 26. RVQ codebook for RVQ with  $M = 2$ ,  $P = 16$ . The dataset consists of Heavy, Heavy-Sports, and Light-Sports motorbike classes.

Table 14. Class-conditional Probability Matrix for RVQ with  $M=2$  and  $P=16$ . Class 1, 2, 3 are Heavy, Heavy-Sports, and Light-Sports motorbikes, respectively.

Stage	Class	[ $CV_1$ , $CV_2$ ]	Stage	Class	[ $CV_1$ , $CV_2$ ]
<b>1</b>	1	[0.23, 0.45]	<b>9</b>	1	[0.26, 0.39]
	2	[0.35, 0.31]		2	[0.46, 0.24]
	3	[0.42, 0.23]		3	[0.28, 0.37]
<b>2</b>	1	[0.29, 0.34]	<b>10</b>	1	[0.24, 0.44]
	2	[0.54, 0.29]		2	[0.39, 0.27]
	3	[0.17, 0.37]		3	[0.37, 0.29]
<b>3</b>	1	[0.32, 0.36]	<b>11</b>	1	[0.53, 0.20]
	2	[0.39, 0.22]		2	[0.20, 0.42]
	3	[0.29, 0.42]		3	[0.27, 0.38]
<b>4</b>	1	[0.16, 0.48]	<b>12</b>	1	[0.34, 0.33]
	2	[0.37, 0.30]		2	[0.39, 0.30]
	3	[0.47, 0.22]		3	[0.27, 0.37]
<b>5</b>	1	[0.29, 0.38]	<b>13</b>	1	[0.18, 0.34]
	2	[0.18, 0.49]		2	[0.45, 0.33]
	3	[0.53, 0.13]		3	[0.36, 0.33]
<b>6</b>	1	[0.56, 0.22]	<b>14</b>	1	[0.37, 0.29]
	2	[0.40, 0.30]		2	[0.31, 0.36]
	3	[0.03, 0.48]		3	[0.32, 0.35]
<b>7</b>	1	[0.43, 0.24]	<b>15</b>	1	[0.28, 0.40]
	2	[0.07, 0.59]		2	[0.34, 0.33]
	3	[0.50, 0.17]		3	[0.39, 0.27]
<b>8</b>	1	[0.39, 0.26]	<b>16</b>	1	[0.38, 0.31]
	2	[0.31, 0.36]		2	[0.35, 0.31]
	3	[0.30, 0.38]		3	[0.35, 0.32]

Table 15. Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has  $M = 2$  codevectors-per-stage and  $P = 16$  stages.

Class	MBRVQ, RoE				Feature-count Rule, RoE				MBRVQ, CostERoE				Feature-count, CostERoE			
	1	2	3		1	2	3		1	2	3		1	2	3	
1	229	92	21	342	148	59	12	219	230	92	13	335	154	80	20	254
2	21	89	7	117	88	121	4	213	20	88	5	113	41	95	5	141
3	44	37	79	160	58	38	91	187	44	38	89	171	99	43	82	224
	294	218	107	619	294	218	107	619	294	218	107	619	294	218	107	619
Prod Acc %	77.9	40.8	73.8		50.3	55.5	85		78.2	40.4	83.2		52.4	43.6	76.6	
User Acc %	67	76.1	49.4		67.6	56.8	48.7		68.7	77.9	52		60.6	67.4	36.6	
Overall Acc %	64.14				58.16				65.75				53.47			

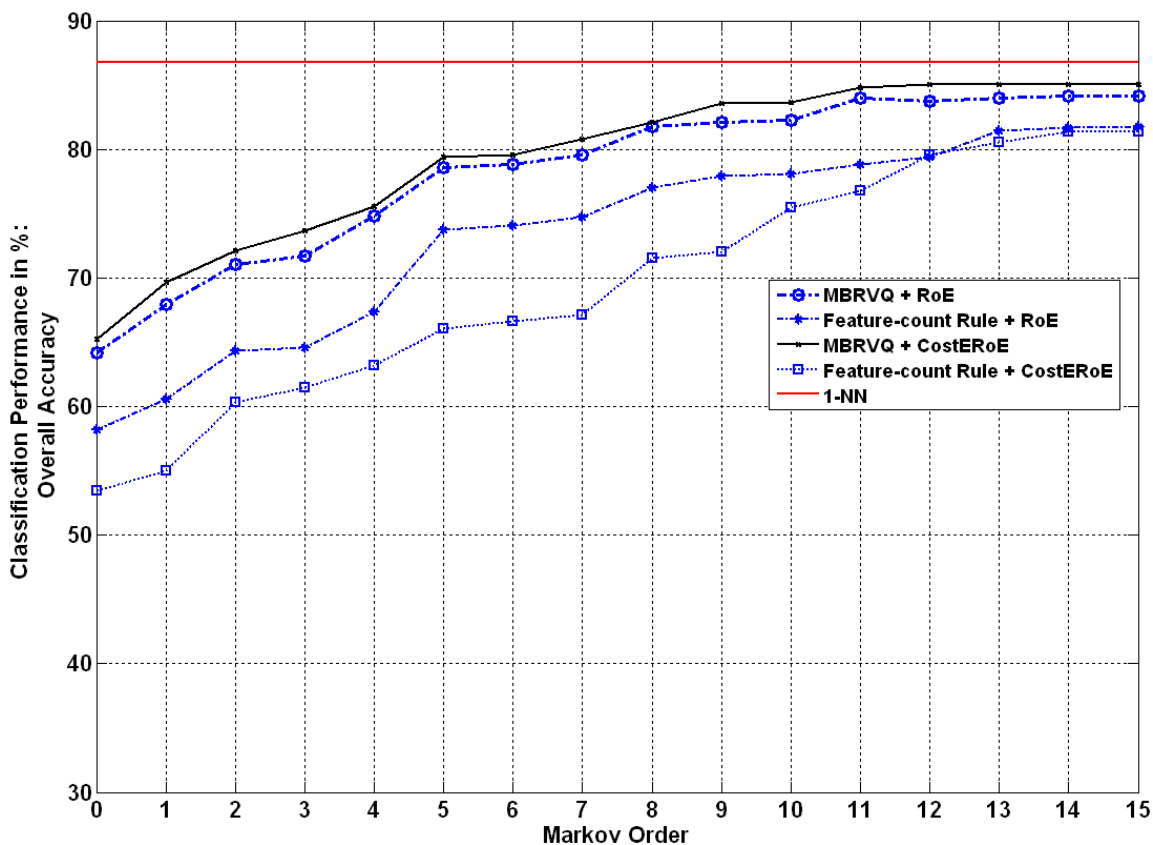


Figure 27. Classification performance for different RVQ-based classifiers with  $M = 2$  and  $P = 16$ . The dataset consists of Heavy, Heavy-Sports, and Light-Sports motorbike classes from Caltech101.



Figure 28. RVQ codebook for RVQ with  $M = 2$ ,  $P = 16$ . The dataset consists of Heavy, Heavy-Sports, and Light-Sports motorbike classes.

Table 16. Class-conditional Probability Matrix for RVQ with  $M = 4$  and  $P = 8$ . Class 1, 2, 3 are Heavy, Heavy-Sports, and Light-Sports motorbikes, respectively.

Stage	Class	$CV_1$	$CV_2$	$CV_3$	$CV_4$
<b>1</b>	1	0.21	0.47	0.27	0.51
	2	0.37	0.16	0.51	0.27
	3	0.41	0.37	0.22	0.22
<b>2</b>	1	0.18	0.78	0.50	0.06
	2	0.19	0.11	0.14	0.89
	3	0.63	0.11	0.36	0.05
<b>3</b>	1	0.6	0.39	0.23	0.31
	2	0.4	0.52	0.23	0.15
	3	0	0.09	0.54	0.54
<b>4</b>	1	0.83	0.25	0.29	0.29
	2	0.13	0.38	0.33	0.37
	3	0.04	0.38	0.38	0.35
<b>5</b>	1	0.38	0.46	0.19	0.34
	2	0.38	0.25	0.47	0.30
	3	0.25	0.29	0.35	0.36
<b>6</b>	1	0.33	0.38	0.30	0.32
	2	0.54	0.31	0.22	0.34
	3	0.13	0.31	0.48	0.33
<b>7</b>	1	0.20	0.42	0.32	0.40
	2	0.21	0.44	0.52	0.23
	3	0.59	0.14	0.16	0.37
<b>8</b>	1	0.28	0.30	0.28	0.44
	2	0.35	0.17	0.45	0.36
	3	0.38	0.53	0.26	0.20

Table 17. Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has  $M = 4$  codevectors-per-stage and  $P = 8$  stages.

Class	MBRVQ, RoE				Feature-count Rule, RoE				MBRVQ, CostERoE				Feature-count Rule, CostERoE			
	1	2	3		1	2	3		1	2	3		1	2	3	
<b>1</b>	232	90	18	340	151	56	11	218	233	85	13	331	155	51	10	216
<b>2</b>	20	93	6	119	87	125	3	215	19	97	5	121	41	127	5	173
<b>3</b>	42	35	83	160	56	37	93	186	42	36	89	167	98	40	92	230
	294	218	107	619	294	218	107	619	294	218	107	619	294	218	107	619
<b>Prod Acc %</b>	78.9	42.7	77.6		51.4	57.3	86.9		79.3	44.5	83.2		52.7	58.3	86	
<b>User Acc %</b>	68.2	78.2	51.9		69.3	58.1	50		70.4	80.2	53.3		71.8	73.4	40	
<b>Overall Acc %</b>	<b>65.91</b>				<b>59.61</b>				<b>67.69</b>				<b>60.42</b>			



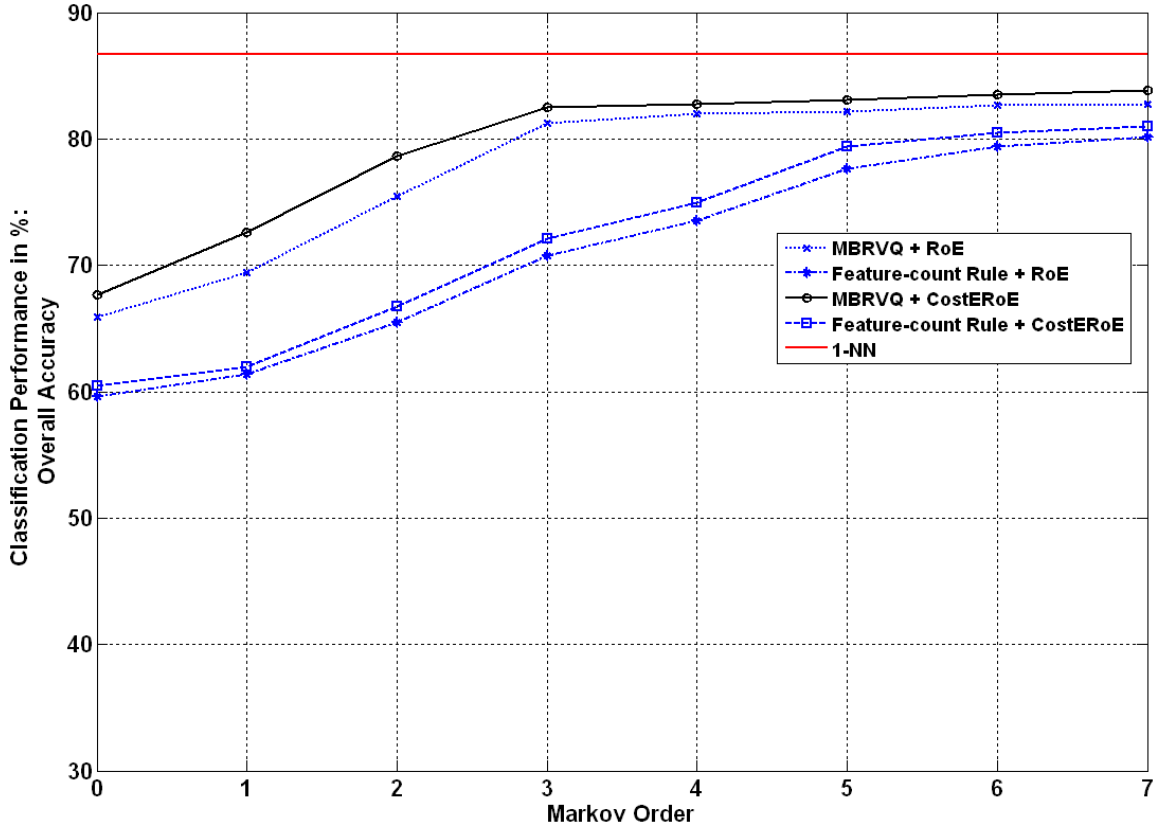


Figure 29. Classification performance for different RVQ-based classifiers with  $M = 4$  and  $P = 8$ . The dataset consists of Heavy, Heavy-Sports, and Light-Sports motorbike classes from Caltech101.

It can be seen in Table 15 and Table 17 that the MBRVQ-based methods are far superior to Feature-count Rule-based methods. Moreover, as compared to the RVQ with  $M = 2$  and  $P = 16$ , the classification performance of the RVQ-based classifier for  $M = 4$  and  $P = 4$  is slightly better for this dataset. For example, in case of the latter, the MBRVQ-based classifiers have an overall accuracy of over 65 %, as shown in Table 17; whereas, it can be seen in Table 15 that the overall accuracy of the MBRVQ-based classifiers for  $M = 2$  and  $P = 16$  is approximately 1% lower, comparatively. The same trend can be seen for the Feature-count Rule-based classification. Moreover, in both RVQ settings, CostERoE constraint yielded better classification results for MBRVQ as compared to RoE constraint. Similarly, it can be seen in Figure 27 and Figure 28 that MBRVQ classifier

with CostERoE constraint outperforms the other RVQ-based schemes tested on this dataset.

After investigating the MRVQ classifier for the 3-category Motorbike dataset shown in Figure 24, the classifier is studied for the 2-category Motorbike dataset (Figure 23), which is formed by combining the Heavy-Sports and Light-Sports motorbikes categories into one category of motorbikes named Sports. The training set size is 600 images, with 300 images in each class. There are 270 images in the test set, with 90 and 180 images in Heavy and Sports motorbike classes, respectively. The images are resized to 150 x 250 and are converted to grayscale.

Similar to the 3-category case, the experiments on the 2-category HS-Motorbike dataset are carried out for two settings of RVQ:  $M = 2$  and  $P = 16$ , and  $M = 4$  and  $P = 8$ . The RVQ codebooks for both the settings look similar to the corresponding RVQ codebooks of the 3-category HHSLs-Motorbike dataset. For  $M = 2$  and  $P = 16$ , Class-conditional Transition Probability Matrix, and the classification results are shown in Table 18, and Table 19; respectively. The corresponding Class-conditional Transitional Probability Matrix, and the error matrix for  $M = 4$  and  $P = 8$  are shown in Table 20, and Table 21; respectively. The Markov order is 0 in these results. It can be seen in Table 19 and Table 20 that the MBRVQ-based methods are far superior to Feature-count Rule-based methods. Moreover, as compared to the RVQ with  $M = 2$  and  $P = 16$ , the classification performance of the RVQ-based classifier for  $M = 4$  and  $P = 8$  is slightly better. The same trend can be seen for the Feature-count Rule-based classification. Moreover, in both RVQ settings, compared to RoE constraint, CostERoE constraint yielded better classification results for MBRVQ classifier. The classification

performances of the four RVQ-based classifiers are also calculated for all the values of Markov order from 0 to  $P-1$ . For  $M=2$  and  $P=16$ , the overall accuracies are plotted for these four methods in Figure 30. Similarly, the classification performances of the RVQ-based classifiers are also plotted in Figure 31 for RVQ with  $M=4$  codevectors-per-stage and  $P = 8$  stages. It can be seen in Figure 30 and Figure 31 that MBRVQ classifier with CostERoE constraint outperforms the other RVQ-based schemes tested on this dataset.

Table 18. Class-conditional Probability Matrix for RVQ with  $M = 2$  and  $P = 16$ . Class 1, 2 are Heavy, and Sports motorbikes, respectively.

Stage	Class	$CV_1$	$CV_2$	Stage	Class	$CV_1$	$CV_2$
<b>1</b>	1	0.45	0.56	<b>9</b>	1	0.52	0.48
	2	0.55	0.44		2	0.48	0.52
<b>2</b>	1	0.65	0.46	<b>10</b>	1	0.49	0.51
	2	0.35	0.54		2	0.51	0.49
<b>3</b>	1	0.65	0.46	<b>11</b>	1	0.56	0.44
	2	0.35	0.54		2	0.44	0.56
<b>4</b>	1	0.37	0.65	<b>12</b>	1	0.51	0.49
	2	0.63	0.35		2	0.49	0.51
<b>5</b>	1	0.73	0.42	<b>13</b>	1	0.60	0.40
	2	0.27	0.58		2	0.40	0.60
<b>6</b>	1	0.74	0.46	<b>14</b>	1	0.48	0.51
	2	0.26	0.54		2	0.52	0.49
<b>7</b>	1	0.55	0.45	<b>15</b>	1	0.62	0.41
	2	0.45	0.55		2	0.38	0.59
<b>8</b>	1	0.51	0.48	<b>16</b>	1	0.49	0.50
	2	0.49	0.52		2	0.51	0.50

Table 19. Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has  $M = 2$  codevectors-per-stage and  $P = 16$  stages.

