A Fast and Area-efficient Architecture for Classifying Images Based on Binarization and a Binary Dual-feature Set

by

Narges Attarmoghaddam
B.Sc., Ferdowsi University of Mashhad, 2009
M.Sc., Ferdowsi University of Mashhad, 2014

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

A Fast and Area-efficient Architecture for Classifying Images Based on Binarization and a Binary Dual-feature Set

by

Narges Attarmoghaddam
B.Sc., Ferdowsi University of Mashhad, 2009
M.Sc., Ferdowsi University of Mashhad, 2014

Supervisory Committee

---

Dr. Kin Fun Li, Supervisor
(Department of Electrical and Computer Engineering)

---

Dr. Watheq El-Kharashi, Departmental Member
(Department of Electrical and Computer Engineering)

---

Dr. Sudhakar Ganti, Outside Member
(Department of Computer Science)

## ABSTRACT

Image classification is an active research area in computer vision with many applications. Image classification is a computationally complex task, but for embedded applications, only limited resources are allowed. Moreover, additional constraints such as reliable classification accuracy, high-throughput performance, power-efficiency, and real-time speed must be fulfilled. Due to potential for parallelism, low power consumption, scalable resource utilization, and reconfigurability, Field-Programmable Gate Array (FPGA) devices are well-suited to overcome these challenges in image classification system implementation.

In this dissertation, we implemented two feature-based image classification systems using binary feature sets. A binary feature of an image pixel is represented using one bit, while a non-binary feature is represented using more bits. Therefore, implementing an image classification system based on a binary feature set needs fewer hardware resources for storage and computation.

The first proposed system is founded on a single binary Histograms of Oriented Gradients (HOG) feature set. Using a binary feature set generally leads to an area-efficient and faster architecture; however, accuracy is lost because binary features contain less information than non-binary features. We, therefore, proposed the second system that uses a dual feature set that combines HOG and Local Binary Pattern (LBP) features to improve classification accuracy performance. The Support Vector Machine (SVM) classification algorithm is utilized as the classifier in both proposed systems.

To obtain binary features, two steps of binarization are applied to the HOG descriptor. First, HOG features are extracted from binary images to simplify the feature extraction process. Second, the block histogram normalization of the both HOG and LBP descriptors is replaced using binarization to reduce hardware resource utilization in the descriptors and the SVM classifier.

Binary-based computation in the proposed systems results in resource utilization reduction, thus allowing an area-efficient architecture even with two feature descriptors implemented. Compared to similar existing works, our system speeds up the classification process while utilizing fewer hardware resources, with comparable accuracy.

# Contents

# List of Tables

# List of Figures

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Kin Fun Li for welcoming me to his lab and amongst his research group. Throughout the years of my Ph.D., he gave me all the support, and encouragement I needed. The completion of this study could not have been possible without his mentoring. I would also like to thank Dr. Watheq El-Kharashi and Dr. Sudhakar Ganti for sitting on my supervisory committee and taking the time to review my dissertation.

I would like to thank all of my friends in the computer architecture lab for providing me with social support and their help in shaping my dissertation path.

Finally, and most importantly, I would like to thank my family for their unconditional love, support, and encouragement to continue my studies. I was lucky to have my lovely parents with me in Canada during the last few months of my Ph.D. program.

Narges Attarmoghaddam

# DEDICATION

I dedicate this work to my supportive family. A special feeling of gratitude to my loving parents, Ali Asghar and Simin, who have always loved me unconditionally. This work is also dedicated to my friends, who have been a constant source of support and encouragement during the challenges of graduate school and life.

# Chapter 1

# Introduction

Over the past few decades, computers have become an essential part of life. They do compute-intensive tasks faster and more accurately than humans. Researchers attempt to improve the computer's abilities to do more intelligent tasks, such as visual and speech analysis, reasoning, and decision-making. However, computers are still far behind humans in performing many high-level tasks such as human recognition. Giving computers the ability to "see" is one goal of researchers in the field of computer vision and machine learning. Computers should be capable of visual analysis and interpretation of images and videos. One of fundamental tasks in this field is detecting various object classes. Object classes can be man-made, such as airplanes, cars, motorbikes, watches, or natural ones, such as horses, faces, pedestrians, and sheep.

Object detection in images and videos has many applications; for example in self-driving vehicles, video surveillance, robotics, and intelligent traffic systems. Object detection aims to detect whether the image includes a particular object of interest. Furthermore, real-time performance is vital in some applications, such as security and pedestrian detection in driver assistance systems. "Real-time" processing refers to the time interval between two steps of processing and means producing output almost simultaneously with the input. In general, the real-time processing rate depends on the application. In hardware implementations, real-time processing means the algorithm runs at the rate of the source; for example, in image processing, the camera supplies the images. The current standard for video capture is typically 30 frames per second (fps). Since real-time processing requires processing all of the frames at the same rate they are captured, the minimum real-time detection rate is 30 fps [1].

Solving a detection problem involves specifying the location of one or more similar or different objects in an image. The image is divided into many windows, and each window is searched for whether it contains the object of interest. This process is called a binary classification that requires a one-class classifier to classify each window into two classes: object-like and non-object. Then, further processing is needed to combine binary classification results for windows to obtain the final detection result. Therefore, a detection problem consists of many binary classification problems. Besides this, the classification can be performed as a multi-class classification. In a multi-class classification problem, the input image is classified into one of the one or more predefined object classes. Therefore, image classification is one of the fundamental problems in computer vision.

Processing speed and classification accuracy are two primary metrics in the performance evaluation of classification systems. The processing speed defines how fast the system can process several input images.

Accuracy can be interpreted in various ways in different domains and applications. In the study of classification systems, accuracy is used as a statistical measure to evaluate how well a classifier can predict correctly. The accuracy metric is defined as the proportion of true predictions among the total number of tested samples.

The feature-based technique is a popular approach to design detection and classification systems [2], [3]. It involves extracting the necessary information from an image using a feature descriptor. Then, it uses a classification algorithm to process the extracted feature vector to determine whether the image contains the object of interest. Commonly-used feature extraction and classification algorithms are often computationally expensive [4], [5], [6]. Furthermore, these algorithms have to process a large amount of data, which makes it hard to achieve the real-time processing rate using a purely software implementation [7], [8], [9], [10]. Therefore, researchers have been motivated to employ hardware implementations to accelerate the classification process. To reach real-time performance, some researchers tried to speed up the process by parallelism using multi-core Central Processing Units (CPUs) [11], [12], Graphic Processing Units (GPUs) [13], [14], or Field-Programmable Gate Arrays (FPGAs) [1], [15], [16].

Using a multi-core CPU, GPU, or FPGA makes it possible to execute more than one task in parallel, so that the overall processing time is reduced. However, a multi-core CPU and GPU require enormous hardware resources that lead to high power consumption [17]. Therefore, they are not suitable for embedded systems. In contrast, many works have shown that the FPGA platform is well-suited to implement real-time classification and object detection systems due to its reasonable power consumption and reconfigurable capability [18], [19], [20], [21].

This research studied the possibility of image classification and object detection based on a binary feature system. We attempted to accelerate the process using FPGA implementations. This chapter introduces the object detection problem and the classification problem's importance in computer vision. The remainder of the chapter presents the statement of the research problem in Section 1.1. In Section 1.2, the contributions of this research are summarized. An outline of the structure of the dissertation is provided in Section 1.3.

## 1.1 Statement of the Research Problem

Classification is typically a compute-intensive and time-consuming process. Further, many of the important applications need real-time classification. Software-based implementation frequently fails to achieve real-time capability for the computationally complex task of the classification systems. Accelerating computationally intensive algorithms of the classification systems using hardware-based implementation is a challenging task that needs to be addressed. High classification accuracy is essential for some applications, like autonomous vehicles, to operate safely. In this research we addressed the issues of overcoming the computational complexity and high resource-demand of the classification systems. FPGA-based systems have achieved reliable performance that satisfies real-time requirements due to the inner parallelism characteristics, reasonable power consumption, and scalable resource utilization.

## 1.2 Summary of Research Contributions

This research project designed and implemented an area-efficient and fast architecture based on the binary dual-feature set for multi-class classification applications. The goals and major contributions of this dissertation can be summarized as follows:

- Implement an area-efficient and fast architecture for the feature-based multi-class classification system. The implemented multi-class classification system is based on a binary HOG-LBP (Histograms of Oriented Gradients-Local Binary Pattern) feature set. The main contribution of this research is utilizing a binary dual-feature set. To the best of our knowledge, no previous work employs a binary dual-feature set. Using *binary* features decreases the hardware resource utilization, and a *dual* feature set improves accuracy.