Classes	MBRVQ, RoE			Feature-count Rule, RoE			MBRVQ, CostERoE			Feature-count Rule, CostERoE		
	1	2		1	2		1	2		1	2	
<b>1</b>	74	55	129	44	76	120	75	56	131	49	91	140
<b>2</b>	16	125	141	46	104	150	15	124	139	41	89	130
	90	180	270	90	180	270	90	180	270	90	180	270
<b>Prod Acc %</b>	82.2	69.4	75.8	48.9	57.8	53.4	83.3	68.9	76.1	54.4	49.4	51.9
<b>User Acc %</b>	57.4	88.7	73.1	36.7	69.3	53	57.3	89.2	73.3	35	68.5	51.8
<b>Overall Acc %</b>	<b>73.70</b>			<b>54.81</b>			<b>73.70</b>			<b>51.11</b>		

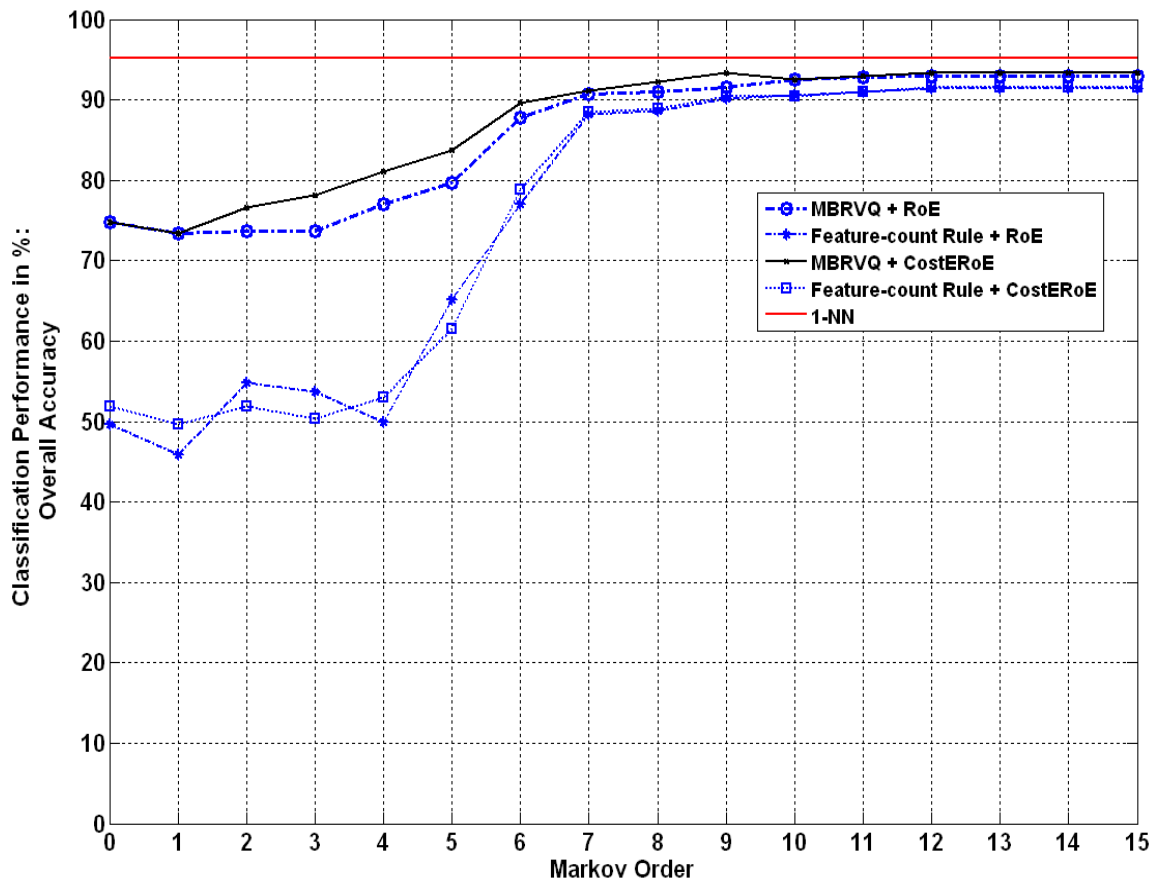


Figure 30. Classification Performance for different RVQ-based classifiers with  $M = 2$  and  $P = 16$ . The dataset consists of Heavy and Sports motorbike classes from Caltech101.

Table 20. Class-conditional Probability Matrix for RVQ with  $M = 4$  and  $P = 8$ . Class 1, 2 are Heavy, and Sports motorbikes, respectively.

Stage	Class	$CV_1$	$CV_2$	$CV_3$	$CV_4$
1	1	0.42	0.69	0.45	0.65
	2	0.58	0.31	0.55	0.35
2	1	0.55	0.61	0.67	0.33
	2	0.45	0.39	0.33	0.67
3	1	0.57	0.47	0.72	0.16
	2	0.43	0.53	0.28	0.84
4	1	0.61	0.58	0.31	0.56
	2	0.39	0.42	0.69	0.44
5	1	0.36	0.59	0.45	0.57
	2	0.64	0.41	0.55	0.43
6	1	0.60	0.57	0.54	0.40
	2	0.40	0.43	0.46	0.60
7	1	0.40	0.71	0.45	0.50
	2	0.60	0.29	0.55	0.50
8	1	0.61	0.30	0.51	0.59
	2	0.39	0.70	0.49	0.41

Table 21. Error matrix for MBRVQ with RoE, Feature-count Rule with RoE, MBRVQ with CostERoE, and Feature-count Rule with CostERoE. The Markov order is zero, and RVQ has  $M = 4$  codevectors-per-stage and  $P = 8$  stages.

Classes	MBRVQ, RoE			Feature-count Rule, RoE			MBRVQ, CostERoE			Feature-count Rule, CostERoE		
	1	2		1	2		1	2		1	2	
1	71	52	123	39	96	135	72	52	124	43	89	132
2	19	128	147	51	84	135	18	128	146	47	91	138
	90	180	270	90	180	270	90	180	270	90	180	270
Prod Acc %	78.9	71.1	75	43.3	46.7	45	80	71.1	75.6	47.8	50.6	49.2
User Acc %	57.7	87.1	72.4	28.9	62.2	45.6	58.1	87.7	72.9	32.6	65.9	49.3
Overall Acc %	73.70			45.56			74.07			49.63		

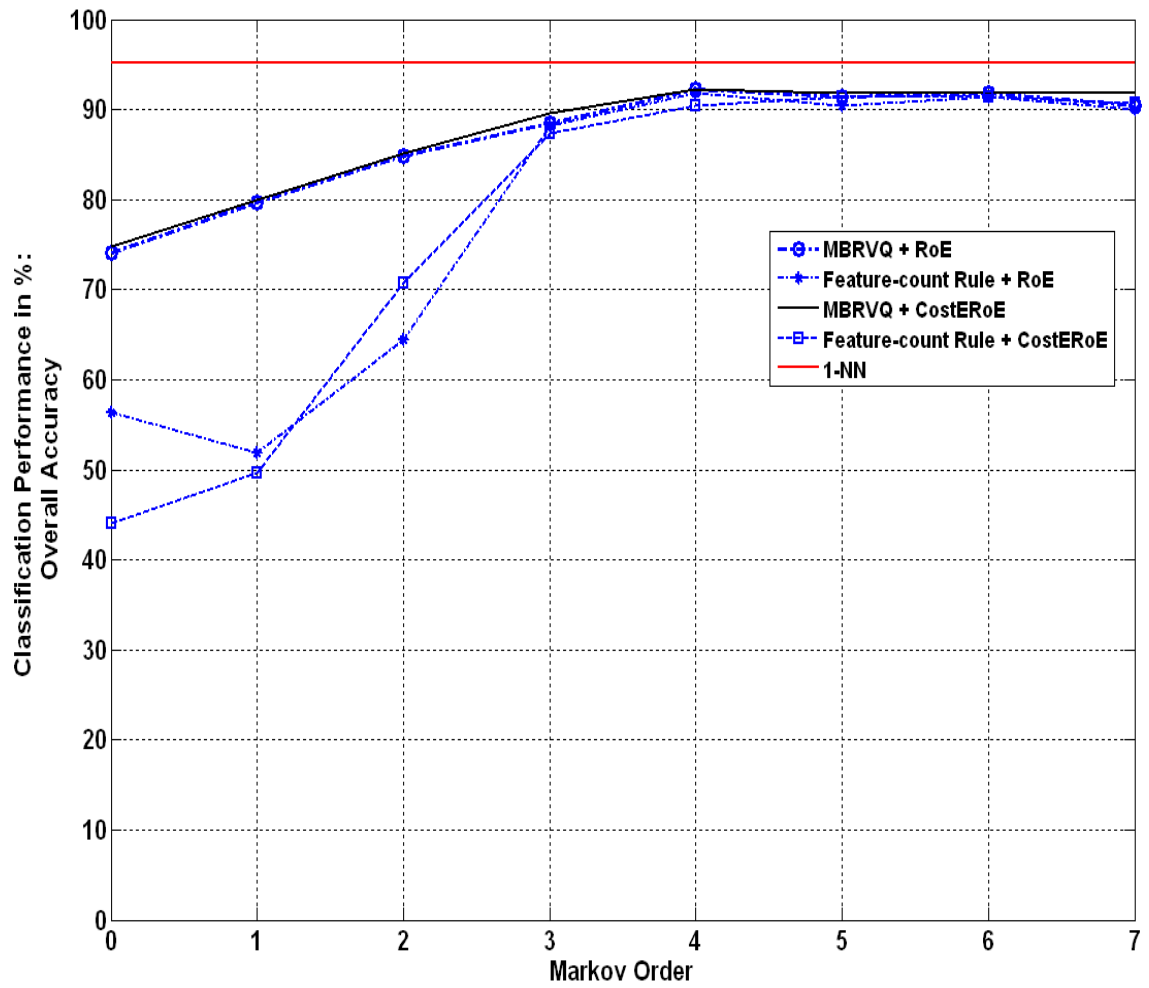


Figure 31. Classification Performance for different RVQ-based classifiers with  $M = 4$  and  $P = 8$ . The dataset consists of Heavy and Sports motorbike classes from Caltech101.

## Computational and Memory Cost Analysis

The computational and memory costs of MBRVQ classifier with CostERoE constraint are given as follows:

Computational Cost =  $kMP$  multiplications and additions.

Memory cost =  $kMP + C(P-O)M^{(O+1)} + |T|$  bytes,

where  $k$  is the dimensionality of the input space,  $C$  is the number of classes,  $O$  is the Markov order and  $|T|$  is the size of the training set. The first, second, and the third terms of the memory cost are storage costs of the RVQ codebook, labels, and CostERoE constraint. At this point, it is pertinent to emphasize the cost effectiveness of MBRVQ classification over the 1-NN-based classification. The computational cost of 1-NN classification is  $k|T|$  multiplications and additions, and the associated memory cost is  $k|T| + T$  bytes, where  $k|T|$  is the cost for the storage of the training set, and  $|T|$  is the cost for storing the class labels. The costs of MBRVQ classification on the datasets shown in Figure 12, 14, 16, 24, and 25 with CostERoE are shown and compared in Figure 32 and Figure 33.

### Experiments: Set 3

Different thresholds  $T_h$  on the class-conditional probabilities associated with the Markov sub-tuples will be applied to see their effects on the classification performance of MRVQ classifier. Intuitively, the threshold  $T_h$  can be thought as a means to weight CAC-clusters in reaching a classification decision. Therefore, with a suitable  $T_h$ , only those CAC clusters can be isolated that contributes most significantly towards reaching the class-membership decision. The value of  $T_h$  is varied from 0 to 0.9. The 2-category HS-Motorbike dataset, Figure 22, is chosen for this set of experiments. MBRVQ scheme is used with CostERoE. In the previous experiments, MBRVQ scheme has been shown to perform better Feature-count rule for MRVQ classification. Moreover, CostERoE is

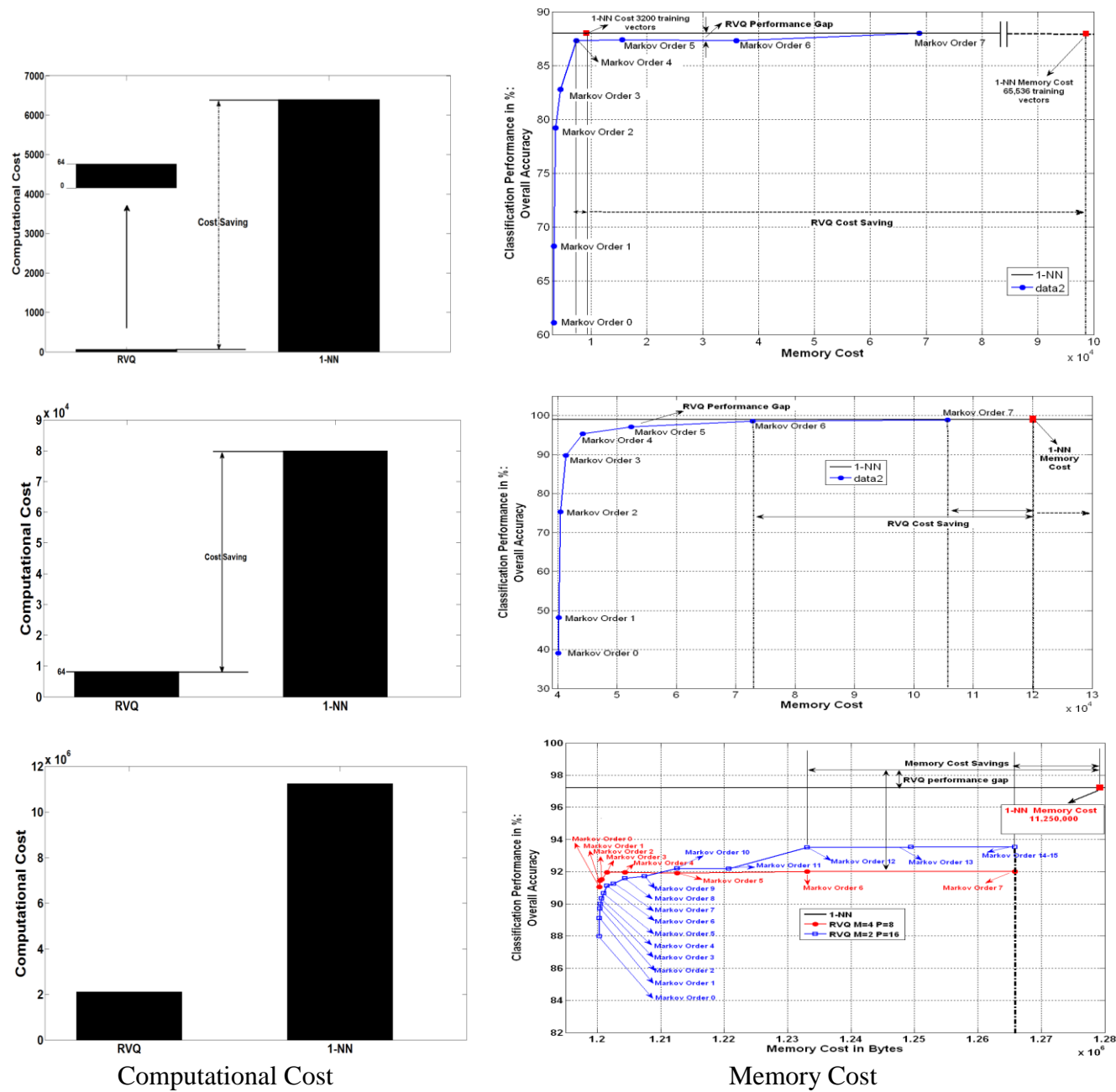


Figure 32. RVQ-versus-1NN: (1<sup>st</sup> Row) 2-category Swiss roll in Figure 13. (2<sup>nd</sup> Row) 4-category Swiss roll in Figure 15. (3<sup>rd</sup> Row) 3-category Caltech101 in Figure 16.