- Accelerate the Support Vector Machine (SVM) classification process by replacing multiplications with additions because of the binary features. This replacement also results in area efficiency in the SVM classifier module.

- Accelerate the SVM classification process for the HOG feature descriptor by performing SVM weights accumulation on block features rather than the final feature vector. This way, SVM weights accumulation can be performed in parallel with the HOG feature descriptor and thus speed up the classification process.

- Implement an extendable multi-class SVM classifier hardware architecture. The implemented One-Vs-One multi-class SVM classifier can be easily extended for a different number of object classes with a minor effect on processing time.

- Implement an area-efficient and fast architecture for a binary HOG feature descriptor. The implemented binary HOG feature descriptor employs two steps of binarization. It extracts features from binary images and generates a binary HOG feature set by applying feature binarization.

- Implement an area-efficient and fast architecture for the binary Uniform LBP feature descriptor. The implemented binary Uniform LBP feature descriptor extracts features from grayscale images and generates a binary LBP feature set by applying feature binarization. To the best of our knowledge, it is the first utilization of a binary LBP feature set in a classification system.

- Develop a new method to implement feature concatenation without creating a dual-feature set. This method accelerates the classification process by parallelizing a significant portion of the classification process for two feature sets.

## 1.3 Outline of the Dissertation

This chapter introduced object detection and classification problems and the strong need for hardware implementation. It then presented the research problem statement and summarized this dissertation's contributions. The remaining chapters are organized as follows:

- **Chapter 2** presents the theoretical background of commonly-used algorithms in a feature-based classification system. It also discusses FPGAs and their suitability for implementing an image classification system. Based on the presented background, it then describes the motivation behind this research approach. After that, it provides a literature review of the hardware implementation of the algorithms we selected for our implemented classification system. The chapter ends with a summary of the literature reviewed and a discussion on the challenges in the hardware implementation of a feature-based system.

- **Chapter 3** introduces our proposed binary single-feature and dual-feature classification systems in two sections. In each section, a system-level model of the proposed system is first introduced. Then, details of the utilized algorithmic-level techniques that optimize the hardware-related characteristics, such as area and speed, are described. This chapter does not provide hardware-level details, which are deferred to Chapter 5.

- **Chapter 4** describes the methodology of experiments to evaluate the accuracy performance of the proposed binary classification systems for human detection and multi-class classification problems in two sections. An overview of accuracy results is also provided in each section to give a sense of the overall performance of our approach to the reader before presenting details of the hardware implementation.

- **Chapter 5** covers details of the hardware implementation for the binary single-feature and dual-feature classification systems in two sections. The first section presents the hardware implementation of the binary single-feature system. The second section describes the modules added to the single-feature system to build the dual-feature system. Each section explains the internal architecture of different modules and how they work.

- **Chapter 6** presents details of the experimental results in two sections. In the first section, hardware-related characteristics, such as processing time and FPGA resource utilization of different modules of our classification systems, are reported and compared with existing solutions. A comparison of the object detection and multi-class classification systems with previous FPGA implementations is also presented. The second section provides the accuracy performance results of our binary single-feature and dual-feature classification systems for human detection and multi-class classification. The accuracy comparison in the hardware and software implementations with previous works are also presented.

- **Chapter 7** concludes the research of this dissertation. It also provides some suggestions for the future directions of this research.

# Chapter 2

# Background and Literature Review

This chapter presents the background for this research and surveys existing research in the field of accelerating classification systems using hardware implementations on FPGA.

This chapter first introduces a general feature-based classification system in Section 2.1. A feature-based classification system consists of two main modules: the feature descriptor and the classifier. Section 2.2 reviews the background of the most commonly-used feature descriptors in the field of computer vision. Section 2.3 presents the theory of the most popular classifier algorithms utilized in the classification systems. As the main focus of this work is hardware implementation, Section 2.2 and 2.3 discuss the significant constraints of the algorithms in hardware implementation to help the reader better understand the contributions of this research. Section 2.4 discusses the compatibility of FPGA technology with image classification applications. In Section 2.5, we introduced commonly-used feature types and provided a literature review on feature combination benefits in the accuracy performance of a classification system. After reviewing the background and challenges of the hardware implementation of a classification system, we introduced the motivation behind our approach in this research in Section 2.6. Based on the discussion in Section 2.6, Section 2.7 surveys existing hardware implementations of the algorithms that have been utilized to design the two classification systems in this work. Finally, Section 2.8 summaries the chapter.

## 2.1 Feature-based vs. CNN-based Image Classification System

Computer vision, in simple terms, aims to develop techniques to enable computers to "see" and interpret the content of images and videos. There are various problems in the field of computer vision, such as object detection and image classification. "Object detection" refers to identifying a specific object in an image. Solving an object detection problem involves specifying the location of one or more similar or different objects in an image. A bounding box is drawn around the identified objects to indicate their location in the image. "Classification" is the process of categorizing images based on their included objects. Classification categorizes images based on the predefined object classes; each image is assigned to one object class. Classification can be done as one-class classification and multi-class classification. Detection can be interpreted as a one-class classification problem (known as binary classification) that classifies images into two classes: object-like and non-object. In the multi-class classification problem, the input image is classified into one of the predefined object classes. Therefore, image classification is a fundamental problem among

existing problems in computer vision. There are two popular approaches for image classification in computer vision: the Convolutional Neural Networks (CNNs) and the feature-based technique.

The first approach uses the neural network architecture, such as Convolutional Neural Networks (CNNs). CNN is a supervised learning technique that consists of two phases. In the first phase, a model is created using annotated data samples. This phase uses the back-propagation algorithm to iteratively update the parameters to improve the model's prediction accuracy. The second phase uses the trained model to classify new samples, known as inference. A CNN model is trained once; thus, its complexity is not a concern in implementation. The existing literature mostly focuses on the inference phase acceleration because the inference is implemented for classifying each new data sample in the delivery platform.

Research on CNNs shows that they achieved high accuracy performance for image classification [22], [23], [24]; however, at the price of a high computational cost. The computational workload of the CNN inference includes the need for many layers with the Multiply Accumulate (MAC) operation. Therefore, the main challenge of developing CNN is to improve classification accuracy with a tolerable computational workload. References [25] and [26] surveyed existing approaches for accelerating the CNN inference in hardware implementation. Due to the parallelism capabilities, the CNN inference computation characteristics are naturally well suited to hardware implementation. However, the large number of parameters, power consumption, and resource utilization remain as challenging issues in their hardware implementation, as discussed in [25], [26].

Another popular image classification approach is the feature-based technique [27]. The feature-based technique is constructed based on the extracted features from an image. Features that capture interesting information of an image are encoded into a feature vector. Figure 2.1 shows a general block diagram of a feature-based classification system. Regardless of the application, a feature-based classification system consists of two main modules: the feature descriptor and the classifier. Like all image processing systems, one or more preprocessing steps are required to prepare the raw input image for the feature extraction process. A raw input image is an image that has been captured by a camera and can be an RGB (red, green, and blue) or grayscale image fed to the system as stream data or from a memory cell. Depending on the system requirements and application, preprocessing includes tasks such as preparation, transformation, cleaning, and reducing the input samples.

A feature descriptor is an algorithm that performs the feature extraction process by taking an image as the input and generating a feature vector/feature descriptor. Feature extraction in an image classification system involves extracting desired information from one pixel or a block of pixels of an image.

The third block of a classification system, shown in Figure 2.1, is a classifier that assigns input images and outputs to the class label. Depending on the purpose of the system, the classifier is a one-class or multi-class classifier.


Figure 2.1: General block diagram of a feature-based classification system

Table 2.1 compares the resource utilization of CNN-based and feature-based classification systems with FPGA implementation. We averaged the FPGA implementation characteristics of 20 randomly-selected

references reviewed in this research and reference [25], for feature-based and CNN-based systems, respectively. It can be seen that the feature-based classification approach is far more area-efficient than the CNN-based.

Table 2.1: CNN vs. feature-based FPGA implementation characteristics

| | **CNN-based** | **Feature-based** |
|---|---|---|
| **LUT (K)** | 340 | 35 |
| **DSP** | 2,682 | 72 |
| **Memory (kB)** | 32,000 | 976 |

To Compare accuracy performance of the feature-based and CNN-based classification systems is not straightforward because the feature-based approach is more popular for object detection systems [1], [27], [28], while the CNN-based approach is more common for image classification systems [22], [23], [24]. Due to the trade-offs between area utilization and accuracy performance, the suitable approach should be determined depending on the target application. In this research, we targeted an area-efficient architecture; therefore, we selected the feature-based approach.