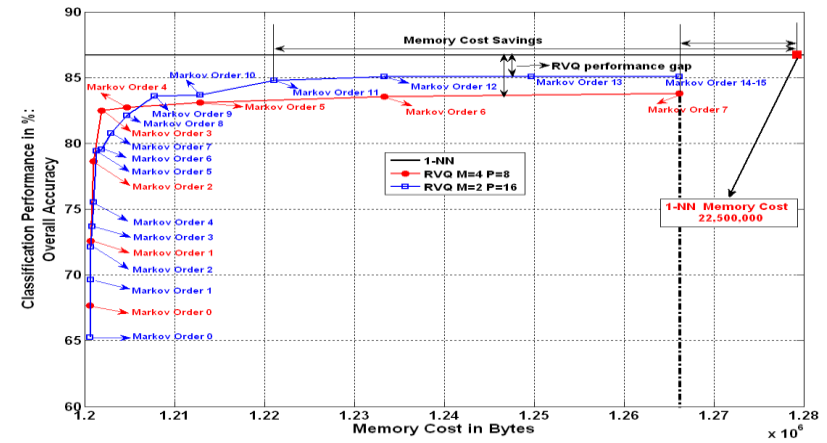
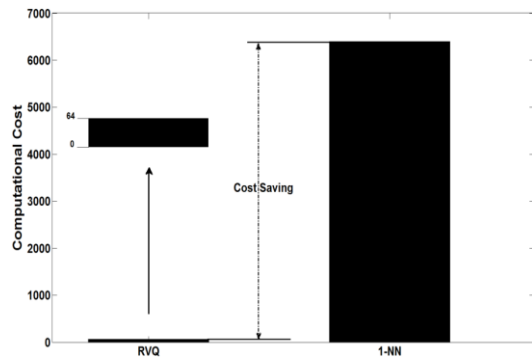
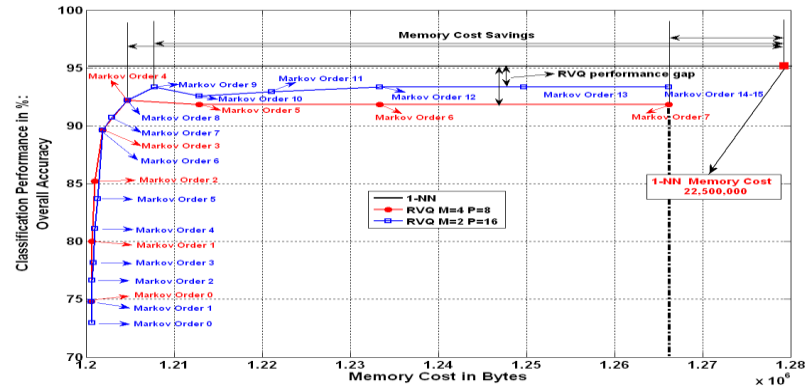
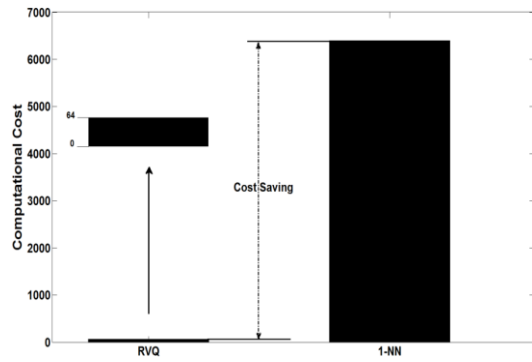


Figure 33. RVQ-versus-1NN: (1<sup>st</sup> Row) 2-category Motorbike dataset from Caltech101 in Figure 23. (2<sup>nd</sup> Row). 3-category Motorbike dataset from Caltech101 in Figure 24.

preferred over RoE because compared to RoE constraint, CostERoE constraint yields comparable classification performance at a better cost.

The experimental results for  $M = 2$  and  $P = 16$ , and  $M = 4$  and  $P = 8$  are shown in Figure 34, where the classification results are measured as the overall accuracy. The results are shown for thresholds  $T_h = \{0, 0.1, 0.2, \dots, 1\}$ . The detailed classification results are presented in Table A1 and Table A2 in Appendix A, respectively, for RVQ settings  $M=2$  and  $P = 16$ , and  $M = 4$  and  $P = 8$ . In Figure 27, the overall accuracies for Markov order ranging from 0 to  $P-1$  are depicted as bars centered at each threshold value of  $T_h$ . Therefore, for  $P = 8$ , the total number of bars for each value of  $T_h$  is eight. Likewise, for  $P = 16$ , there are sixteen bars for each value of  $T_h$ . Moreover, for  $P = 16$ , some values of overall accuracy are capped off at 10 %. This value is to indicate that no class-conditional probabilities of some test inputs were greater than the value of threshold  $T_h$ . As a result, those test inputs were assigned to any of the three categories. Such class assignments are termed as Unknown. In Figure 34, it can be seen that for  $P = 16$  test inputs start to get assigned to Unknown from  $T_h \geq 0.55$  for different values of Markov orders. For  $T_h \geq 0.7$ , all the test inputs for all Markov orders are assigned to the class Unknown.

In the context of the threshold  $T_h$ , it can be seen from the results that the classification performance is upper-bounded by  $T_h = 0$ . It implies that  $T_h > 0$  gives no significant advantage over the case when  $T_h = 0$ . The idea of applying the different values of  $T_h$  is to check if the stage class decision of RVQ can be weighted so that the final class decision is improved. However, the results suggest that under the Markov-Bayesian framework, the class-conditional probabilities associated with each Markov sub-tuples appropriately weight the corresponding stage class decisions to give the final class

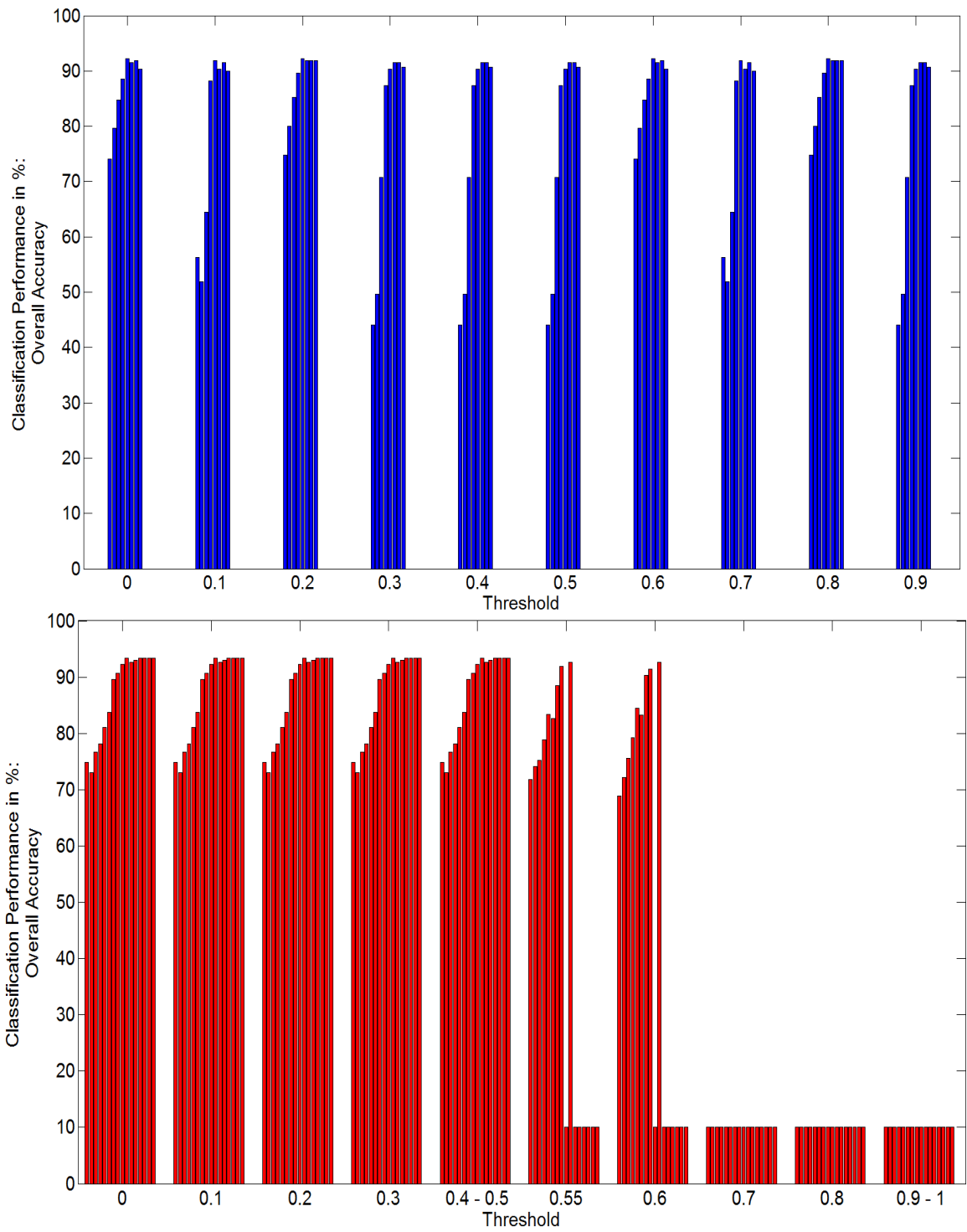


Figure 34. (Top): RVQ  $M = 4$  and  $P = 8$ . (Bottom): RVQ  $M = 2$  and  $P = 16$ .

decision on a test input. Therefore, assigning weights to the stage class decision through different values of  $T_h$  is redundant.

### **More Datasets: Graz**

So far, the test images from Caltech101 database, shown in Figure 16, 24, and 25, largely have very consistent characteristics. The objects in these images are centered with no significant variation in scale, localization and pose. The lighting variations across these images are minimal. Such well-behaved images are very suitable for template matching-based techniques. However, it is felt that more challenging images with objects-of-interest varying in pose, scale, and localization; and variation in overall lighting conditions be considered for the analysis of the proposed RVQ-based classification. Graz dataset [48] has the images with such challenging characteristics. Graz dataset consists of images from three classes: Bicycle, People, and Background. The typical images of each class are shown in Figure 35. The size of each image is 480-by-640 pixels. The training set consists of 200, 200, and 112 images; and the test set comprises 260, 165, and 54 images from Bicycle, People, and Background classes, respectively.

The images in Graz dataset contain objects with varying pose, scale, and localization. Furthermore, multiple objects are also present in many of the images in the dataset. Because of the nature of these variations in Graz dataset, such methods are required that are invariant to these variations. RVQ Being an image-template matching method, classification of objects in Graz dataset using MBRVQ classifier becomes an ill-posed problem.

SVM classifier with SIFT [3] features is a suitable classification method on Graz

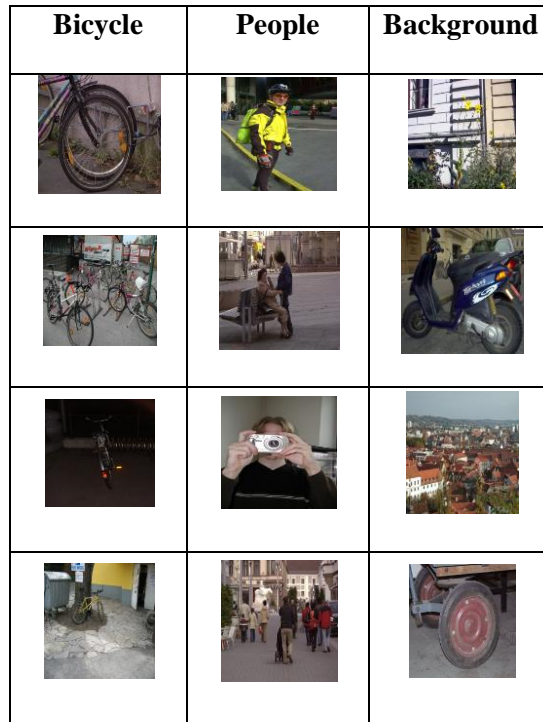


Figure 35. Graz dataset with classes Bicycle, People, and Background.

dataset. SIFT features are largely invariant to the variations present in the images of the Graz dataset. The classification results of the SVM classifier are shown in Table 22, respectively. The overall accuracy of the SVM classifier is 93.95%.

Table 22. Error matrix and classification performance measures for SVM on Graz dataset.

Classes	Bicycle	People	Background	
Bicycle	245	10	0	255
People	7	153	2	162
Background	8	2	52	62
	260	165	54	479
Prod Acc %	94.2	92.7	96.3	
User Acc %	96.1	94.4	83.9	
Overall Acc %	93.95			

As mentioned before, Graz dataset has a great deal of variations in scale, lighting, pose, location of objects, and number of the objects in an image. In the class People in particular, the image varies from just one person to a group of persons at a small scale. The huge range of variations in Graz dataset does not seem to go well with the image pixel-based template matching of RVQ. At this point, two suggestions are proposed to improve the classification performance of MBRVQ classifier: (1) Modify the design method of RVQ codebook suited to SIFT features so that the proposed RVQ classifier can become a robust classifier on a variety of images. Chi-squared distance function is the one of the most suitable distance criterion for SIFT features. Incorporate chi-squared distance function in the design process of RVQ codebook. (2) Employ a sliding window approach where a window is made to slide across a test image and a class decision is made for the image snippet in the window. However, adapting the window to the varying scale of the objects-of-interest in an image will pose a big challenge.

Both the approaches, discussed above, are outside the scope of this research. These methods are left to be investigated in future works of this research.

### **Handwritten-Digits Dataset**

The MNIST database of handwritten digits [49], [50] consists of images of handwritten digits from zero to nine. All the images are centered and nearly uniformly scaled, and have the same size of 28-by-28 pixels in binary scale. The lighting variations are minimal, if any. They typical images in the handwriting database are shown in Figure 36. This database holds another major advantage over the previously used datasets. The size of the handwritten-digits dataset is very large, relatively. The total number of images is 70,000, with 60,000 images in the training set and 10,000 images in the test set.

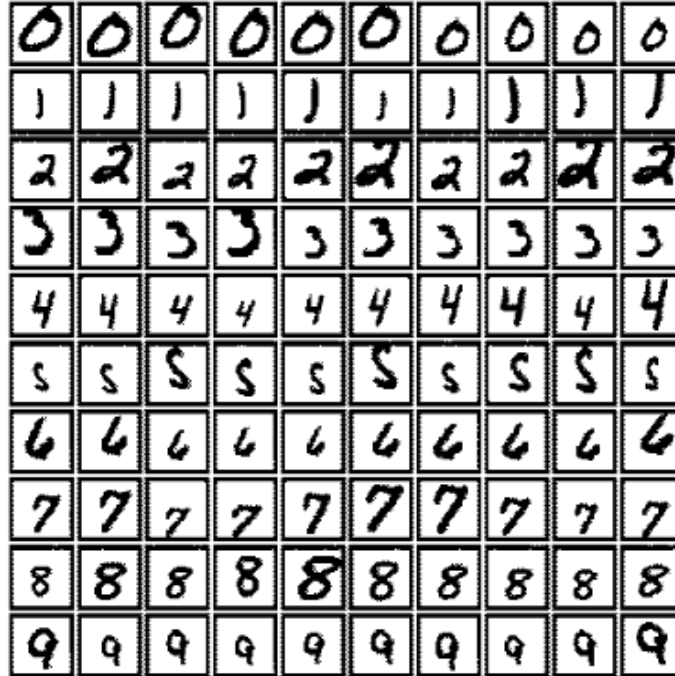


Figure 36. The MNIST database of handwritten digits.

Yann Lecun et al [49] report comprehensive results on the classification of the handwriting database for the digits from zero to nine. The results reported in [49] also serve as a collection of benchmark performances to compare the performance of MBRVQ classifier with. For an initial test on this dataset, MBRVQ classifier is first realized by designing RVQ codebook with  $M = 4$  codevectors-per-stage and  $P = 8$  stages. Markov order is four. The training set consists of 60,000 images for the handwritten digits ranging from zero to nine. The test set comprises 10,000 images. The aim of classification on this dataset is to recognize between each of the ten digits. As a result, the RVQ classifier is designed with one codebook for 10-category classification. The codebook is shown in Figure 37. CostERoE constraint is used to implement the RVQ-based classifier. The classification results are shown in the error matrix in Table 23.

To see the effect of the varying codevectors-per-stage, the classification on

Table 23. Classification performance of MBRVQ classifier with CostERoE constraint on handwritten digits dataset. RVQ has  $M = 4$  and  $P = 8$ , Markov order = 4.