## 2.2 Feature Descriptors

The feature descriptor acts as a sort of numerical fingerprint which extracts beneficial information from the image and encodes the information as features in a feature vector. The feature descriptor plays a considerable role in the overall classification accuracy of the feature-based image classification system because classification is performed based on the feature vector. Histograms of Oriented Gradients (HOG), Haar-like features, and Local Binary Pattern (LBP) are among the most commonly used feature descriptors in classification systems [29], [2], [30], [31], [32]. Therefore, this subsection reviews these three feature descriptors and their constraints in hardware implementation.

### 2.2.1 Histograms of Oriented Gradients (HOG)

The concept of Histograms of Oriented Gradients (HOG) feature descriptor was first introduced in 1995 [33]. However, HOG became widespread when N. Dalal and B. Triggs showed it outperforms other feature descriptors for human detection in 2005 [4]. As shown in Figure 2.2, HOG consists of four steps to extract features. The input image is divided into blocks, and each block is divided into cells (in our case four cells). The first step is the calculation of the gradient along the x and y axes for each cell. This step is followed by the computation of the magnitude and angle of the gradients. The next step of the process is orientation binning, which generates cell histograms by accumulating gradients of the magnitude of bins. Illumination and foreground-background contrast can affect the gradient magnitude, and so gradients must be normalized to provide better illumination invariance. To this end, the HOG feature extraction process ends with the block histogram normalization process. The final HOG descriptor is a vector made from the concatenation of the normalized cell histograms from all block regions.

The main weakness of most sliding window algorithms, like HOG, is that they are relatively slow. In a sliding window algorithm, data is processed window-by-window. Moreover, a significant constraint of the

HOG descriptor is the gradient computation that needs complex operations such as square, square root, and arctangent, which are not area-efficient in the hardware implementation.

**Normalization:** As described in [4], there are four block normalization methods for the HOG descriptor: L1-norm, L1-sqrt (L1-norm followed by square root), L2-norm, and L2-Hys. Regardless of the accuracy of the different block normalization methods, they all include square, square root, and division operations. These complex operations are not area-efficient in hardware implementation. Therefore, the normalization step is another constraint of the HOG descriptor in hardware implementation.

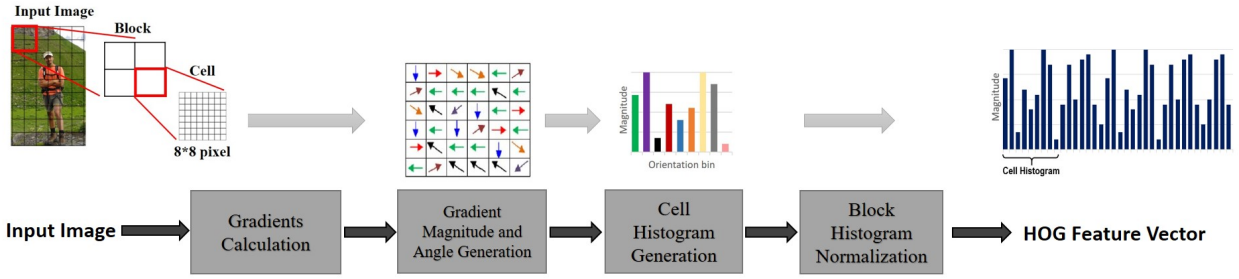



Figure 2.2: HOG feature descriptor scheme

### 2.2.2 Local Binary Pattern (LBP)

The Local Binary Pattern (LBP) feature descriptor was first introduced by Ojala et al. in 1996 [34] for the purpose of texture classification. The basic LBP feature for a given pixel is made by using a given pixel as a threshold surrounded by a set of equally spaced neighbors ($P$). Figure 2.3 shows how the LBP feature is calculated for a $3 \times 3$ block, which is the minimum possible block size because it corresponds to zero space between the center pixel and its neighbors. Let $j_c$ be the center pixel and $j_n$ ($n = 0, 1, \ldots, 7$) be the surrounding pixels. For every $j_n$, if $j_n$ is bigger than $j_c$, the binary result of the pixel is set to 1; otherwise, it is set to 0. All the results are combined as an 8-bit integer. The decimal value, which is between 0 and $2^P$, is the LBP feature.

The main drawbacks of the LBP are sensitivity to rotation and an exponential increase in the feature size with increasing number of neighbors.

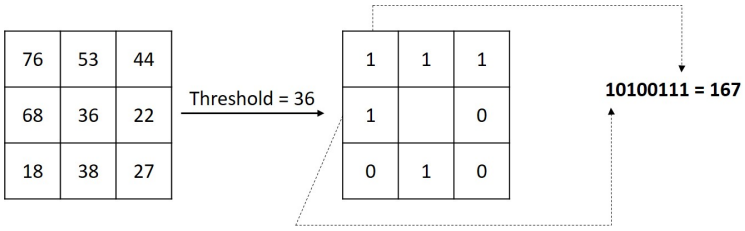



Figure 2.3: Basic LBP descriptor

Later in 2002, Ojala et al. [35] extended their work and introduced the Uniform LBP concept that is a rotation-invariant texture descriptor. A binary LBP feature is considered uniform if the number of transitions between 0's and 1's in the binary pattern is at most two 0-1 or 1-0 transitions. For example, 00100000 (2 transitions) is a uniform pattern, but 01010010 (6 transitions) is not. Figure 2.4 illustrates the idea of a

Uniform LBP. For an 8-bit LBP feature, there are 58 uniform patterns. To form the LBP histogram, a separate bin is considered for every uniform pattern, and there is one single bin for all non-uniform patterns. Therefore, the length of the histogram reduces from $2^P$, in the basic LBP descriptor, to 59 in the Uniform LBP descriptor.

The basic and Uniform LBP descriptors have low computational complexity and are not resource demanding for hardware implementation. Their weakness is that they are slow because they are sliding-window algorithms.

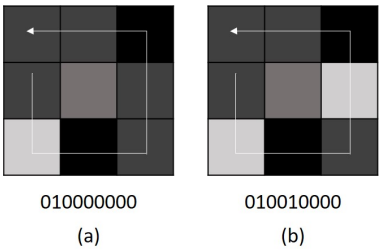



Figure 2.4: Uniform LBP descriptor. (a) Uniform pattern with two transitions (b) Non-uniform pattern with four transitions

### 2.2.3 Haar-like Features

Haar-like features are fixed-size black and white rectangles, as shown in Figure 2.5. These features act as filters that indicate the existence or absence of specific visual characteristics in an image. The basic idea is to calculate the sum of the pixel values in the white rectangles minus the sum of pixel values in the black rectangles. After accumulating the sum for all rectangles in the feature, a predefined threshold is used to determine if the framework is likely to contain the object or not. If the sum is greater than the threshold, the frame is accepted for further processing; otherwise, it is rejected. Each feature's size and the number of rectangles vary based on the object of interest. Therefore, Haar-like features should be designed specifically for every application and object of interest.

There are three kinds of rectangle Haar-like features: the 2-rectangle feature, the 3-rectangle feature, and the 4-rectangle feature, as depicted in Figure 2.5. In classification systems, Haar-like features are used with a classifier algorithm that determines the feature size, threshold, and all other required characteristics during the training phase. Haar-like features are often used as weak classifiers. Weak classifiers are classifiers that have only slightly better performance than a random classifier. If the Haar-like feature is used as a weak classifier, the result has a high false positive rate. Many cascade stages are needed to decrease the rate of false positives; however, this will result in a remarkable increase in processing time and a decrease in the classification rate. Haar-like features have low computational complexity and are not challenging to implement in hardware.

## 2.3 Classifiers

Machine Learning (ML) algorithms are the most popular and common classifiers in feature-based classification systems. There are three types of ML algorithms, including supervised learning, unsupervised

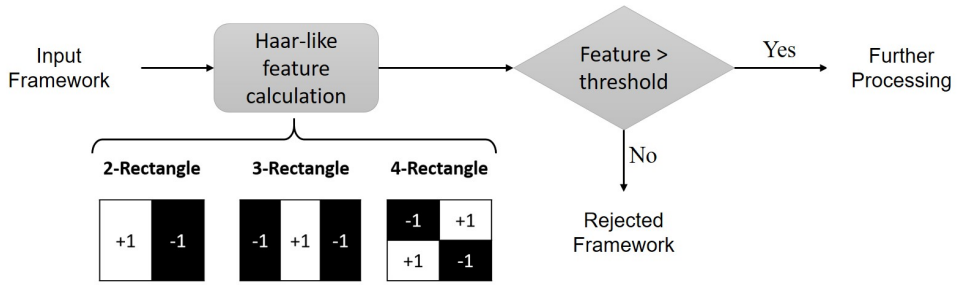



Figure 2.5: Haar-like features

learning, and reinforcement learning. The most popular ML algorithms used as the classifier are Support Vector Machine (SVM) [36], [37], [15], AdaBoost [1], [7], K-Nearest Neighbors (KNN) [38], Naive Bayes [39], and K-Means [18], [40].

Although there are different classification techniques, some essential factors should be considered when a hardware implementation is targeted. First, there is the accuracy performance of the algorithm in hardware implementation using fixed-point precision. Typically, classifiers are suitable for hardware implementation, which results in area-efficient hardware architecture. Support Vector Machine (SVM) and AdaBoost are the most popular and efficient classifiers for different classification applications [36], [37], [15], [41], [42]. Therefore, this subsection reviews these classifiers.

### 2.3.1 Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a supervised learning method introduced in 1998 by Vapnik [6]. The basic idea of SVM is finding a hyperplane in an N-dimensional space with the maximum distance between data points of two classes. As it is often impossible to separate sample points from different classes using hyperplanes, SVM uses a specific mathematical function, called a "kernel", to transform the training data onto a new space in which is possible to find separating hyperplanes. The most common kernels are linear, polynomial, radial basis function (RBF), and sigmoid kernels [43]. There is no advantage in using one kernel rather than the other in general. Depending on the application, the best fit should be determined. It will be discussed in Chapter 4 that we used the linear kernel in this work. Therefore, we only focus on that from now on.

Figure 2.6 shows a general block diagram of the SVM classifier with the linear kernel. As shown in Figure 2.6, SVM consists of two phases: the training phase and the prediction phase. The SVM weights are calculated during from the training phase and used to classify a new sample in the prediction phase. The training phase involves high algorithmic complexity due to the use of a kernel function. However, the training phase does not need to be implemented in hardware, so this is not an issue. The prediction phase requires the multiplication of the input vector and SVM weights. Then a bias value, obtained during the training phase, is added. In the decision block, the value obtained from new data using the previous computations is compared with a threshold. If the value is greater than the threshold, the sample is classified as object-like; otherwise, it is classified as a non-object one. The threshold is obtained empirically and depends on the application.

In a classification system, the input vector that feeds the classifier is a high-dimensional feature vector.

Therefore, the hardware implementation of the prediction phase of the SVM classifier with the linear kernel demands implementing many multipliers that impose a burden on the required resources.

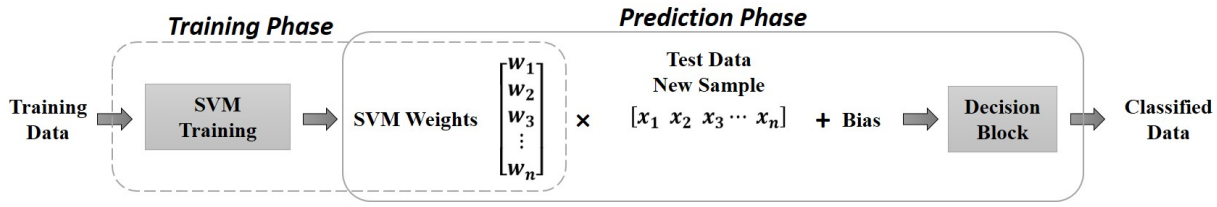



Figure 2.6: General block diagram of the SVM classifier with a linear kernel

The SVM classifier is originally designed as a one-class classifier (known as a binary classifier) to classify between two classes: object-like and non-object. There are also some techniques to solve multi-class problems with the SVM classifier. In general, there are two main approaches for implementing a multi-class SVM. One is based on combining several binary classifiers, while the other one is based on considering all data as one optimization problem [44].

In this work, we used the One-Vs-One (OVO) method. Figure 2.7 shows the general scheme of a multi-class SVM classifier. In this method, the multi-class classifier consists of *n(n-1)/2* independent binary SVM classifiers for $n$ classes, one binary classifier for every possible pair of classes. Each classifier assigns the sample between two classes. The sample is then classified based on a majority voting scheme [44].

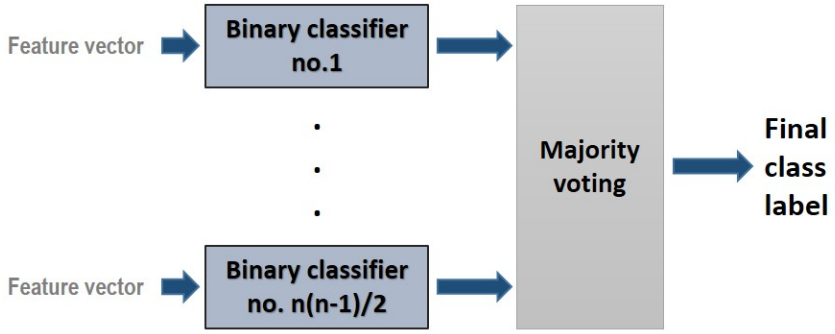


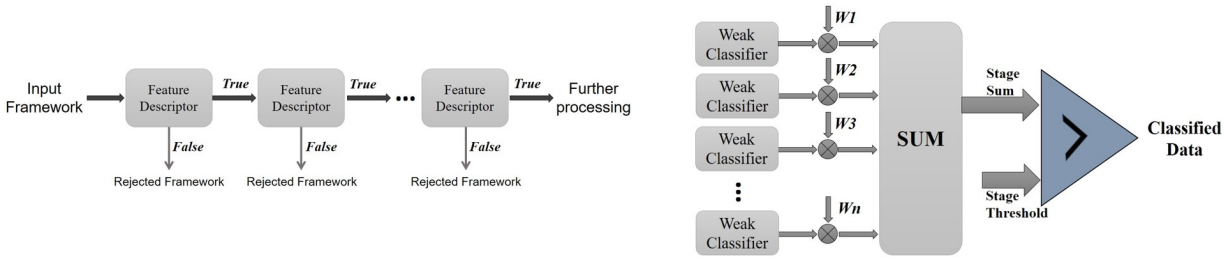

Figure 2.7: General scheme of a One-Vs-One multi-class SVM classifier

### 2.3.2 AdaBoost

Viola and Jones [5] first introduced using AdaBoost as a classifier algorithm for classification applications. The basic idea of AdaBoost is to use a cascade structure of weak classifiers to construct a strong classifier. Weak classifiers are classifiers that have only slightly better performance than a random classifier. The most popular weak classifier used with AdaBoost is based on Haar-like features. As shown in Figure 2.8(a), each weak classifier consists of a group of feature descriptors. If the feature difference exceeds a predefined threshold, the outcome is true, which means the feature exists, and the sample is passed to the next feature descriptor. If not, the output is false, the sample is rejected, and no further processing is done. This process

is illustrated in Figure 2.8(a). The strong classifier is a linear combination of weighted results of the weak classifiers. At the end of a stage, as Figure 2.8(b) illustrates, a stage threshold is used to determine if the frame is an object-like one that can be moved to the next stage or not. The weights, size, and number of the features are generated by AdaBoost and depend on the application and object of interest.

AdaBoost gets some benefits from the cascade structure. First, the weakness of any on weak classifier is compensated. Second, the cascade approach significantly reduces processing time because simpler classifiers effectively reject the majority of non-object frames. Then, more complex classifiers produce a more accurate classification of object-like frames. Moreover, various classification algorithms can be used as weak classifiers in a cascade structure. Besides the advantages, boosting algorithms suffer from higher complexity and longer processing time. This is because they need a considerable number of stages to have a reliable accuracy performance. Therefore, they require many operations that are resource-demanding in the hardware implementation.




(a) Weak Classifier (b) Strong Classifier

Figure 2.8: AdaBoost: (a) Cascade architecture of a weak classifier, (b) Strong classifier of AdaBoost

## 2.4 Field-Programmable Gate Array (FPGA)

The Field-Programmable Gate Array (FPGA) is a semiconductor device that consists of a matrix of configurable logic blocks (CLBs), memories, all other usefull reconfigurable resources. A programmable routing architecture connects CLBs. FPGAs are designed to be configured for any functionality requirement after manufacturing. Therefore, they are widely used for prototyping hardware designs.

As discussed in the previous sections, the feature descriptors and classifiers that are used in a classification system contain many recursive functions, such as operations that should be repeated for every image block. Therefore, they have a highly inherent parallelism structure that is compatible with hardware implementation. The recursive operations can be performed in parallel on hardware to speed up the process compared to software. Moreover, the computational complexity of the algorithms in a classification system, discussed in the previous sections, makes them a good challenge to be solved on a hardware implementation.