Classes	0	1	2	3	4	5	6	7	8	9	
0	917	0	18	8	2	24	21	3	16	10	1019
1	1	1099	14	5	3	4	6	17	7	6	1162
2	4	2	860	33	15	9	10	28	39	6	1006
3	7	4	32	819	4	52	3	4	61	9	995
4	1	2	23	3	739	10	10	10	15	109	922
5	24	0	5	58	13	723	21	10	44	19	917
6	14	5	22	3	26	25	869	2	21	3	990
7	2	2	20	12	19	8	1	879	18	46	1007
8	8	17	34	54	11	23	15	13	727	19	921
9	2	4	4	15	150	14	2	62	26	782	1061
	980	1135	1032	1010	982	892	958	1028	974	1009	10000
Producer Accuracy %	93.57	96.82	83.33	81.1	75.26	81.1	90.71	85.51	74.64	77.5	
User Accuracy %	90.00	94.58	85.5	82.31	80.15	78.84	87.78	87.3	78.94	73.71	
Overall Accuracy %	84.14										

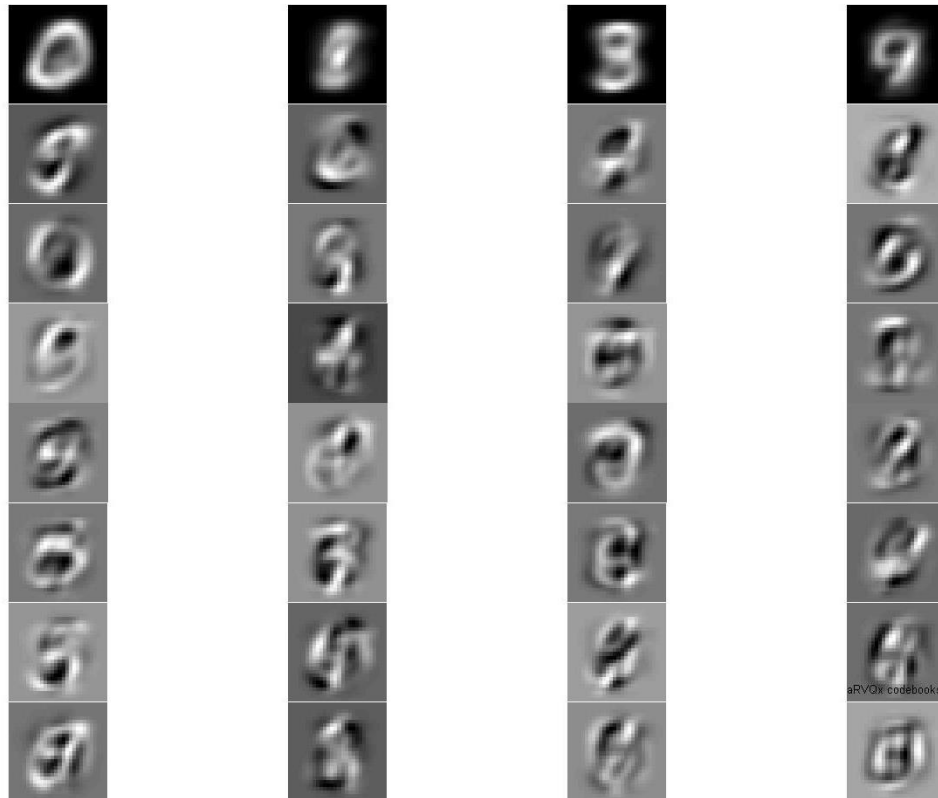


Figure 37. Handwritten digit database. RVQ codebook with  $M = 4$  codevectors-per-stage and  $P = 8$  stages.



dataset is also done for  $M = \{2,3,4,5,6,7,8,9,10,11,12\}$  and  $P = 8$ . The classification performance of MBRVQ classifier is shown in Figure 37, in which the overall accuracies are plotted for all the values of  $M$  and Markov order varying from 0 to  $P-1 = 7$ . It can be seen that the mean overall accuracy peaks at 91.32 % for  $M = 7$  before it starts to decrease. The results suggest that the MBRVQ classifier begins to over-fit for  $M > 7$ .

Since in the performance of the MBRVQ classifier peaks at  $M=7$  (shown in Figure 38), its classification performance at  $M = 7$  is also calculated over a range of Markov order from 0 to  $P-1 = 7$ , where  $P = 8$  stages. The classification performance in terms of the overall accuracy is shown in Figure 39. The overall accuracy for 1-NN classifier, which is 94.12 %, is also shown for comparison. It can be seen that MBRVQ classifier tends to converge from the fourth Markov order, when the overall accuracy is

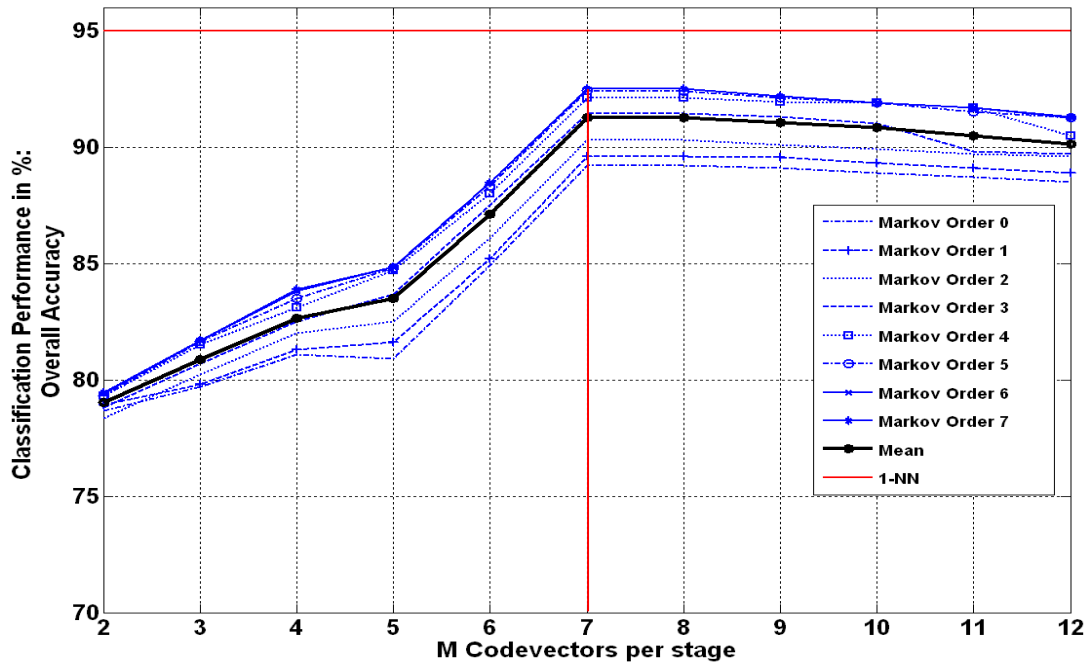


Figure 38. RVQ classification performance for  $M = \{2,3,4,5,6,7,8,9,10,11,12\}$ , and  $P=8$ : Mean overall accuracy averaged over Markov orders 0 to 7.

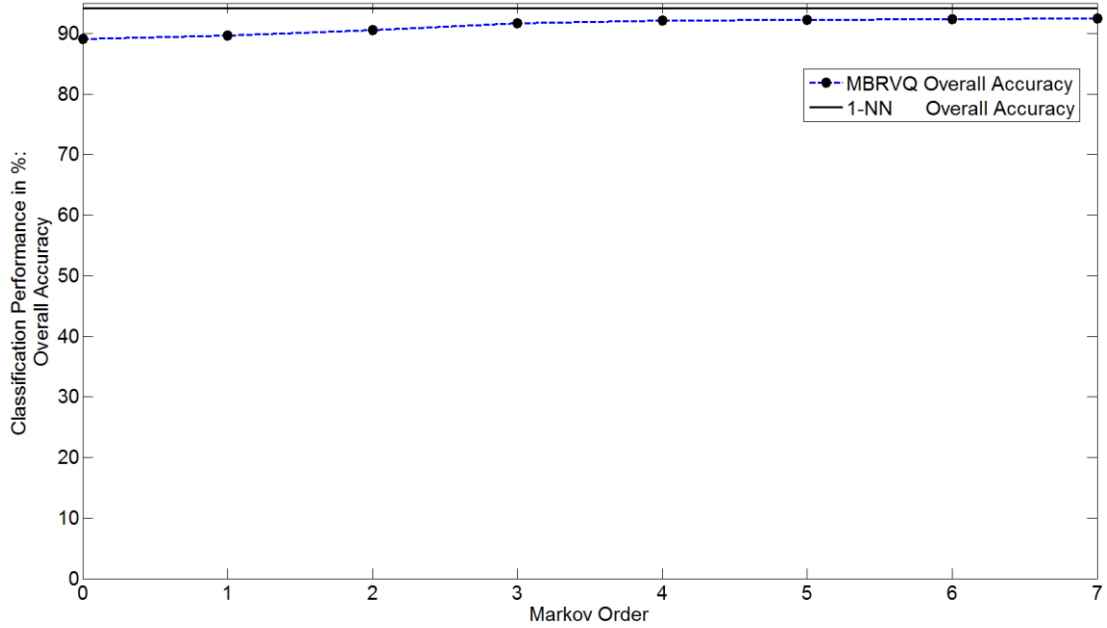


Figure 39. RVQ classification performance for  $M = 7$  and  $P=8$ : overall accuracy for Markov order =  $\{0, 1, 2, 3, 4, 5, 6, 7\}$ .

92.11 %. The overall accuracy of MBRVQ classifier stabilizes to its apparent limit value of 92.5 % for *sixth* and *seventh* Markov orders. For this experiment, the best classification performance of the MBRVQ classifier is within 3% of the performance of 1-NN classifier. This performance measure of MBRVQ is in line with results of different classifiers reported in [49].

The comparison of computational and memory costs between 1-NN and MBRVQ classifiers are shown in Figure 40. MBRVQ classifier offers a great deal of cost savings. It can be observed in Figure 40 that MBRVQ classifier, with  $M=7$  and  $P=8$ , approaches its best performance from the 4<sup>th</sup> Markov order. The overall classification accuracy is 92.11%, which is within 3% of the performance of 1-NN classifier. For the dimensionality  $k = 28 \times 28$  of the handwritten digit image, the memory cost of the MBRVQ classifier is 0.74 MB, approximately, as compared to 1-NN classifier's memory cost that is 45 MB, approximately. Similarly, for Euclidean distance used as the measure

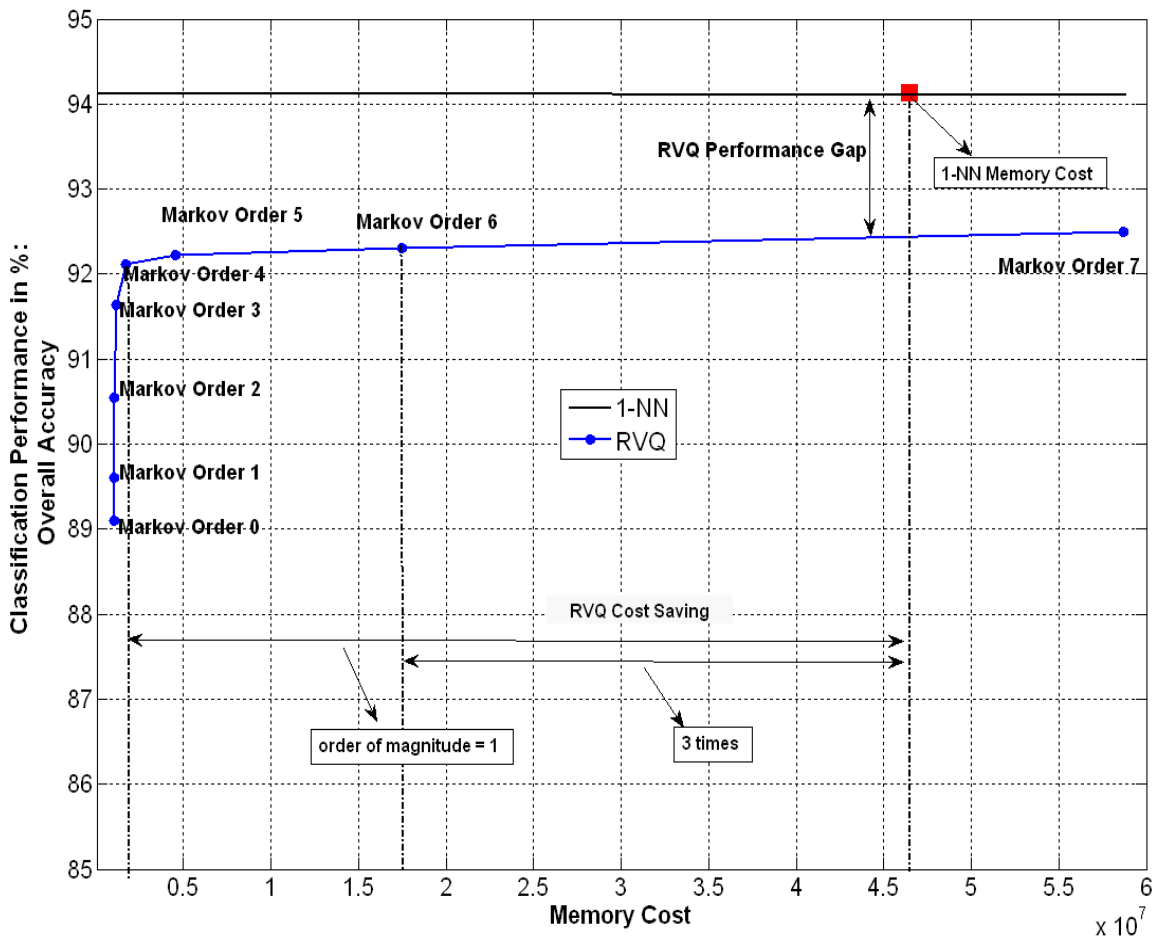
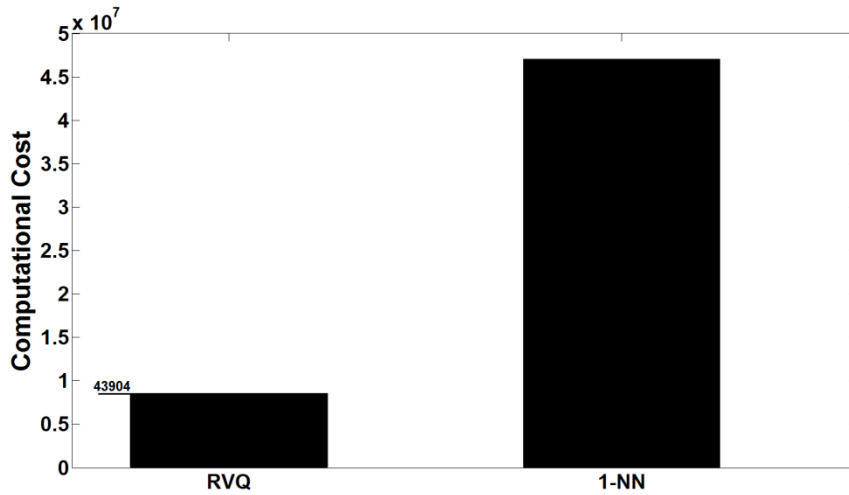


Figure 40. RVQ-versus-1NN: (Top) Computational cost. (Bottom) Memory cost.

of nearness, the computational cost of MBRVQ classifier is  $kMP = 43904$  multiplications and additions, as compared to the computational cost of 1-NN classifier that is  $k|T| + |T| = 47,040,000$  multiplications and additions, where  $|T|$  is the size of the training set. In other words, for the given dimensionality of the input vector of a handwritten digit  $k = 28 \times 28 = 784$ , the number of searches from MBRVQ classifier is  $MP = 56$ , whereas the number of searches for the 1-NN classifier is  $|T| = 60,000$ .

## CHAPTER 7

### CONCLUSION AND FUTURE RESEARCH

#### Conclusion

Residual vector quantization (RVQ) is a 1-nearest neighbor (1-NN) type of technique. RVQ is a multi-stage implementation of regular vector quantization. An input is successively quantized to the nearest codevector in each stage codebook. In classification, nearest neighbor techniques are very attractive since these techniques very accurately model the ideal Bayes class boundaries. However, nearest neighbor classification techniques require a large size of representative dataset. Since in such techniques a test input is assigned a class membership after an exhaustive search the entire training set, a reasonably large training set can make the implementation cost of the nearest neighbor classifier unfeasibly costly. Although, the k-d tree structure [52] offers a far more efficient implementation of 1-NN search, however, the cost of storing the data points can become prohibitive, especially in higher dimensionality.

However, RVQ also offers a nice solution to a cost-effective implementation of 1-NN-based classification. Because of the direct-sum structure of the RVQ codebook, the memory and computational of cost 1-NN-based system is greatly reduced. For example, RVQ codebook with  $M = 4$  codevectors-per-stage and  $P = 8$  stages can potentially represent  $M^P = 65536$  training vectors with the cost of only  $MP = 32$  codevectors. Although, as compared to an equivalent 1-NN system, the multi-stage implementation of the RVQ codebook compromises the accuracy of the class boundaries, yet the classification error has been empirically shown to be within 3% to 4% of the performance of an equivalent 1-NN-based classifier.

RVQ is an image template-based matching technique. Therefore, it uses Euclidean distance as a nearness measure for matching an input to the nearest codevector. As a result, the images RVQ can be applied on have to be very controlled with minimal variations. Moreover, since RVQ is a 1-NN-based technique, the size of the training set is recommended to be large so that the codebook is a good representation of the data. Subsequently, the class boundaries between the data points of different classes will also accurately defined by the RVQ.