There are many published works that propose different approaches to speed up the processing speed [45], [46] or that simplify the implementation of the complex operations on an FPGA [47], [48], [49]. Over the past decade, many works have shown that the FPGA is a well-suited platform for classification applications due to the inner parallelism characteristics, reasonable power consumption, and reconfigurability [18], [19],

[17], [50]. Therefore, we used an FPGA device in this research to implement and evaluate our proposed classification systems.

## 2.5 Feature Combination Benefits in Classification Accuracy

An object can be described by different types of features such as shape, texture, edge and boundary. In an image, the edges and boundaries are areas with strong intensity contrast to their adjacent pixels. Edge and boundary detection play an important role in image classification since objects, and their location, can be estimated by detecting their boundaries. The shape feature is one of the most practical visual features in classification applications because the shape of an object is related to its physical structure, invariant to scale and rotation. A shape descriptor attempts to quantify the object's shape in a way that accords with human intuition. Texture refers to the spatial distribution of intensity levels in an image. Texture features are used in image classification applications to split an image into regions of interest.

There needs to be an accurate description of the object with sufficient features to have an accurate object detection. Different studies have investigated the influence of feature combination on classification systems. These works have an analogous outcome which says complementary features should be combined to achieve better classification accuracy performance [3], [32], [51]. A classification system with a combined feature set is called a dual-feature classification system. A review of the existing works on dual-feature classification systems shows that combining the shape and texture information can better describe the object and improve classification accuracy performance [52], [32].

HOG features are good at representing the local edge and shape information. Haar-like features and LBP features are good for describing texture information [53]. LBP and Haar features can detect regions that are brighter or darker than their immediate surrounding region better than HOG features. In short, HOG features can describe shape better, while LBP and Haar features can describe texture better. For this reason, both LBP and Haar-like features can be an excellent complementary feature set for the HOG features; hence dual HOG-LBP and HOG-Haar-like feature sets can result in a better classification accuracy performance [32].

A literature review of the existing dual-feature classification systems showed they utilized a combination of two of three feature descriptors, including HOG, LBP, and Haar-like feature sets. Therefore, this subsection reviews some previous works investigated combining HOG, LBP, and Haar-like feature sets.

In [32] two independent models are trained for pedestrian detection: an AdaBoost model for Haar-like features and an SVM model for HOG features. For each testing sample, the sample is first evaluated by the AdaBoost model using the Haar-like features. In this test, if the sample is predicted that includes a pedestrian, it is evaluated using the SVM model to make a verification. In other words, object-like samples are double-checked to verify they contain the object of interest. They showed that their proposed approach could achieve a higher detection rate and lower false positive rate than only using the HOG descriptor or only the Haar-like features. Here the detection rate is the proportion of the correctly detected samples, and the false positive rate refers to the negative samples predicted as positive ones. In their system, the false positive and detection rate performance varies depending on the number of cascade stages of the Haar-like features and SVM classifier threshold value. Depending on the number of cascade stages and SVM classifier threshold value, the false positive rate varies between 0.24% and 6.1%, while the detection rate varies between 98.2% and 81.4%. They reported a 95.14% detection rate and 2.36% false positive rate as the optimum performance

of their proposed system.

The authors of [54] investigated the performance of HOG and LBP feature descriptors and Haar-like features for on-road vehicle detection on an 8-core processor. They evaluated the false positive and detection rates of single-feature systems and their combination using the AdaBoost algorithm as the classifier. In their proposed combined system, there is no dual-feature set and they combined the detection results of three single-feature systems. The sample is only accepted as a positive detection if it is detected as a vehicle by at least two single-feature systems, and the bounding boxes have more than 50% overlap. Among the single-feature systems, the LBP feature descriptor has the best results with a detection rate of 88%. After that, Haar-like features with a detection rate of 84% achieves the second best detection performance and then HOG feature descriptor, which has a detection rate of 82%. Using a combination of all three feature descriptors and the AdaBoost cascade classifier, they reported a detection rate of 91%.

The authors of [55] studied the classification accuracy performance of the HOG, Uniform LBP, and HOG-LBP feature descriptors using the SVM classifier for human detection. Their classification system could achieve an 88% detection rate for the HOG feature descriptor, and the LBP feature descriptor showed no reliable performance for human detection. The dual HOG-LBP feature set enhances the classification accuracy performance and reaches a 95% detection rate.

Another work that evaluated the combination of the HOG and LBP feature descriptors on the accuracy performance of the object detection is presented in [56]. They used the SVM classifier and tested three different schemes for combining features. The first scheme is the naive combination that directly concatenates two feature sets into a dual HOG-LBP feature set. The second scheme is Multiple Kernel Learning (MKL), which uses a combination of kernel functions to combine data from different sources [57]. The third scheme, boosting, refers to using a cascade structure to improve the accuracy performance [58]. They reported that the boosting scheme for feature combination has the best accuracy performance in their system. They evaluated the accuracy performance of their proposed system compared to previous similar works for different objects, such as dog, horse, bicycle, and plane. The accuracy performance improvement of their system compared to other existing works varies for different objects. They showed that their proposed dual-feature system could enhance the accuracy performance by an average of 2.2% over 20 categories.

The authors of [59] proposed a human detection system based on a dual HOG-LBP feature set and the SVM classifier. Their proposed model can handle partial occlusion (a portion of the pedestrian is occluded) and includes two kinds of detectors. The first is a global detector to scan the whole image, and the second is a part detector to scan local regions. They investigated the inner product of the SVM weights with the HOG features and observed that the inner product is negative in the case of partial occlusion in the image. For these images, they run the part detector to segment the image into regions with positive and negative inner products. The segmented regions with a negative inner product are considered as the occluded regions. This method improved the accuracy performance for partially occluded people; therefore, the overall detection rate increased. They could achieve a detection rate of 97.9% at False Positive Per Window (FPPW) of $10^{-4}$ on the INRIA dataset, using the HOG-LBP feature set.

All the above-reviewed works are software implementations of feature combination techniques. The authors of [52] presented a hardware implementation of a multi-class object detection system using a HOG-LBP feature set. They implemented the HOG and LBP feature descriptors, as described in Section 2.2, and used the LUT structure of the FPGA to implement complex operations such as division. They reported an 86.2% accuracy for multi-class classification on the MNIST dataset, using the HOG-LBP feature set, but

did not provide details of their experiment.

In summary, it is observed that a classification system with a combined feature set outperforms the single-feature classification system from an accuracy performance point of view. The application-wise challenge of utilizing feature combination technique is finding an efficient approach for combining different feature sets that improves accuracy performance. On the other hand, the hardware-wise challenge of designing a classification system with multiple feature descriptors is hardware resource limitations. Implementing two feature descriptors requires comparably more hardware resources than one feature descriptor. This is the reason that most of the existing works on feature combination utilized software implementation. Therefore, employing an approach that reduces the hardware resource utilization of feature descriptors can provide the capability of implementing an area-efficient dual-feature classification system.

## 2.6 Motivation of Our Approaches

In the previous sections, we presented background on different feature descriptors and classifiers for a feature-based classification system. Among the feature descriptors, the HOG feature descriptor shows a reliable accuracy performance for various applications [4], [54], [60], [61], [60]. Moreover, it shows more capability for optimization on hardware implementation due to its complex operations including square, square root, arctangent, and division.

The literature reviewed on feature combination shows accuracy improvement of the classification system with a dual-feature set compared to a single-feature set. It is discussed that the HOG descriptor is a shape descriptor, and combining the HOG feature set with a texture descriptor enhances the accuracy performance. The above literature review on the feature combination's influence on the classification systems' accuracy shows that both Haar-like features and LBP features are powerful complementary feature sets for the HOG descriptor.

As discussed in Subsection 2.2.3, the feature size and the number of rectangles for each Haar-like feature depends on the application and object of interest. Since we aim to design a classification system based on a new approach that can be applied to different applications and objects, the Haar-like feature is not an appropriate choice for our research. This is because the Haar-like feature should be designed specifically for each application and object of interest. Therefore, we focused on designing a classification system based on the HOG-LBP feature set. We proposed using a binary dual-feature system. To the best of our knowledge, this is a new approach to designing a dual-feature classification system, which will be introduced in Chapter 3. In addition, the LBP descriptor is popular for hardware implementation due to its low computational complexity. Moreover, both the HOG and LBP descriptors are very slow due to being a sliding window algorithm. Therefore, the hardware implementation can speed up their process.

Among classification algorithms, SVM has been shown to give reliable accuracy performance for different classification applications with the HOG descriptor, LBP descriptor, and HOG-LBP dual-feature set [36], [37], [49]. Moreover, the SVM classifier shows capability for hardware design innovation due to its hardware resource-demanding decision function, discussed in Subsection 2.3.1. Besides, the SVM classifier does not need to be designed specifically for the application nor object of interest. It can be designed and implemented once and be used for different applications.

Therefore, we aim to design a classification system consisting of the HOG-LBP feature set followed by an SVM classifier. Details of our proposed classification systems will be introduced in Chapter 3 and their

hardware implementations will be described in Chapter 5.

## 2.7 Literature Review on the Hardware Implementation of Classification Systems

There are many works that describe for hardware implementations of classification systems. In this section we review some of the existing works that proposed different approaches to optimize area utilization, speed or accuracy performance of the classification system. As mentioned in Section 2.6, we aim to design a dual-feature classification system by combining the HOG and LBP descriptors and the SVM classifier. Therefore, this section presents a literature review on hardware implementations of these three algorithms in three following subsections.

### 2.7.1 HOG Descriptor Implementation

As discussed above, the HOG descriptor computation, such as normalization and the gradient magnitude and angle generation, requires complex operations including square, square root, arctangent and division operations that are not area-efficient in hardware implementations. Existing works have utilized different techniques to overcome the computational complexity of these operations. Some of the techniques and simplification methods are discussed in this subsection.

- A common approach to simplify hardware implementation in most of the domains is employing fixed-point arithmetic instead of floating-point. Authors in [29], [2] showed that using fixed-point arithmetic for HOG computation, can achieve accuracy comparable with floating-point arithmetic.
- Parallelizing histogram calculation is another approach which is compatible with the innate parallelism characteristics of FPGA. Authors in [62], [61] and [9] proposed parallelized architectures for cell histogram generation which resulted in fast processing.
- A common approach that has been employed to compute square root, arctangent and division is using Look-up-tables (LUTs). Authors in [29] implemented square root operation using LUTs and in [63] arctangent and division are implemented using LUTs.
- In [29] for the arctangent operation, more specific simplification is applied. The gradient angles generation only requires an arctangent operation to determine the bin of a gradient magnitude. They used a constant value for proportional distribution of gradient magnitudes between bins. Their experimental result shows that the simplification does not degrade detection accuracy.
- The authors of [2] derived a conditional expression to assign gradient magnitudes to bins instead of implementing an arctangent function for gradient angles calculation of the HOG descriptor. They used the relationship between the gradient orientation and the signs of the gradients to derive the conditional expression.
- As the normalization process contains the square, square root, and division operations, its substitution for a simplified process results in simplification and fewer required hardware resources. In [49] the normalization process is replaced with a modified binarization process. Not only does this modification

simplify the normalization process, but also all multiplication operations are replaced by addition operations in the classifier, which is SVM. As a result, these simplifications reduce the hardware resources. The authors of [2] used a similar binarization process to simplify the HOG calculation. Using this technique, they could achieve 1/64 reduction in memory capacity.

- Using the relationship between the gradient orientation and the sign of the differences of luminance, authors in [2] have derived a conditional expression. They obtained a quantized gradient orientation using this conditional expression to avoid calculating the arctangent.
- Due to the histogram calculation and SVM prediction workload, merging these two processes leads to required resource reduction. The work presented in [61] and [9] using this approach to reduce the needed hardware resources.
- Window-based scanning and cell-based scanning are two methods which are used for image scanning in HOG. In the window-based method, windows overlap with an offset of only 1 cell, while in the cell-based scanning method, there is no cell overlap with other cells. Therefore, the memory bandwidth is much more in window-based method [9], [29], [63], [10]. In [61] and [2], cell-based scanning is used to decrease the memory bandwidth.
- Using dual-port block RAM (BRAM) is another technique to accelerate HOG computation, as described in [63].

The above-presented literature review lists some of the techniques existing works utilized to simplify hardware implementation of the HOG descriptor. All the above-listed techniques use a grayscale image with the calculation using decimal arithmetic. As will be introduced in Chapter 4, we utilized binary images to extract HOG features. Thus our classification system is based on the binary arithmetic calculation that results in a fast and area-efficient HOG descriptor, as shown in Subsection 6.1.1.

### 2.7.2 LBP Descriptor Implementation

The LBP is known as a high-performance feature descriptor for local texture classification [34]. As discussed in Subsection 2.2.2, the algorithm simply consists of several comparison operations, so it is very computationally efficient. Therefore, it is widely used for hardware implementation for various applications.

The LBP feature descriptor operation can be divided into three main steps, as shown in Figure 2.9. Due to the simplicity of the LBP algorithm, there is not much room for innovation in its hardware implementation, and most of the existing works are pretty similar in architecture. Therefore, for the literature review of the existing works on the hardware implementation of the LBP feature descriptor, we present common architectures that have been utilized to implement each step.

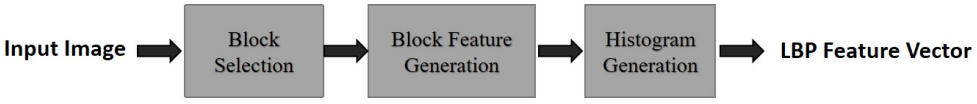



Figure 2.9: General block diagram of the LBP feature descriptor

The primary LBP feature value for a given pixel is made by using a given pixel as a threshold surrounded by a set of equally spaced neighbors (P). Therefore, to start the feature extraction process, an $n \times n$ block

of the input image pixels is selected. Some existing works implemented this step using line buffers [16], [64], [53], and [65]. Reference [30] utilized a RAM to store the input image and two First-In, First-Out (FIFO) memories to pass the input pixels to the next module. We implemented a similar architecture with three FIFO memories in our proposed architecture for the LBP descriptor.

In the second step, the center pixel and the surrounding pixels are compared to generate the block feature. The most common approach for implementing this function is using P comparators to compare the center pixel with the P surrounding pixels, and generate P-bit binary patterns with $2^P$ different values [16], [64], [65], [30], [31]. We adopted this approach in our implemented LBP descriptor module. Reference [53] employed P subtractors to obtain the difference value between the center pixel and the surrounding pixels. As they need the sum of P absolute difference values for other computation purposes, they chose to use subtractors rather than comparators to generate the LBP feature for each pixel.

The final LBP feature vector of an image is a histogram of all the LBP feature values that shows the occurrence frequency of each LBP value. Therefore, after computing all the LBP feature values in an image, the histogram is generated by the concatenation of the $2^P$ or 59 possible LBP values for the basic LBP feature descriptor or the Uniform LBP feature descriptor, respectively.

Among existing works, the most common approach to perform histogram counting is using a map table that is compatible with the LUT architecture of the FPGAs [30], [28], [65]. Reference [64] employed a ping-pong RAM architecture consisting of two sets of RAMs, two selector circuits, and an adder. According to the LBP pattern, the address is searched, and then the adder adds one to the data in that location of the memory.

Reference [31] proposed using the Logic Based Higher Performance Binary Content Addressable Memory (LH-CAM) technique [66] to improve the pattern recognition speed. By replacing RAM with a CAM, LH-CAM takes advantage of the parallel capability of CAM that is compatible with LUT architecture of the FPGAs. LH-CAM is based on the concept of match-lines that has the capability to store one bit and a comparison circuitry to compare the stored bit with the input bit. The number of pattern bits determines the number of match-lines in an LH-CAM. They reported at least 37.12% speed increase and 7.7× fewer LUTs compared to similar works. We employed this architecture as the mapping table to generate the LBP histogram.

As mentioned at the beginning of this subsection, there is not much room for innovation in the hardware implementation of the LBP descriptor. To have an optimum LBP descriptor module, we implemented each sub-module using the best existing architecture among previous works, as described in this subsection.

### 2.7.3 SVM Classifier Implementation

For a classification system, we need to implement the decision function of the SVM classifier. Regardless of the kernel function, which can be linear or nonlinear, the decision function implementation requires the multiplication of the input vector and SVM weights vector. As described in Subsection 2.3.1, the input vector is a high-dimensional feature vector in a classification system, in most cases. Therefore, the hardware implementation of the decision function of the SVM classifier requires implementing many multipliers that are not area-efficient. Many previous works implemented the decision function of the SVM algorithm by implementing multiplier and adder modules that are used repeatedly to complete the decision function calculations [67], [68], [69], [70], [71].