### **Future Research**

As pointed out earlier, RVQ is currently constrained to use only Euclidean distance measure. As a result, RVQ-based classification is not robust to variations in images. To address this issue, it is proposed that RVQ codebook design be investigated for other distance functions so that RVQ can be used on feature templates composed of more robust features like scale-invariant feature transform (SIFT). As a result, a RVQ-based classifier will become more robust and will become suited to applications of classification, such as object detection and recognition, on a wide variety of images.

### Appendix A

Table A.1. Class 1 = Heavy ; Class 2 = Sports, Training set size = 300 sample-per-class, total 600 samples. Test set size = 270 total samples, Input size = 150x250 grayscale pixels,  $M = 4$ ,  $P = 8$ .

<b>MBRVQ , CostERoE</b>																											
<b>Threshold</b>	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>				
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2			
	1	74	54	128	80	45	125	82	33	115	80	21	101	80	11	91	80	13	93	83	15	98	82	18	100		
<b>= 0.0</b>	2	16	126	142	10	135	145	8	147	155	10	159	169	10	169	179	10	167	177	7	165	172	8	162	170		
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270			
	Prod Acc %	82.2	70	70	88.9	75	76.5	91.1	81.7	81.2	88.9	88.3	84.1	88.9	94	88.4	88.9	92.8	87.5	92.2	91.7	88.5	91.1	90	86.6		
	User Acc %	57.8	88.7	73.3	64	93.1	78.6	71.3	94.8	83.1	79.2	94.1	86.7	87.9	94	91.2	86	94.4	90.2	84.7	95.9	90.3	82	95.3	88.7		
	Overall Acc %	74.07			79.63			84.81			88.52			92.22			91.48			91.85			90.37				
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2
<b>= 0.1</b>	1	55	83	138	43	83	126	60	66	126	80	22	102	82	14	96	78	14	92	80	13	93	79	16	95		
	2	35	97	132	47	97	144	30	114	144	10	158	168	8	166	174	12	166	178	10	167	177	11	164	175		
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270			
	Prod Acc %	61.1	53.9	50.5	47.8	53.9	41	66.7	63.3	57.2	88.9	87.8	83.7	91.1	92.2	88.3	86.7	92.2	85.8	88.9	92.8	87.5	87.8	91.1	85.5		
	User Acc %	39.9	73.5	56.7	34.1	67.4	50.8	47.6	79.2	63.4	78.4	94	86.2	85.4	95.4	90.4	84.8	93.3	89.1	86	94.4	90.2	83.2	93.7	88.5		
	Overall Acc %	56.3			51.85			64.44			88.15			91.85			90.37			91.48			90.00				
<b>= 0.2</b>	1	75	53	128	80	44	124	82	32	114	80	18	98	82	13	95	82	14	96	83	15	98	83	15	98		
	2	15	127	142	10	136	146	8	148	156	10	162	172	8	167	175	8	166	174	7	165	172	7	165	172		
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270			
	Prod Acc %	83.3	70.6	71	88.9	75.6	77	91.1	82.2	81.5	88.9	90	85.3	91.1	93	88.7	91.1	92.2	88.3	92.2	91.7	88.5	92.2	91.7	88.5		
	User Acc %	58.6	89.4	74	64.5	93.2	79	71.9	94.9	83.4	81.6	94.2	87.9	86.3	95	90.9	85.4	95.4	90.4	84.7	95.9	90.3	84.7	95.9	90.3		
	Overall Acc %	74.81			80.00			85.19			89.63			92.22			91.85			91.85			91.85				
<b>= 0.3</b>	1	40	101	141	43	89	132	65	54	119	81	25	106	80	16	96	78	11	89	79	12	91	77	12	89		
	2	50	79	129	47	91	138	25	126	151	9	155	164	10	164	174	12	169	181	11	168	179	13	168	181		
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270			
	Prod Acc %	44.4	43.9	36.4	47.8	50.6	40.2	72.2	70	63.4	90	86.1	83.2	88.9	91.1	86.1	86.7	93.9	87.2	87.8	93.3	87.3	85.6	93.3	86.1		
	User Acc %	28.4	61.2	44.8	32.6	65.9	49.3	54.6	83.4	69	76.4	94.5	85.5	83.3	94.3	88.8	87.6	93.4	90.5	86.8	93.9	90.4	86.5	92.8	89.7		
	Overall Acc %	44.07			49.63			70.74			87.41			90.37			91.48			91.48			90.74				

<b>Threshold = 0.4</b>	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2				
	1	40	101	141	43	89	132	65	54	119	81	25	106	80	16	96	78	11	89	79	12	91	77	12	89
	2	50	79	129	47	91	138	25	126	151	9	155	164	10	164	174	12	169	181	11	168	179	13	168	181
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
	Prod Acc %	44.4	43.9	36.4	47.8	50.6	40.2	72.2	70	63.4	90	86.1	83.2	88.9	91.1	86.1	86.7	93.9	87.2	87.8	93.3	87.3	85.6	93.3	86.1
	User Acc %	28.4	61.2	44.8	32.6	65.9	49.3	54.6	83.4	69	76.4	94.5	85.5	83.3	94.3	88.8	87.6	93.4	90.5	86.8	93.9	90.4	86.5	92.8	89.7
Overall Acc %	44.07			49.63			70.74			87.41			90.37			91.48			91.48			90.74			
<b>Threshold = 0.5</b>	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	40	101	141	43	89	132	65	54	119	81	25	106	80	16	96	78	11	89	79	12	91	77	12	89
	2	50	79	129	47	91	138	25	126	151	9	155	164	10	164	174	12	169	181	11	168	179	13	168	181
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
	Prod Acc %	44.4	43.9	36.4	47.8	50.6	40.2	72.2	70	63.4	90	86.1	83.2	88.9	91.1	86.1	86.7	93.9	87.2	87.8	93.3	87.3	85.6	93.3	86.1
	User Acc %	28.4	61.2	44.8	32.6	65.9	49.3	54.6	83.4	69	76.4	94.5	85.5	83.3	94.3	88.8	87.6	93.4	90.5	86.8	93.9	90.4	86.5	92.8	89.7
Overall Acc %	44.07			49.63			70.74			87.41			90.37			91.48			91.48			90.74			

<b>Threshold = 0.6</b>	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
		Prod Rule, *My RoE			Prod Rule, *My RoE			Prod Rule, *My RoE			Prod Rule, *My RoE			Prod Rule, *My RoE			Prod Rule, *My RoE			Prod Rule, *My RoE			Prod Rule, *My RoE		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	74	54	128	80	45	125	82	33	115	80	21	101	80	11	91	80	13	93	83	15	98	82	18	100
	2	16	126	142	10	135	145	8	147	155	10	159	169	10	169	179	10	167	177	7	165	172	8	162	170
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
Prod Acc %	82.2	70	70	88.9	75	76.5	91.1	81.7	81.2	88.9	88.3	84.1	88.9	94	88.4	88.9	92.8	87.5	92.2	91.7	88.5	91.1	90	86.6	
User Acc %	57.8	88.7	73.3	64	93.1	78.6	71.3	94.8	83.1	79.2	94.1	86.7	87.9	94	91.2	86	94.4	90.2	84.7	95.9	90.3	82	95.3	88.7	
Overall Acc %	74.07			79.63			84.81			88.52			92.22			91.48			91.85			90.37			
<b>Threshold = 0.7</b>	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	55	83	138	43	83	126	60	66	126	80	22	102	82	14	96	78	14	92	80	13	93	79	16	95
	2	35	97	132	47	97	144	30	114	144	10	158	168	8	166	174	12	166	178	10	167	177	11	164	175
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
	Prod Acc %	61.1	53.9	50.5	47.8	53.9	41	66.7	63.3	57.2	88.9	87.8	83.7	91.1	92.2	88.3	86.7	92.2	85.8	88.9	92.8	87.5	87.8	91.1	85.5
	User Acc %	39.9	73.5	56.7	34.1	67.4	50.8	47.6	79.2	63.4	78.4	94	86.2	85.4	95.4	90.4	84.8	93.3	89.1	86	94.4	90.2	83.2	93.7	88.5
Overall Acc %	56.3			51.85			64.44			88.15			91.85			90.37			91.48			90.00			
<b>Threshold = 0.8</b>	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	75	53	128	80	44	124	82	32	114	80	18	98	82	13	95	82	14	96	83	15	98	83	15	98
	2	15	127	142	10	136	146	8	148	156	10	162	172	8	167	175	8	166	174	7	165	172	7	165	172
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
Prod	83.3	70.6	71	88.9	75.6	77	91.	82.2	81.5	88.9	90	85.3	91.1	93	88.7	91.1	92.2	88.3	92.2	91.7	88.5	92.2	91.7	88.5	



	Acc %							1																	
	User Acc %	58.6	89.4	74	64.5	93.2	79	71.9	94.9	83.4	81.6	94.2	87.9	86.3	95	90.9	85.4	95.4	90.4	84.7	95.9	90.3	84.7	95.9	90.3
	Overall Acc %	74.81			80.00			85.19			89.63			92.22			91.85			91.85			91.85		
Threshold = 0.9	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	40	101	141	43	89	132	65	54	119	81	25	106	80	16	96	78	11	89	79	12	91	77	12	89
	2	50	79	129	47	91	138	25	126	151	9	155	164	10	164	174	12	169	181	11	168	179	13	168	181
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
	Prod Acc %	44.4	43.9	36.4	47.8	50.6	40.2	72.2	70	63.4	90	86.1	83.2	88.9	91.1	86.1	86.7	93.9	87.2	87.8	93.3	87.3	85.6	93.3	86.1
	User Acc %	28.4	61.2	44.8	32.6	65.9	49.3	54.6	83.4	69	76.4	94.5	85.5	83.3	94.3	88.8	87.6	93.4	90.5	86.8	93.9	90.4	86.5	92.8	89.7
Overall Acc %	44.07			49.63			70.74			87.41			90.37			91.48			91.48			90.74			

Table A.2. Class 1 = Heavy ; Class 2 = Sports, Training set size = 300 sample-per-class, total 600 samples  
 Test set size = 270 total samples, Input size = 71x101 grayscale pixels,  $M = 2$ ,  $P = 16$ .

MBRVQ, CostERoE																									
Threshold = 0.0	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	79	57	136	79	62	141	80	53	133	82	51	133	72	33	105	73	27	100	81	19	100	82	17	99
	2	11	123	134	11	118	129	10	127	137	8	129	137	18	147	165	17	153	170	9	161	170	8	163	171
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
	Prod Acc %	87.8	68.3	78.1	87.8	65.6	76.7	88.9	70.6	79.8	91.1	71.7	81.4	80	81.7	80.9	81.1	85	83.1	90	89.4	89.7	91.1	90.6	90.9
	User Acc %	58.1	91.8	75	56	91.5	73.8	60.2	92.7	76.5	61.7	94.2	78	68.6	89.1	78.9	73	90	81.5	81	94.7	87.9	82.8	95.3	89.1
	Overall Acc %	74.81			72.96			76.67			78.15			81.11			83.7			89.63			90.74		
	Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
1	83	14	97	85	13	98	83	13	96	85	14	99	86	14	100	86	14	100	86	14	100	86	14	100	
2	7	166	173	5	167	172	7	167	174	5	166	171	4	166	170	4	166	170	4	166	170	4	166	170	
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	
Prod Acc %	92.2	92.2	92.2	94.4	92.8	93.6	92.2	92.8	92.5	94.4	92.2	93.3	95.6	92.2	93.9	95.6	92.2	93.9	95.6	92.2	93.9	95.6	92.2	93.9	
User Acc %	85.6	96	90.8	86.7	97.1	91.9	86.5	96	91.3	85.9	97.1	91.5	86	97.6	91.8	86	97.6	91.8	86	97.6	91.8	86	97.6	91.8	
Overall Acc %	92.22			93.33			92.59			92.96			93.33			93.33			93.33			93.33			
Threshold = 0.3 to 0.5	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	79	57	136	79	62	141	80	53	133	82	51	133	72	33	105	73	27	100	81	19	100	82	17	99
	2	11	123	134	11	118	129	10	127	137	8	129	137	18	147	165	17	153	170	9	161	170	8	163	171
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
	Prod Acc %	87.8	68.3	78.1	87.8	65.6	76.7	88.9	70.6	79.8	91.1	71.7	81.4	80	81.7	80.9	81.1	85	83.1	90	89.4	89.7	91.1	90.6	90.9
	User Acc %	58.1	91.8	75	56	91.5	73.8	60.2	92.7	76.5	61.7	94.2	78	68.6	89.1	78.9	73	90	81.5	81	94.7	87.9	82.8	95.3	89.1
	Overall Acc %	74.81			72.96			76.67			78.15			81.11			83.7			89.63			90.74		
	Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
1	83	14	97	85	13	98	83	13	96	85	14	99	86	14	100	86	14	100	86	14	100	86	14	100	
2	7	166	173	5	167	172	7	167	174	5	166	171	4	166	170	4	166	170	4	166	170	4	166	170	
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	
Prod Acc %	92.2	92.2	92.2	94.4	92.8	93.6	92.2	92.8	92.5	94.4	92.2	93.3	95.6	92.2	93.9	95.6	92.2	93.9	95.6	92.2	93.9	95.6	92.2	93.9	
User Acc %	85.6	96	90.8	86.7	97.1	91.9	86.5	96	91.3	85.9	97.1	91.5	86	97.6	91.8	86	97.6	91.8	86	97.6	91.8	86	97.6	91.8	
Overall Acc %	92.22			93.33			92.59			92.96			93.33			93.33			93.33			93.33			

MBRVQ, CostERoE RoE																									
Threshold = 0.55	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	83	69	152	77	57	134	79	56	135	81	48	129	81	36	117	76	33	109	83	24	107	84	16	100
	2	7	111	118	13	123	136	11	124	135	9	132	141	9	144	153	14	147	161	7	156	163	6	164	170
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
	Prod Acc %	92.2	61.7	77	85.6	68.3	77	87.8	68.9	78.4	90	73.3	81.7	90	80	85	84.4	81.7	83.1	92.2	86.7	89.5	93.3	91.1	92.2
	User Acc %	54.6	94.1	74.4	57.5	90.4	74	58.5	91.9	75.2	62.8	93.6	78.2	69.2	94.1	81.7	69.7	91.3	80.5	77.6	95.7	86.7	84	96.5	90.3
	Overall Acc %	71.85			74.07			75.19			78.89			83.33			82.59			88.52			91.85		
	Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
1	84	13	97	85	15	100	84	12	96	80	11	91	80	11	91	80	9	89	79	8	87	79	7	86	
2	5	166	171	5	165	170	4	166	170	4	165	169	4	165	169	4	167	171	4	167	171	4	166	170	
	89	179	268	90	180	270	88	178	266	84	176	260	84	176	260	84	176	260	83	175	258	83	173	256	
Prod Acc %				94.4	91.7	93.1																			
User Acc %				85	97.1	91.1																			
Overall Acc %	----			92.59			----			----			----			----			----			----			
Threshold = 0.6	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	82	76	158	80	65	145	74	50	124	78	44	122	80	32	112	75	30	105	83	19	102	84	17	101
	2	8	104	112	10	115	125	16	130	146	12	136	148	10	148	158	15	148	163	7	161	168	6	162	168
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	178	268	90	180	270	90	179	269
	Prod Acc %	91.1	57.8	74.5	88.9	63.9	76.4	82.2	72.2	77.2	86.7	75.6	81.2	88.9	82.2	85.6	83.3	83.1	83.2	92.2	89.4	90.8	93.3	90.5	91.9
	User Acc %	51.9	92.9	72.4	55.2	92	73.6	59.7	89	74.4	63.9	91.9	77.9	71.4	93.7	82.6	71.4	90.8	81.1	81.4	95.8	88.6	83.2	96.4	89.8
	Overall Acc %	68.89			72.22			75.56			79.26			84.44			83.21			90.37			91.45		
	Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
1	84	13	97	86	16	102	84	12	96	80	10	90	80	11	91	80	10	90	79	8	87	79	7	86	
2	5	166	171	4	164	168	4	166	170	4	166	170	4	165	169	4	166	170	4	167	171	4	166	170	
	89	179	268	90	180	270	88	178	266	84	176	260	84	176	260	84	176	260	83	175	258	83	173	256	
Prod Acc %	94.4	92.7	93.6	95.6	91.1	93.4	95.5	93.3	94.4	95.2	94.3	94.8	95.2	93.8	94.5	95.2	94.3	94.8	95.2	95.4	95.3	95.2	96	95.6	
User Acc %	86.6	97.1	91.9	84.3	97.6	91	87.5	97.6	92.6	88.9	97.6	93.3	87.9	97.6	92.8	88.9	97.6	93.3	90.8	97.7	94.3	91.9	97.6	94.8	
Overall Acc %	93.28			92.59			93.98			94.62			94.23			94.62			95.35			95.7			
MBRVQ, CostERoE RoE																									
Threshold	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	68	38	106	58	38	96	67	33	100	80	45	125	76	28	104	75	28	103	79	19	98	79	15	94