On the other hand, some references in the literature propose new ideas to overcome the complexity of the hardware implementation of the SVM algorithm. This is more vital for the hardware implementation of the nonlinear kernels that are more complex and demand more area resources than the linear kernel.

This subsection reviews some methods that previous works have utilized in the FPGA implementation of the SVM classifier.

- Reference [72] introduces a coordinate rotation digital computer (CORDIC)-like algorithm for computing the decision function of SVM using fixed-point arithmetic. This method, which is known to be a hardware-friendly kernel, is a simplified version of the conventional Gaussian kernel that uses only shift and add operations and does not require multiplications. The main differences compared to the Gaussian kernel is the use of the L1 norm in the Laplacian kernel, and the base change from e to 2. Their experimental results show that their proposed model could compete with the Gaussian kernel in the prediction error rate.

- Reference [73] proposed a configurable multi-class SVM classifier based on "one-versus-one" training strategy. They implemented a module that computes the contribution of each support vector (SV block) based on the CORDIC algorithm, proposed in [72]. SV blocks work in parallel and the number of support vectors and the resolution of the parameters are configurable. In order to evaluate the performance of the hardware-friendly kernel, they estimated the sinc function for two dimensions by randomly choosing 400 input values. They reported an average error of 0.02 between the real value and the estimated output value.

- Another work that utilized the CORDIC algorithm to implement nonlinear kernel is presented in [41]. They also implemented linear kernel and reported a 10.67% reduction in error rate for nonlinear kernel compared to linear one. The error rate improvement was at the cost of 12 times higher number of LUTs and eight times higher registers utilization.

- Reference [74] proposed a systolic array based architecture that uses a multiplierless kernel instead of a conventional vector product kernel to decrease the power consumption of the SVM computations. They used Canonic Signed Digit (CSD) in the multiplierless block to reduce the maximum number of adders. CSD is a number system that represents a floating-point number in 2's complement form that results in 33% fewer non-zero elements than 2's complement. Using CSD form in a hardware implementation of digital signal processing leads to more efficient implementation of add and subtract networks. They investigated the power reduction using the multiplierless kernel compared to vector product kernel and reported 1%, 2.7%, and 3.5% reduction in power consumption for binary linear, binary nonlinear, and multi-class SVM, respectively.

- Another work that utilized a systolic array architecture for matrix multiplication of the multi-class SVM is presented in [42]. Using a systolic array architecture allows more efficient memory management and data transfer mechanisms, and reduces design complexity. They also used the Partial Reconfiguration (PR) scheme of Xilinx for a systolic array implementation of the SVM classifier that results in 3%-5% reduction in power consumption.

Although existing works came up with different ideas to facilitate hardware implementation of the decision function of the SVM, the multiplication of high-dimensional feature vector and SVM weights vector is still

a challenge in the hardware implementation of a classification system. To overcome this challenge, as will be introduced in Chapter 4, we proposed using a binary feature set that replaces all multiplication with addition, which is much more area-efficient and faster.

## 2.8 Summary and Motivations of our Research

As described in this chapter, the classification problem is fundamental in the field of computer vision. The need for real-time classification in most applications and the frequent inability of software-based implementations in achieving real-time capability [7], [8], [9], [10] motivated us to utilize hardware acceleration. It is noted in this chapter that there are two popular approaches to designing a classification system: the feature-based technique and CNN. The feature-based technique is better suited to hardware implementation than CNN. Therefore, we aim to design and implement an area-efficient architecture to meet real-time requirements for classification applications.

This chapter presented the background on the popular feature descriptors and classifiers in designing a feature-based classification system and reviewed existing solutions in accelerating classification systems using hardware. The hardware implementation of a system encounters hardware-related restrictions, such as processing time, resource utilization, and power consumption. This chapter describes the main constraints of the hardware implementations of the various algorithms.

There is no perfect system that meets all of the requirements of various applications. A system should be customized for each application to satisfy the accuracy performance and hardware-related requirements of the specific application. In this research, we adopted a new approach; a binary dual-feature set, for designing a classification system. As a new approach, it should be evaluated for different applications. As discussed in this chapter, some algorithms should be specifically designed for a particular application, and be built upon other system algorithms, such as Haar-like features and the AdaBoost classifier. These algorithms are unsuitable for our system because of their design specifications. As a result, we focused on the HOG and LBP feature descriptors and the SVM classifier that can be designed independently of the application and the other parts of the classification system. Moreover, the HOG descriptor and SVM classifier showed reliable accuracy performance for various applications. Therefore, they are more likely to have reliable accuracy performance in a binary dual-feature classification system, which is our new approach.

A review of the existing works on implementation of HOG and SVM algorithms shows that, although various approaches have been proposed for optimizing their hardware implementation, there is still room for innovation. This is because of their algorithmic complexity and area-demanding hardware implementation. The HOG algorithm needs complex operations, including square, square root, arctangent, and division, and the decision function of the SVM algorithm requires many multiplications that demand a significant amount of hardware resources. On the other hand, reliable accuracy performance is essential for every classification system. Feature combination is one technique to enhance the accuracy of classification systems. A review of the existing dual-feature classification systems shows that the HOG and LBP features are powerful complementary feature sets. Additionally, the LBP descriptor is known as a hardware-friendly feature descriptor due to its low computational complexity. Therefore, we aim to design a binary HOG-LBP feature classification system.

We evaluated our proposed classification system on the FPGA because it is a suitable platform for hardware implementation evaluation due to its inner parallelism capabilities and scalable resource utiliza-

tion. Details of our proposed classification systems will be introduced in Chapter 3 and their hardware implementations will be described in Chapter 5.

# Chapter 3

# Proposed Binary Classification Systems

This dissertation aims to design and implement an area-efficient architecture using binary images to meet real-time requirements for classification applications. As discussed in Chapter 2, the feature-based technique and the CNN are two popular approaches to designing a classification system in the field of computer vision, with the feature-based technique being more suited to the hardware implementation than CNN. As we aim for the hardware implementation, we focus on the feature-based technique in this research.

The main contribution of this research is utilizing a binary feature set to reduce the hardware resource utilization so that more feature descriptors can be implemented in an area-efficient classification system. In this dissertation, we proposed two feature-based classification systems based on using binary feature sets. A binary feature of an image pixel is represented using one bit, while a non-binary feature is represented using more bits. Therefore, implementing an image classification system based on the binary feature set requires fewer storage cells and hardware resources.

The first proposed system is founded on a binary HOG feature set. In general, using a binary feature set leads to an area-efficient and faster architecture; however, it loses accuracy because binary features contain less information than non-binary features. To improve classification accuracy performance, we proposed the second system which uses a dual-feature set. In the second system, binary HOG and LBP feature sets are combined to form a binary dual-feature set to compensate for the accuracy loss.

Section 3.1 introduces the proposed binary single-feature classification system. In Section 3.2, it is described how the binary dual-feature classification system is built based on the binary single-feature classification system. The binary dual-feature system consists of the binary single-feature system and additional modules to extract the second feature set and combine the two feature sets. In each section, a system-level model of the proposed system is first introduced. Then, details of the utilized algorithmic-level techniques and theoretical analysis of their effects on optimization of the hardware-related characteristics, that is, area and speed, are described. Details of the experimental results are presented in Chapter 6.

## 3.1 Binary Single-feature Classification System

Subsections 2.2.1 and 2.3.1 describe the original HOG and SVM algorithms and their constraints in hardware implementation. The significant constraints on the HOG descriptor in hardware implementation are the gradients and normalization computations. These computations need complex operations such as square, square root, and arctangent, which are not area-efficient in the hardware implementation. Moreover, the hardware implementation of the prediction phase of the SVM demands implementing many multipliers that impose a burden on required hardware resources. To overcome these constraints, we proposed two steps of binarization. Figure 3.1 shows the proposed binary single-feature classification system that consists of input image binarization as the preprocessing step, the binary HOG feature descriptor, and the SVM classifier. Two steps of binarization are shown by red arrows in Figure 3.1 and are introduced in the following paragraphs.