<b>= 0.7</b>	<b>2</b>	0	0	0	20	111	131	17	132	149	9	132	141	13	146	159	8	146	154	7	157	164	5	160	165
		68	38	106	78	149	227	84	165	249	89	177	266	89	174	263	83	174	257	86	176	262	84	175	259
	<b>Prod Acc %</b>				74.4	74.5	74.5	79.8	80	79.9	89.9	74.6	82.3	85.4	83.9	84.7	90.4	83.9	87.2	91.9	89.2	90.6	94	91.4	92.7
	<b>User Acc %</b>				60.4	84.7	72.6	67	88.6	77.8	64	93.6	78.8	73.1	91.8	82.5	72.8	94.8	83.8	80.6	95.7	88.2	84	97	90.5
	<b>Overall Acc %</b>	---			74.45			79.92			79.7			84.41			85.99			90.08			92.28		
	<b>Markov</b>	<b>8<sup>th</sup></b>			<b>9<sup>th</sup></b>			<b>10<sup>th</sup></b>			<b>11<sup>th</sup></b>			<b>12<sup>th</sup></b>			<b>13<sup>th</sup></b>			<b>14<sup>th</sup></b>			<b>15<sup>th</sup></b>		
	<b>Classes</b>	<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>	
	<b>1</b>	79	15	94	77	12	89	77	10	87	77	10	87	77	9	86	77	7	84	76	7	83	76	7	83
	<b>2</b>	5	160	165	4	163	167	4	164	168	4	162	166	4	160	164	4	161	165	4	161	165	4	161	165
		84	175	259	81	175	256	81	174	255	81	172	253	81	169	250	81	168	249	80	168	248	80	168	248
<b>Prod Acc %</b>	94	91.4	92.7	95.1	93.1	94.1	95.1	94.3	94.7	95.1	94.2	94.7	95.1	94.7	94.9	95.1	95.8	95.5	95	95.8	95.4	95	95.8	95.4	
<b>User Acc %</b>	84	97	90.5	86.5	97.6	92.1	88.5	97.6	93.1	88.5	97.6	93.1	89.5	97.6	93.6	91.7	97.6	94.7	91.6	97.6	94.6	91.6	97.6	94.6	
<b>Overall Acc %</b>	92.28			93.75			94.51			94.47			94.8			95.58			95.56			95.56			
<b>Threshold = 0.8</b>	<b>Markov</b>	<b>0<sup>th</sup></b>			<b>1<sup>st</sup></b>			<b>2<sup>nd</sup></b>			<b>3<sup>rd</sup></b>			<b>4<sup>th</sup></b>			<b>5<sup>th</sup></b>			<b>6<sup>th</sup></b>			<b>7<sup>th</sup></b>		
	<b>Classes</b>	<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>	
	<b>1</b>	0	0	0	0	0	0	18	11	29	75	30	105	75	29	104	74	24	98	75	14	89	77	11	88
	<b>2</b>	0	0	0	0	0	0	0	0	0	8	108	116	10	117	127	6	140	146	7	159	166	6	160	166
		0	0	0	0	0	0	18	11	29	83	138	221	85	146	231	80	164	244	82	173	255	83	171	254
	<b>Prod Acc %</b>										90.4	78.3	84.4	88.2	80.1	84.2	92.5	85.4	89	91.5	91.9	91.7	92.8	93.6	93.2
	<b>User Acc %</b>										71.4	93.1	82.3	72.1	92.1	82.1	75.5	95.9	85.7	84.3	95.8	90.1	87.5	96.4	92
	<b>Overall Acc %</b>										82.81			83.12			87.7			91.76			93.31		
	<b>Markov</b>	<b>8<sup>th</sup></b>			<b>9<sup>th</sup></b>			<b>10<sup>th</sup></b>			<b>11<sup>th</sup></b>			<b>12<sup>th</sup></b>			<b>13<sup>th</sup></b>			<b>14<sup>th</sup></b>			<b>15<sup>th</sup></b>		
	<b>Classes</b>	<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>	
<b>1</b>	77	11	88	77	10	87	76	9	85	76	8	84	76	8	84	76	8	84	76	8	84	76	8	84	
<b>2</b>	3	157	160	4	159	163	5	161	166	4	160	164	4	160	164	4	160	164	4	160	164	4	160	164	
	80	168	248	81	169	250	81	170	251	80	168	248	80	168	248	80	168	248	80	168	248	80	168	248	
<b>Prod Acc %</b>	96.3	93.5	94.9	95.1	94.1	94.6	93.8	94.7	94.3	95	95.2	95.1	95	95.2	95.1	95	95.2	95.1	95	95.2	95.1	95	95.2	95.1	
<b>User Acc %</b>	87.5	98.1	92.8	88.5	97.5	93	89.4	97	93.2	90.5	97.6	94.1	90.5	97.6	94.1	90.5	97.6	94.1	90.5	97.6	94.1	90.5	97.6	94.1	
<b>Overall Acc %</b>	94.35			94.4			94.42			95.16			95.16			95.16			95.16			95.16			
<b>MBRVQ, CostERoE RoE</b>																									
<b>Threshold = 0.9</b>	<b>Markov</b>	<b>0<sup>th</sup></b>			<b>1<sup>st</sup></b>			<b>2<sup>nd</sup></b>			<b>3<sup>rd</sup></b>			<b>4<sup>th</sup></b>			<b>5<sup>th</sup></b>			<b>6<sup>th</sup></b>			<b>7<sup>th</sup></b>		
	<b>Classes</b>	<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>	
	<b>1</b>	0	0	0	0	0	0	0	0	0	12	1	13	44	5	49	60	9	69	66	7	73	69	7	76
	<b>2</b>	0	0	0	0	0	0	0	0	0	0	0	0	4	78	82	6	105	111	5	127	132	5	147	152
		0	0	0	0	0	0	0	0	0	12	1	13	48	83	131	66	114	180	71	134	205	74	154	228
	<b>Prod Acc %</b>													91.7	94	92.9	90.9	92.1	91.5	93	94.8	93.9	93.2	95.5	94.4
<b>User Acc %</b>													89.8	95.1	92.5	87	94.6	90.8	90.4	96.2	93.3	90.8	96.7	93.8	

	Overall Acc %	----			----			----			----			93.13			91.67			94.15			94.74		
	Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	74	7	81	75	7	82	75	7	82	75	7	82	75	7	82	73	7	80	73	7	80	73	7	80
	2	3	152	155	4	157	161	4	158	162	4	158	162	4	160	164	4	161	165	4	161	165	4	161	165
		77	159	236	79	164	243	79	165	244	79	165	244	79	167	246	77	168	245	77	168	245	77	168	245
	Prod Acc %	96.1	95.6	95.9	94.9	95.7	95.3	94.9	95.8	95.4	94.9	95.8	95.4	94.9	95.8	95.4	94.8	95.8	95.3	94.8	95.8	95.3	94.8	95.8	95.3
	User Acc %	91.4	98.1	94.8	91.5	97.5	94.5	91.5	97.5	94.5	91.5	97.5	94.5	91.5	97.6	94.6	91.3	97.6	94.5	91.3	97.6	94.5	91.3	97.6	94.5
Overall Acc %	95.76			95.47			95.49			95.49			95.53			95.51			95.51			95.51			
Threshold = 1	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	0	0	0	0	0	0	0	0	0	5	0	5	14	2	16	44	7	51	62	7	69	65	7	72
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	0	26	26	2	99	101	3	145	148
		0	0	0	0	0	0	0	0	0	5	0	5	14	5	19	44	33	77	64	106	170	68	152	220
	Prod Acc %													100	60	80	100	78.8	89.4	96.9	93.4	95.2	95.6	95.4	95.5
	User Acc %													87.5	100	93.8	86.3	100	93.2	89.9	98	94	90.3	98	94.2
	Overall Acc %	----			----			----			----			89.47			90.91			94.71			95.45		
	Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	70	7	77	72	7	79	72	7	79	72	7	79	72	7	79	72	7	79	72	7	79	72	7	79
	2	3	152	155	4	157	161	4	158	162	4	158	162	4	161	165	4	161	165	4	161	165	4	161	165
		73	159	232	76	164	240	76	165	241	76	165	241	76	168	244	76	168	244	76	168	244	76	168	244
	Prod Acc %	95.9	95.6	95.8	94.7	95.7	95.2	94.7	95.8	95.3	94.7	95.8	95.3	94.7	95.8	95.3	94.7	95.8	95.3	94.7	95.8	95.3	94.7	95.8	95.3
User Acc %	90.9	98.1	94.5	91.1	97.5	94.3	91.1	97.5	94.3	91.1	97.5	94.3	91.1	97.6	94.4	91.1	97.6	94.4	91.1	97.6	94.4	91.1	97.6	94.4	
Overall Acc %	95.69			95.42			95.44			95.44			95.49			95.49			95.49			95.49			
<b>Feature-Count Rule, CostERoE RoE</b>																									
Threshold = 0.0	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	40	101	141	43	89	132	40	96	136	57	86	143	51	77	128	69	80	149	76	44	120	80	18	98
	2	50	79	129	47	91	138	50	84	134	33	94	127	39	103	142	21	100	121	14	136	150	10	162	172
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
	Prod Acc %	44.4	43.9	44.2	47.8	50.6	49.2	44.4	46.7	45.6	63.3	52.2	57.8	56.7	57.2	57	76.7	55.6	66.2	84.4	75.6	80	88.9	90	89.5
	User Acc %	28.4	61.2	44.8	32.6	65.9	49.3	29.4	62.7	46.1	39.9	74	57	39.8	72.5	56.2	46.3	82.6	64.5	63.3	90.7	77	81.6	94.2	87.9
	Overall Acc %	44.07			49.63			45.93			55.93			57.04			62.59			78.52			89.63		
	Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
1	77	16	93	81	14	95	79	14	93	76	14	90	79	15	94	80	17	97	79	12	91	80	12	92	

	2	13	164	177	9	166	175	11	166	177	14	166	180	11	165	176	10	163	173	11	168	179	10	168	178
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
	Prod Acc %	85.6	91.1	88.4	90	92.2	91.1	87.8	92.2	90	84.4	92.2	88.3	87.8	91.7	89.8	88.9	90.6	89.8	87.8	93.3	90.6	88.9	93.3	91.1
	User Acc %	82.8	92.7	87.8	85.3	94.9	90.1	84.9	93.8	89.4	84.4	92.2	88.3	84	93.8	88.9	82.5	94.2	88.4	86.8	93.9	90.4	87	94.4	90.7
	Overall Acc %	89.26			91.48			90.74			89.63			90.37			90.00			91.48			91.85		
Threshold = 0.3	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	77	109	186	75	58	133	78	47	125	78	37	115	78	41	119	76	37	113	79	26	105	80	21	101
	2	13	71	84	15	122	137	12	133	145	12	143	155	12	139	151	14	143	157	11	154	165	10	159	169
		90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
	Prod Acc %	85.6	39.4	62.5	83.3	67.8	75.6	86.7	73.9	80.3	86.7	79.4	83.1	86.7	77.2	82	84.4	79.4	81.9	87.8	85.6	86.7	88.9	88.3	88.6
	User Acc %	41.4	84.5	63	56.4	89.1	72.8	62.4	91.7	77.1	67.8	92.3	80.1	65.5	92.1	78.8	67.3	91.1	79.2	75.2	93.3	84.3	79.2	94.1	86.7
	Overall Acc %	54.81			72.96			78.15			81.85			80.37			81.11			86.3			88.52		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	77	18	95	80	15	95	83	14	97	80	15	95	79	14	93	80	14	94	81	11	92	80	12	92
	2	13	162	175	10	165	175	7	166	173	10	165	175	11	166	177	10	166	176	9	169	178	10	168	178
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	
Prod Acc %	85.6	90	87.8	88.9	91.7	90.3	92.2	92.2	92.2	88.9	91.7	90.3	87.8	92.2	90	88.9	92.2	90.6	90	93.9	92	88.9	93.3	91.1	
User Acc %	81.1	92.6	86.9	84.2	94.3	89.3	85.6	96	90.8	84.2	94.3	89.3	84.9	93.8	89.4	85.1	94.3	89.7	88	94.9	91.5	87	94.4	90.7	
Overall Acc %	88.52			90.74			92.22			90.74			90.74			91.11			92.59			91.85			

**Feature-Count Rule, CostERoE RoE**

Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	
	1	86	131	217	79	75	154	76	47	123	76	46	122	78	38	116	81	39	120	80	31	111	82	25
2	4	49	53	11	105	116	14	133	147	14	134	148	12	142	154	9	141	150	10	149	159	8	155	163
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
Prod Acc %	95.6	27.2	61.4	87.8	58.3	73.1	84.4	73.9	79.2	84.4	74.4	79.4	86.7	78.9	82.8	90	78.3	84.2	88.9	82.8	85.9	91.1	86.1	88.6
User Acc %	39.6	92.5	66.1	51.3	90.5	70.9	61.8	90.5	76.2	62.3	90.5	76.4	67.2	92.2	79.7	67.5	94	80.8	72.1	93.7	82.9	76.6	95.1	85.9
Overall Acc %	50.00			68.15			77.41			77.78			81.48			82.22			84.81			87.78		
Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	
	1	81	21	102	80	18	98	81	17	98	81	11	92	81	15	96	82	13	95	83	11	94	83	13
2	9	159	168	10	162	172	9	163	172	9	169	178	9	165	174	8	167	175	7	169	176	7	167	174
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
Prod Acc %	90	88.3	89.2	88.9	90	89.5	90	90.6	90.3	90	93.9	92	90	91.7	90.9	91.1	92.8	92	92.2	93.9	93.1	92.2	92.8	92.5
User Acc %	79.4	94.6	87	81.6	94.2	87.9	82.7	94.8	88.8	88	94.9	91.5	84.4	94.8	89.6	86.3	95.4	90.9	88.3	96	92.2	86.5	96	91.3
Overall Acc %	88.89			89.63			90.37			92.59			91.11			92.22			93.33			92.59		
Markov	16 <sup>th</sup>			17 <sup>th</sup>			18 <sup>th</sup>			19 <sup>th</sup>			20 <sup>th</sup>			21 <sup>th</sup>			22 <sup>th</sup>			23 <sup>th</sup>		
	Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	
	1	84	84	168	81	60	141	78	58	136	78	46	124	79	52	131	82	38	120	81	32	113	82	27
2	6	96	102	9	120	129	12	122	134	12	134	146	11	128	139	8	142	150	9	148	157	8	153	161
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
Prod Acc %	93.3	53.3	73.3	90	66.7	78.4	86.7	67.8	77.3	86.7	74.4	80.6	87.8	71.1	79.5	91.1	78.9	85	90	82.2	86.1	91.1	85	88.1
User Acc %	50	94.1	72.1	57.4	93	75.2	57.4	91	74.2	62.9	91.8	77.4	60.3	92.1	76.2	68.3	94.7	81.5	71.7	94.3	83	75.2	95	85.1
Overall Acc %	66.67			74.44			74.07			78.52			76.67			82.96			84.81			87.04		
Markov	24 <sup>th</sup>			25 <sup>th</sup>			26 <sup>th</sup>			27 <sup>th</sup>			28 <sup>th</sup>			29 <sup>th</sup>			30 <sup>th</sup>			31 <sup>th</sup>		
	Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	
	1	84	22	106	85	19	104	83	20	103	82	16	98	85	14	99	83	13	96	83	12	95	82	10
2	6	158	164	5	161	166	7	160	167	8	164	172	5	166	171	7	167	174	7	168	175	8	170	178
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270
Prod Acc %	93.3	87.8	90.6	94.4	89.4	91.9	92.2	88.9	90.6	91.1	91.1	91.1	94.4	92.2	93.3	92.2	92.8	92.5	92.2	93.3	92.8	91.1	94.4	92.8
User Acc %	79.2	96.3	87.8	81.7	97	89.4	80.6	95.8	88.2	83.7	95.3	89.5	85.9	97.1	91.5	86.5	96	91.3	87.4	96	91.7	89.1	95.5	92.3
Overall Acc %	89.63			91.11			90.00			91.11			92.96			92.59			92.96			93.33		

Threshold = 0.35

Threshold = 0.4

**Feature-Count Rule, CostERoE RoE**

Threshold = 0.45	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	77	52	129	70	38	108	80	60	140	81	61	142	81	52	133	83	39	122	80	35	115	81	31	112
2	13	128	141	20	142	162	10	120	130	9	119	128	9	128	137	7	141	148	10	145	155	9	149	158	
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	
Prod Acc %	85.6	71.1	78.4	77.8	78.9	78.4	88.9	66.7	77.8	90	66.1	78.1	90	71.1	80.6	92.2	78.3	85.3	88.9	80.6	84.8	90	82.8	86.4	
User Acc %	59.7	90.8	75.3	64.8	87.7	76.3	57.1	92.3	74.7	57	93	75	60.9	93.4	77.2	68	95.3	81.7	69.6	93.5	81.6	72.3	94.3	83.3	
Overall Acc %	75.93			78.52			74.07			74.07			77.41			82.96			83.33			85.19			
Threshold = 0.45	Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	84	23	107	84	20	104	82	16	98	83	13	96	82	13	95	81	13	94	82	12	94	83	11	94
2	6	157	163	6	160	166	8	164	172	7	167	174	8	167	175	9	167	176	8	168	176	7	169	176	
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	
Prod Acc %	93.3	87.2	90.3	93.3	88.9	91.1	91.1	91.1	91.1	92.2	92.8	92.5	91.1	92.8	92	90	92.8	91.4	91.1	93.3	92.2	92.2	93.9	93.1	
User Acc %	78.5	96.3	87.4	80.8	96.4	88.6	83.7	95.3	89.5	86.5	96	91.3	86.3	95.4	90.9	86.2	94.9	90.6	87.2	95.5	91.4	88.3	96	92.2	
Overall Acc %	89.26			90.37			91.11			92.59			92.22			91.85			92.59			93.33			
Threshold = 0.5	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	21	17	38	74	52	126	77	55	132	82	67	149	81	53	134	83	48	131	81	34	115	81	32	113
	2	69	163	232	16	128	144	13	125	138	8	113	121	9	127	136	7	132	139	9	146	155	9	148	157
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	
Prod Acc %	23.3	90.6	57	82.2	71.1	76.7	85.6	69.4	77.5	91.1	62.8	77	90	70.6	80.3	92.2	73.3	82.8	90	81.1	85.6	90	82.2	86.1	
User Acc %	55.3	70.3	62.8	58.7	88.9	73.8	58.3	90.6	74.5	55	93.4	74.2	60.4	93.4	76.9	63.4	95	79.2	70.4	94.2	82.3	71.7	94.3	83	
Overall Acc %	68.15			74.81			74.81			72.22			77.04			79.63			84.07			84.81			
Threshold = 0.5	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	83	24	107	84	19	103	81	20	101	84	15	99	84	15	99	83	11	94	81	10	91	82	10	92
	2	7	156	163	6	161	167	9	160	169	6	165	171	6	165	171	7	169	176	9	170	179	8	170	178
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	
Prod Acc %	92.2	86.7	89.5	93.3	89.4	91.4	90	88.9	89.5	93.3	91.7	92.5	93.3	91.7	92.5	92.2	93.9	93.1	90	94.4	92.2	91.1	94.4	92.8	
User Acc %	77.6	95.7	86.7	81.6	96.4	89	80.2	94.7	87.5	84.8	96.5	90.7	84.8	96.5	90.7	88.3	96	92.2	89	95	92	89.1	95.5	92.3	
Overall Acc %	88.52			90.74			89.26			92.22			92.22			93.33			92.96			93.33			



**Feature-Count Rule, CostERoE RoE**

Threshold = 0.55	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	58	32	90	70	42	112	79	56	135	82	61	143	81	54	135	83	46	129	80	35	115	81	30	111
2	32	148	180	20	138	158	11	124	135	8	119	127	9	126	135	7	134	141	10	145	155	9	150	159	
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	
Prod Acc %	64.4	82.2	73.3	77.8	76.7	77.3	87.8	68.9	78.4	91.1	66.1	78.6	90	70	80	92.2	74.4	83.3	88.9	80.6	84.8	90	83.3	86.7	
User Acc %	64.4	82.2	73.3	62.5	87.3	74.9	58.5	91.9	75.2	57.3	93.7	75.5	60	93.3	76.7	64.3	95	79.7	69.6	93.5	81.6	73	94.3	83.7	
Overall Acc %	76.3			77.04			75.19			74.44			76.67			80.37			83.33			85.56			
Threshold = 0.6	Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	82	24	106	83	20	103	82	16	98	80	11	91	80	12	92	80	11	91	79	10	89	79	7	86
2	7	155	162	7	160	167	6	162	168	4	165	169	4	164	168	4	165	169	4	165	169	4	166	170	
	89	179	268	90	180	270	88	178	266	84	176	260	84	176	260	84	176	260	83	175	258	83	173	256	
Prod Acc %	92.1	86.6	89.4	92.2	88.9	90.6	93.2	91	92.1	95.2	93.8	94.5	95.2	93.2	94.2	95.2	93.8	94.5	95.2	94.3	94.8	95.2	96	95.6	
User Acc %	77.4	95.7	86.6	80.6	95.8	88.2	83.7	96.4	90.1	87.9	97.6	92.8	87	97.6	92.3	87.9	97.6	92.8	88.8	97.6	93.2	91.9	97.6	94.8	
Overall Acc %	88.43			90.00			91.73			94.23			93.85			94.23			94.57			95.7			
Threshold = 0.6	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	72	55	127	79	61	140	78	56	134	80	54	134	80	54	134	82	36	118	81	33	114	82	27	109
	2	18	125	143	11	119	130	12	124	136	10	126	136	10	126	136	8	142	150	9	147	156	8	152	160
	90	180	270	90	180	270	90	180	270	90	180	270	90	180	270	90	178	268	90	180	270	90	179	269	
Prod Acc %	80	69.4	74.7	87.8	66.1	77	86.7	68.9	77.8	88.9	70	79.5	88.9	70	79.5	91.1	79.8	85.5	90	81.7	85.9	91.1	84.9	88	
User Acc %	56.7	87.4	72.1	56.4	91.5	74	58.2	91.2	74.7	59.7	92.6	76.2	59.7	92.6	76.2	69.5	94.7	82.1	71.1	94.2	82.7	75.2	95	85.1	
Overall Acc %	72.96			73.33			74.81			76.3			76.3			83.58			84.44			86.99			
Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2		
1	83	20	103	85	20	105	83	17	100	80	13	93	80	12	92	80	11	91	79	10	89	79	7	86	
2	6	159	165	5	160	165	5	161	166	4	163	167	4	164	168	4	165	169	4	165	169	4	166	170	
	89	179	268	90	180	270	88	178	266	84	176	260	84	176	260	84	176	260	83	175	258	83	173	256	
Prod Acc %	93.3	88.8	91.1	94.4	88.9	91.7	94.3	90.4	92.4	95.2	92.6	93.9	95.2	93.2	94.2	95.2	93.8	94.5	95.2	94.3	94.8	95.2	96	95.6	
User Acc %	80.6	96.4	88.5	81	97	89	83	97	90	86	97.6	91.8	87	97.6	92.3	87.9	97.6	92.8	88.8	97.6	93.2	91.9	97.6	94.8	
Overall Acc %	90.3			90.74			91.73			93.46			93.85			94.23			94.57			95.7			

**Feature-Count Rule, CostERoE RoE**

Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>			
	Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2		
	1	82	76	158	78	62	140	75	50	125	76	45	121	79	40	119	81	36	117	80	31	111	82	25	107
2	0	0	0	9	115	124	15	130	145	13	134	147	10	138	148	9	139	148	10	149	159	8	154	162	
	82	76	158	87	177	264	90	180	270	89	179	268	89	178	267	90	175	265	90	180	270	90	179	269	
Prod Acc %				89.7	65	77.4	83.3	72.2	77.8	85.4	74.9	80.2	88.8	77.5	83.2	90	79.4	84.7	88.9	82.8	85.9	91.1	86	88.6	
User Acc %				55.7	92.7	74.2	60	89.7	74.9	62.8	91.2	77	66.4	93.2	79.8	69.2	93.9	81.6	72.1	93.7	82.9	76.6	95.1	85.9	
Overall Acc %				73.11			75.93			78.36			81.27			83.02			84.81			87.73			
Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>			
	Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2		
	1	78	20	98	80	21	101	80	16	96	77	12	89	77	11	88	77	10	87	76	9	85	76	7	83
2	7	159	166	5	159	164	4	162	166	4	164	168	4	162	166	4	163	167	4	163	167	4	164	168	
	85	179	264	85	180	265	84	178	262	81	176	257	81	173	254	81	173	254	80	172	252	80	171	251	
Prod Acc %	91.8	88.8	90.3	94.1	88.3	91.2	95.2	91	93.1	95.1	93.2	94.2	95.1	93.6	94.4	95.1	94.2	94.7	95	94.8	94.9	95	95.9	95.5	
User Acc %	79.6	95.8	87.7	79.2	97	88.1	83.3	97.6	90.5	86.5	97.6	92.1	87.5	97.6	92.6	88.5	97.6	93.1	89.4	97.6	93.5	91.6	97.6	94.6	
Overall Acc %	89.77			90.19			92.37			93.77			94.09			94.49			94.84			95.62			
Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2			
	1	68	38	106	69	39	108	75	48	123	78	43	121	76	41	117	71	30	101	77	25	102	77	18	95
	2	0	0	0	9	110	119	13	122	135	11	134	145	13	134	147	12	144	156	9	152	161	8	159	167
	68	38	106	78	149	227	88	170	258	89	177	266	89	175	264	83	174	257	86	177	263	85	177	262	
Prod Acc %				88.5	73.8	81.2	85.2	71.8	78.5	87.6	75.7	81.7	85.4	76.6	81	85.5	82.8	84.2	89.5	85.9	87.7	90.6	89.8	90.2	
User Acc %				63.9	92.4	78.2	61	90.4	75.7	64.5	92.4	78.5	65	91.2	78.1	70.3	92.3	81.3	75.5	94.4	85	81.1	95.2	88.2	
Overall Acc %				78.85			76.36			79.7			79.55			83.66			87.07			90.08			
Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2			
	1	76	18	94	76	14	90	77	11	88	77	10	87	77	9	86	77	7	84	76	7	83	76	7	83
	2	8	157	165	5	161	166	4	163	167	4	162	166	4	160	164	4	161	165	4	161	165	4	161	165
	84	175	259	81	175	256	81	174	255	81	172	253	81	169	250	81	168	249	80	168	248	80	168	248	
Prod Acc %	90.5	89.7	90.1	93.8	92	92.9	95.1	93.7	94.4	95.1	94.2	94.7	95.1	94.7	94.9	95.1	95.8	95.5	95	95.8	95.4	95	95.8	95.4	
User Acc %	80.9	95.2	88.1	84.4	97	90.7	87.5	97.6	92.6	88.5	97.6	93.1	89.5	97.6	93.6	91.7	97.6	94.7	91.6	97.6	94.6	91.6	97.6	94.6	
Overall Acc %	89.96			92.58			94.12			94.47			94.8			95.58			95.56			95.56			

Threshold = 0.65

Threshold = 0.7

**Feature-Count Rule, CostERoE RoE**

Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>			
	Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2		
	1	23	16	39	57	34	91	63	30	93	76	41	117	79	32	111	76	28	104	78	22	100	76	15	91
2	0	0	0	0	0	0	16	129	145	11	121	132	7	134	141	6	143	149	8	154	162	9	162	171	
	23	16	39	57	34	91	79	159	238	87	162	249	86	166	252	82	171	253	86	176	262	85	177	262	
Prod Acc %							79.7	81.1	80.4	87.4	74.7	81.1	91.9	80.7	86.3	92.7	83.6	88.2	90.7	87.5	89.1	89.4	91.5	90.5	
User Acc %							67.7	89	78.4	65	91.7	78.4	71.2	95	83.1	73.1	96	84.6	78	95.1	86.6	83.5	94.7	89.1	
Overall Acc %							80.67			79.12			84.52			86.56			88.55			90.84			
Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>			
	Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2		
	1	77	18	95	77	12	89	77	11	88	77	9	86	77	9	86	77	7	84	76	7	83	76	7	83
2	6	156	162	4	162	166	4	163	167	4	163	167	4	160	164	4	161	165	4	161	165	4	161	165	
	83	174	257	81	174	255	81	174	255	81	172	253	81	169	250	81	168	249	80	168	248	80	168	248	
Prod Acc %	92.8	89.7	91.3	95.1	93.1	94.1	95.1	93.7	94.4	95.1	94.8	95	95.1	94.7	94.9	95.1	95.8	95.5	95	95.8	95.4	95	95.8	95.4	
User Acc %	81.1	96.3	88.7	86.5	97.6	92.1	87.5	97.6	92.6	89.5	97.6	93.6	89.5	97.6	93.6	91.7	97.6	94.7	91.6	97.6	94.6	91.6	97.6	94.6	
Overall Acc %	90.66			93.73			94.12			94.86			94.8			95.58			95.56			95.56			
Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2			
	1	0	0	0	5	9	14	16	12	28	74	26	100	77	29	106	72	21	93	75	20	95	76	15	91
	2	0	0	0	0	0	0	14	111	125	12	123	135	8	118	126	8	143	151	8	154	162	7	156	163
	0	0	0	5	9	14	30	123	153	86	149	235	85	147	232	80	164	244	83	174	257	83	171	254	
Prod Acc %							53.3	90.2	71.8	86	82.6	84.3	90.6	80.3	85.5	90	87.2	88.6	90.4	88.5	89.5	91.6	91.2	91.4	
User Acc %							57.1	88.8	73	74	91.1	82.6	72.6	93.7	83.2	77.4	94.7	86.1	78.9	95.1	87	83.5	95.7	89.6	
Overall Acc %							83.01			83.83			84.05			88.11			89.11			91.34			
Classes	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2			
	1	76	13	89	77	13	90	77	9	86	76	9	85	76	8	84	76	7	83	76	7	83	76	7	83
	2	4	155	159	4	156	160	4	161	165	4	159	163	4	160	164	4	161	165	4	161	165	4	161	165
	80	168	248	81	169	250	81	170	251	80	168	248	80	168	248	80	168	248	80	168	248	80	168	248	
Prod Acc %	95	92.3	93.7	95.1	92.3	93.7	95.1	94.7	94.9	95	94.6	94.8	95	95.2	95.1	95	95.8	95.4	95	95.8	95.4	95	95.8	95.4	
User Acc %	85.4	97.5	91.5	85.6	97.5	91.6	89.5	97.6	93.6	89.4	97.5	93.5	90.5	97.6	94.1	91.6	97.6	94.6	91.6	97.6	94.6	91.6	97.6	94.6	
Overall Acc %	93.15			93.2			94.82			94.76			95.16			95.56			95.56			95.56			

**Feature-Count Rule, CostERoE RoE**

<b>Threshold = 0.85</b>	<b>Markov</b>	<b>0<sup>th</sup></b>			<b>1<sup>st</sup></b>			<b>2<sup>nd</sup></b>			<b>3<sup>rd</sup></b>			<b>4<sup>th</sup></b>			<b>5<sup>th</sup></b>			<b>6<sup>th</sup></b>			<b>7<sup>th</sup></b>		
	<b>Classes</b>	<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>	
	<b>1</b>	0	0	0	0	0	0	8	1	9	58	19	77	66	20	86	70	14	84	71	13	84	72	9	81
	<b>2</b>	0	0	0	0	0	0	0	0	0	6	99	105	6	116	122	7	142	149	5	148	153	5	153	158
		0	0	0	0	0	0	8	1	9	64	118	182	72	136	208	77	156	233	76	161	237	77	162	239
	<b>Prod Acc %</b>										90.6	83.9	87.3	91.7	85.3	88.5	90.9	91	91	93.4	91.9	92.7	93.5	94.4	94
	<b>User Acc %</b>										75.3	94.3	84.8	76.7	95.1	85.9	83.3	95.3	89.3	84.5	96.7	90.6	88.9	96.8	92.9
	<b>Overall Acc %</b>										86.26			87.5			90.99			92.41			94.14		
	<b>Markov</b>	<b>8<sup>th</sup></b>			<b>9<sup>th</sup></b>			<b>10<sup>th</sup></b>			<b>11<sup>th</sup></b>			<b>12<sup>th</sup></b>			<b>13<sup>th</sup></b>			<b>14<sup>th</sup></b>			<b>15<sup>th</sup></b>		
	<b>Classes</b>	<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>	
	<b>1</b>	74	9	83	76	9	85	76	7	83	76	7	83	76	7	83	76	7	83	76	7	83	76	7	83
	<b>2</b>	5	153	158	4	157	161	4	158	162	4	158	162	4	160	164	4	161	165	4	161	165	4	161	165
		79	162	241	80	166	246	80	165	245	80	165	245	80	167	247	80	168	248	80	168	248	80	168	248
	<b>Prod Acc %</b>	93.7	94.4	94.1	95	94.6	94.8	95	95.8	95.4	95	95.8	95.4	95	95.8	95.4	95	95.8	95.4	95	95.8	95.4	95	95.8	95.4
	<b>User Acc %</b>	89.2	96.8	93	89.4	97.5	93.5	91.6	97.5	94.6	91.6	97.5	94.6	91.6	97.6	94.6	91.6	97.6	94.6	91.6	97.6	94.6	91.6	97.6	94.6
<b>Overall Acc %</b>	94.19			94.72			95.51			95.51			95.55			95.56			95.56			95.56			
<b>Threshold = 0.9</b>	<b>Classes</b>	<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>	
	<b>1</b>	0	0	0	0	0	0	0	0	0	32	3	35	46	6	52	61	10	71	66	6	72	69	6	75
	<b>2</b>	0	0	0	0	0	0	0	0	0	0	0	0	6	110	116	6	126	132	5	130	135	5	148	153
		0	0	0	0	0	0	0	0	0	32	3	35	52	116	168	67	136	203	71	136	207	74	154	228
	<b>Prod Acc %</b>													88.5	94.8	91.7	91	92.6	91.8	93	95.6	94.3	93.2	96.1	94.7
	<b>User Acc %</b>													88.5	94.8	91.7	85.9	95.5	90.7	91.7	96.3	94	92	96.7	94.4
	<b>Overall Acc %</b>													92.86			92.12			94.69			95.18		
	<b>Classes</b>	<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>		<b>1</b>	<b>2</b>	
	<b>1</b>	73	7	80	75	7	82	75	7	82	75	7	82	75	7	82	73	7	80	73	7	80	73	7	80
	<b>2</b>	4	152	156	4	157	161	4	158	162	4	158	162	4	160	164	4	161	165	4	161	165	4	161	165
		77	159	236	79	164	243	79	165	244	79	165	244	79	167	246	77	168	245	77	168	245	77	168	245
	<b>Prod Acc %</b>	94.8	95.6	95.2	94.9	95.7	95.3	94.9	95.8	95.4	94.9	95.8	95.4	94.9	95.8	95.4	94.8	95.8	95.3	94.8	95.8	95.3	94.8	95.8	95.3
	<b>User Acc %</b>	91.3	97.4	94.4	91.5	97.5	94.5	91.5	97.5	94.5	91.5	97.5	94.5	91.5	97.6	94.6	91.3	97.6	94.5	91.3	97.6	94.5	91.3	97.6	94.5
	<b>Overall Acc %</b>	95.34			95.47			95.49			95.49			95.53			95.51			95.51			95.51		

**Feature-Count Rule, CostERoE RoE**

Threshold = 0.95	Markov	0 <sup>th</sup>			1 <sup>st</sup>			2 <sup>nd</sup>			3 <sup>rd</sup>			4 <sup>th</sup>			5 <sup>th</sup>			6 <sup>th</sup>			7 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	0	0	0	0	0	0	0	0	0	5	0	5	32	2	34	58	6	64	66	6	72	69	6	75
2	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	4	70	74	5	126	131	5	148	153	
	0	0	0	0	0	0	0	0	0	5	0	5	32	5	37	62	76	138	71	132	203	74	154	228	
Prod Acc %													100	60	80	93.5	92.1	92.8	93	95.5	94.3	93.2	96.1	94.7	
User Acc %													94.1	100	97.1	90.6	94.6	92.6	91.7	96.2	94	92	96.7	94.4	
Overall Acc %													94.59			92.75			94.58			95.18			
Threshold = 1	Markov	8 <sup>th</sup>			9 <sup>th</sup>			10 <sup>th</sup>			11 <sup>th</sup>			12 <sup>th</sup>			13 <sup>th</sup>			14 <sup>th</sup>			15 <sup>th</sup>		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	73	7	80	75	7	82	75	7	82	75	7	82	75	7	82	73	7	80	73	7	80	73	7	80
	2	4	152	156	4	157	161	4	158	162	4	158	162	4	160	164	4	161	165	4	161	165	4	161	165
		77	159	236	79	164	243	79	165	244	79	165	244	79	167	246	77	168	245	77	168	245	77	168	245
	Prod Acc %	94.8	95.6	95.2	94.9	95.7	95.3	94.9	95.8	95.4	94.9	95.8	95.4	94.9	95.8	95.4	94.8	95.8	95.3	94.8	95.8	95.3	94.8	95.8	95.3
	User Acc %	91.3	97.4	94.4	91.5	97.5	94.5	91.5	97.5	94.5	91.5	97.5	94.5	91.5	97.6	94.6	91.3	97.6	94.5	91.3	97.6	94.5	91.3	97.6	94.5
	Overall Acc %	95.34			95.47			95.49			95.49			95.53			95.51			95.51			95.51		
	Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2	
	1	0	0	0	0	0	0	0	0	0	5	0	5	14	2	16	44	7	51	62	7	69	65	7	72
2	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	0	26	26	2	99	101	3	145	148	
	0	0	0	0	0	0	0	0	0	5	0	5	14	5	19	44	33	77	64	106	170	68	152	220	
Prod Acc %													100	60	80	100	78.8	89.4	96.9	93.4	95.2	95.6	95.4	95.5	
User Acc %													87.5	100	93.8	86.3	100	93.2	89.9	98	94	90.3	98	94.2	
Overall Acc %													89.47			90.91			94.71			95.45			
Classes	1	2		1	2		1	2		1	2		1	2		1	2		1	2		1	2		
1	70	7	77	72	7	79	72	7	79	72	7	79	72	7	79	72	7	79	72	7	79	72	7	79	
2	3	152	155	4	157	161	4	158	162	4	158	162	4	160	164	4	161	165	4	161	165	4	161	165	
	73	159	232	76	164	240	76	165	241	76	165	241	76	167	243	76	168	244	76	168	244	76	168	244	
Prod Acc %	95.9	95.6	95.8	94.7	95.7	95.2	94.7	95.8	95.3	94.7	95.8	95.3	94.7	95.8	95.3	94.7	95.8	95.3	94.7	95.8	95.3	94.7	95.8	95.3	
User Acc %	90.9	98.1	94.5	91.1	97.5	94.3	91.1	97.5	94.3	91.1	97.5	94.3	91.1	97.6	94.4	91.1	97.6	94.4	91.1	97.6	94.4	91.1	97.6	94.4	
Overall Acc %	95.69			95.42			95.44			95.44			95.47			95.49			95.49			95.49			

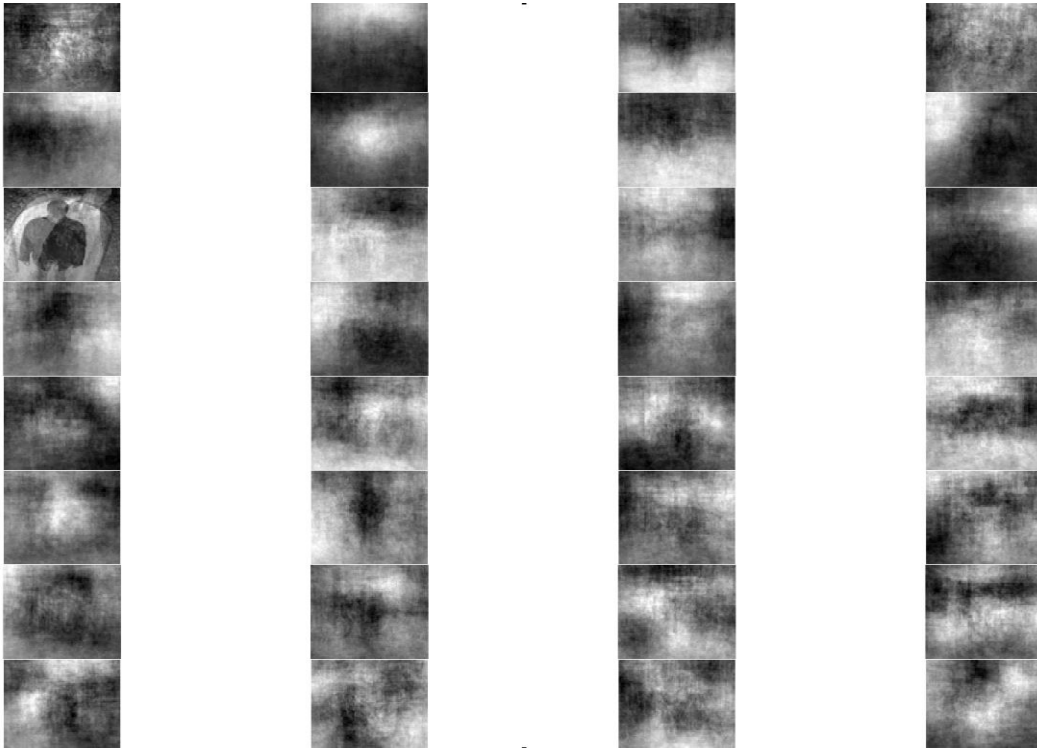


Figure A.1. RVQ codebook with  $M = 4$  and  $P = 8$  for Graz dataset.

Table A.3 . Error matrix and classification performance measures for SVM on Graz dataset

<b>Classes</b>	<b>Bicycle</b>	<b>People</b>	<b>Background</b>	
<b>Bicycle</b>	245	10	0	255
<b>People</b>	7	153	2	162
<b>Background</b>	8	2	52	62
	260	165	54	479
<b>Prod Acc %</b>	<b>94.2</b>	<b>92.7</b>	<b>96.3</b>	
<b>User Acc %</b>	<b>96.1</b>	<b>94.4</b>	<b>83.9</b>	
<b>Overall Acc %</b>	<b>93.95</b>			

## REFERENCES

- [1] C. Barnes and R. Frost, "Vector Quantizers with Direct Sum Codebooks," *IEEE Transactions on Information Theory*, vol. 39, no. 2, March 1993.
- [2] T. Morris, *Computer Vision and Image Processing*, Palgrave Macmillan, 2004.
- [3] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999.
- [4] S. Lazebnit, C. Schmid and J. Ponce, "Semi-Local Affine Parts for Object Recognition," in *Proceedings of the British Machine Vision Conference*, 2004.
- [5] H. Bay, T. Tuytelaars and L.V. Gool, "SURF: Speeded Up Robust Features," in *Proceedings of the ninth European Conference on Computer Vision*, 2006.
- [6] K. Mikolajczyk and C. Schmid, "a performance evaluation of local descriptors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 10, pp. 1615-1630, October 2005.
- [7] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research 3 (2003) 1157-1182*, vol. 3, pp. 1157-1182, 2003.
- [8] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
- [9] C. Cortes and V. N.Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, 1995.
- [10] C. Zhang and E. Baltsavias, "Knowledge-Based Image Analysis for 3D Edge Extraction and Road Reconstruction," *International Archives of Photogrammetry and Remote Sensing*, vol. XXXIII, 2000.
- [11] P. Zimmermann, "A New Framework for Automatic Building Detection Analysing Multiple Cue Data," *International Archive of Photogrammetry and Remote Sensing*, vol. XXXIII, no. B3, 2000.
- [12] B. Sirmacek and C. Unsalan, "Building Detection from Aerial Images Using Invariant Color Features and Shadow Information," in *23rd International Symposium on Computer and Information Sciences*, 2008.
- [13] M. Rizon, H. Yazid, P. Saad and A. Y. M. Shakkaf, "Object Detection Using Geometric Invariant Moment," *American Journal of Applied Sciences*, vol. 2, no. 6, pp. 1876-1878, 2006.
- [14] S. Z. Li, "A Markov Random Field Model for Object Matching under Contextual Constraints", Proceeding of IEEE CVPR'94, pp. 866-869, 1995," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1995.
- [15] R. Brunelli and T. Poggio, "Face Recognition: Features versus Templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042-1052, October 1993.
- [16] Theodoridis.S. and K. Koutroumbas, *Pattern Recognition*, 3 ed., San Diego, CA: Academic Press, c2006.
- [17] B. Draper, K. Baek, M. Bartlett and J. R. Beveridge, "Recognizing Faces with PCA and ICA," *Computer vision and image understanding, Special issue of face*

- recognition*, vol. 91, no. (1/2), pp. 115-137, 2003.
- [18] M. Bartlett, J. R. Movellan and T. J. Sejnowski, "Face Recognition by Independent Component Analysis," *IEEE Transactions on Neural Networks*, vol. 13, no. 6, 2002.
- [19] C. Barnes and S. Rizvi, "Advances in Residual Vector Quantization: A Review," *IEEE Transactions on Image Processing*, vol. 5, no. 2, February 1996.
- [20] Y. Linde, A. Buzo and R. Gray, "An Algorithm for Vector Quantization Design," *IEEE Transactions on Communications*, Vols. Com-28, no. 1, p. 1980.
- [21] C. F. Barnes and R. L. Frost, "Residual Vector Quantizers with Jointly Optimized Code Books," in *Advances in Electronics and Electron Physics*, 1992.
- [22] C. F. Barnes, "Image-Driven Data Mining for Image Content Segmentation, Classification and Attribution , Vol. 45, No.9. 2007," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 9, September 2007.
- [23] C. Barnes, "Hurricane Disaster Assessments with Image-Driven Data Mining in High-Resolution Satellite Imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 6, June 2007.
- [24] C. F. Barnes and J. Burki, "Late-Season Rural Land-Cover Estimation With Polarimetric-SAR Intensity Pixel Blocks and  $\sigma$ -Tree-Structured Near-Neighbor Classifiers," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 9, 2009.
- [25] K. Sayood, *Introduction to Data Compression*, 3 ed., Morgan Kaufmann, 2005.
- [26] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*, Springer, 1991.
- [27] A. G. A. Buzo, R. M. Gray and J. Markell, ", "Speech coding based upon vector quantization," vol. 28, pp. 562 – 574, Oct 1980.," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, pp. 562-574, October 1980.
- [28] G. Motta, F. Rizzo and J. Storer, "Partitioned Vector Quantization: Application to Lossless Compression of Hyperspectral Images," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.
- [29] B. H. Juang and A. H. Gray, "Multiple Stage Vector Quantization for Speech Coding," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, 1982.
- [30] T. Kohonen, *Self-Organizing Maps*, Berlin: Springer, 1997.
- [31] B. Ramamurthi and A. Gresho, "Classified Vector Quantization of Images," *IEEE Transactions on Communications*, Vols. com-34, no. 11, November 1986.
- [32] S. Supot and S. Manas, "Codebook Design Algorithm for Classified Vector Quantization Based on Fuzzy Clustering," in *IEEE International Conference on Industrial Technology*, 2002.
- [33] H.-H. Chen, H.-T. Sheu and J.-J. Ding, "Quadtree Classified Vector Quantization Based Image Retrieval Scheme," in *Eighteenth IEEE International Conference on Image Processing*, 2011.
- [34] N. R. Pal and J. C. Bezdek, "On Cluster Validity for the Fuzzy C-Means Model," *IEEE Transactions on Fuzzy System*, vol. 3, pp. 370-372, 1995.
- [35] X. Yang, D. Xu and Y.-J. Qi, "Bag-of-words Image Representation Based on



- Classified Vector Quantization," in *Proceedings of the Ninth International Conference on Machine Learning and Cybernetics*, Qingdao, 2010.
- [36] L. Fei-Fei, R. Fergus and A. Torralba, "Recognizing and Learning Object Categories: A Short Course," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [37] B. Zhang and Y. Zhou, "Reliable Vehicle Type Classification by Classified Vector Quantization," in *IEEE Fifth International Congress on Image and Signal Processing*, 2012.
- [38] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2 ed., Prentice Hall, 1998.
- [39] N. Pal, J. Bezdek and E.-K. Tsao, "Generalized Clustering Networks and Kohonen's Self-Organizing Scheme," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 549-557, 1993.
- [40] S. Hotta, "Learning Vector Quantization with Local Subspace Classifier," in *The 19th IEEE International Conference on Pattern Recognition*, 2008.
- [41] P. Schneider, M. Biehl and B. Hammer, "Distance Learning in Discriminative Vector Quantization," *Neural Computation*, vol. 21, no. 10, pp. 2942-2969, October 2009.
- [42] Fahad and Sikander, "Classification of Textual documents Using Learning Vector Quantization," *Information Technology Journal*, vol. 6, no. 1, 2007.
- [43] M. Pilevar, H. Feili and M. Soltani, "Classification of Persian textual documents using learning vector quantization," in *Natural Language Processing and Knowledge Engineering*, 2009.
- [44] P. C. C. R. M. G. a. J. M. K. L. Oehler, "Classification Using Vector Quantization," in *Conference Record of the Twenty Fifth Asilomar Conference on Signals, Systems and Computers*, 1991.
- [45] K. L. Oehler and R. M. Gray, "Combining Image Compression and Classification Using Vector Quantization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 461-473, 1995.
- [46] L. Rokach and O. Maimon, *Data Mining with Decision Trees: Theory and Applications*, World Scientific Pub Co Inc, 2008.
- [47] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The PASCAL Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303-338, 2010.
- [48] A. Opelt, A. Pinz, M. Fussenegger and P. Auer, "Generic object recognition with boosting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 3, pp. 416-431, 2006.
- [49] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," in *Proceedings of the IEEE*, 1998.
- [50] Y. LeCun, C. Cortes and C. J. Burges, "The MNIST Database of Handwritten Database," [Online].
- [51] R. G. .. Congalton, *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*, 2 ed., Boca Raton: CRC Press, 2008.

- [52] J. H. Friedman, J. Bentely and R. A. and Finkel, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," *ACM Transactions on Mathematical Software* 3, 209, 1977.
- [53] F. Kossentini, M. Smith and C. Barnes, "Image Coding Using Entropy-Constrained Residual Vector Quantization," *IEEE Transactions on Image Processing*, vol. 4, no. 10, pp. 1349-1357, 1995.