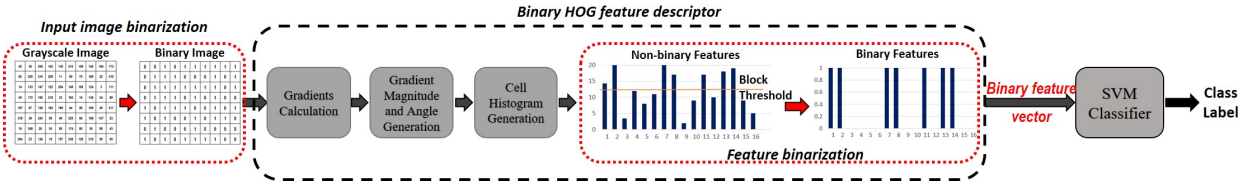



Figure 3.1: The proposed binary single-feature classification system

Reference [47] investigated applying the HOG algorithm to binary images. They implemented three different versions of the HOG algorithm: first, the original HOG algorithm that uses the grayscale image as the input; second, a simplified version of the original HOG that uses binary input images; and third, an improved binary HOG algorithm. The simplified version of the HOG is similar to the original one, with adjustments for using binary images as inputs. However, the improved binary HOG implementation is a customized architecture designed for computation based on an input binary image. The simplified HOG for binary inputs and improved binary HOG algorithm showed around 11% and 98% reduction in hardware resource utilization than the original one, respectively. For accuracy performance evaluation, they measured precision and recall values, as well as the precision-recall curves and their area under curve (AUC). The simplified HOG for input binary images and the improved binary HOG algorithm showed the same accuracy performance for binary images. They reported a 0.02 reduction in precision, 0.05 drop in recall, and 0.01 reduction in AUC for the improved binary HOG algorithm compared to the original HOG for grayscale input images. They did not provide any analysis of speed optimization of their binary HOG descriptor. We used their improved binary HOG algorithm in our proposed systems because from our survey, it is already a highly optimized binary HOG algorithm and it will be difficult to gain further optimization. Details of the binary HOG algorithm will be discussed in Subsection 3.1.1.

Reference [49] replaced the normalization step of the HOG algorithm with binarization. They used the original HOG descriptor to extract the HOG feature vector but replaced normalization with binarization. They used the mean of the block features as the threshold for each block histogram binarization. If the block feature is greater than the threshold, it is set to 1; otherwise, it is set to 0. This replacement results in a binary HOG feature vector. As they used a linear SVM classifier, the classification process consists of the

SVM weights and HOG features product. With the binary HOG feature vector, the classification process can be performed, instead of multiplication, by adding the SVM weights that correspond to binary HOG features having a value of 1. Reference [2] also replaced the normalization step of the HOG algorithm with binarization; however, they used the AdaBoost classifier in their human detection system. Reported results in [49] showed that their human detection system could achieve 5 times fewer resources than [2], with 2.6 times faster detection speed.

**Our Proposed Binary HOG Descriptor:** In our binary single-feature classification system, we proposed combining the techniques mentioned above: input image binarization and feature binarization. As shown in Figure 3.1, it can be interpreted as two steps of binarization. The first step is the input image binarization; if the grayscale image pixel is greater than a predefined, global threshold, the binary pixel is set to 1; otherwise, it is set to 0. The threshold has been obtained empirically by averaging grayscale pixel values for randomly selected images from the dataset of interest. This binarization step simplifies the hardware implementation of the HOG feature descriptor since it removes complex operations of the original HOG algorithm. This will be discussed in Subsection 3.1.1.

The second binarization step replaces the HOG descriptor's normalization step with binarization. To do so, we used the mean of block features as the threshold for each block feature binarization. If the block feature is greater than the threshold, the binary feature is set to 1; otherwise, it is set to 0. As will be discussed in Subsection 3.1.1, although features are extracted from the binary image, the output of cell histogram generation is a non-binary feature vector. Applying feature binarization at this point yields binary features. Feeding a binary feature vector to the SVM classifier reduces the hardware resource utilization of the SVM module. Details of the hardware resource reduction will be discussed in Chapter 5.

### 3.1.1 Binary HOG Descriptor vs. Original HOG Descriptor

In this work, we referred to the presented HOG algorithm in [4] as the original HOG descriptor. We called applying the original HOG descriptor on binary images as the binary HOG descriptor. The binary HOG descriptor has the same algorithmic processing steps as the original HOG descriptor, including gradients calculation, gradient magnitude and angle generation, and cell histogram generation. However, there are significant differences in the hardware implementation due to the use of binary images. Figure 3.2 illustrates the data flow of the binary HOG descriptor and original HOG descriptor. This subsection describes how each processing step of the binary HOG descriptor is different from the original HOG descriptor. The HOG algorithm refers to both the original and binary HOG descriptors in the following parts.

A theoretical analysis of the area reduction and speed optimization is included in each part of the following comparison between the original and binary HOG descriptors. Accuracy performance is application-dependent, and therefore, it is not possible to do theoretical analysis for each step separately. A brief accuracy performance discussion is presented at the end of this subsection, and a detailed evaluation is presented in Chapter 6.

- **Gradients Calculation:**

  The first step of the HOG algorithm is the gradient calculation. Reference [4] showed that a [-1;0;1] gradient filter has the best detection accuracy performance for the original HOG descriptor. Utilizing

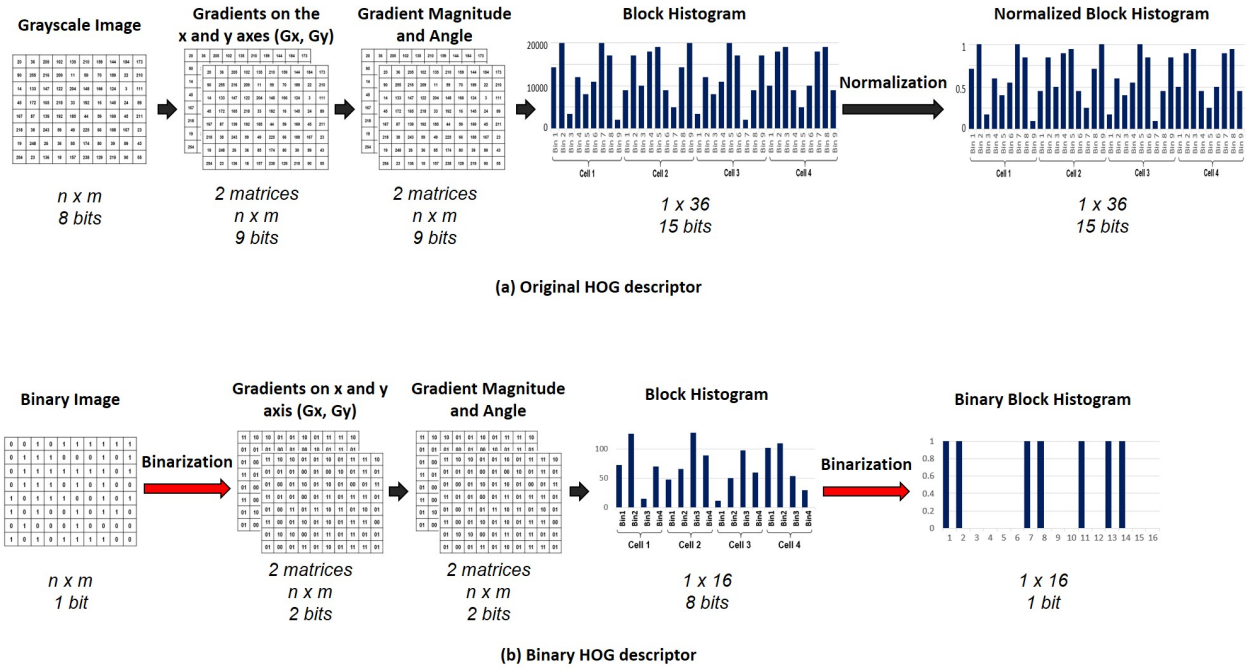



Figure 3.2: Data flow of (a) original HOG descriptor vs. (b) binary HOG descriptor

the [-1;0;1] gradient filter to calculate gradients on the x and y axes ($G_x$ and $G_y$) for each pixel of the input image (I(r,c)) is based on the following two formulas:

$$G_x(r,c) = I(r,c+1) - I(r,c-1) \tag{3.1}$$

$$G_y(r,c) = I(r+1,c) - I(r-1,c) \tag{3.2}$$

where $r$, $c$ refer to rows and columns, respectively. Therefore, $G_x$ and $G_y$ calculation need the subtraction of two rows or columns of the input image, respectively. A grayscale pixel is represented by 8 bits (values are in the range of 0 to 255), so $G_x$ and $G_y$ are 9 bits (8 + sign bit) for the original HOG descriptor. However, a binary pixel is represented by 1 bit; therefore, $G_x$ and $G_y$ can be represented by 2 bits for the binary HOG descriptor (the possible values for $G_x$ and $G_y$ are 0, 1, -1). Therefore, gradients calculation of the original HOG descriptor needs 9-bit subtractors, while gradient calculations of the binary HOG descriptor only needs 2-bit subtractors.

To have a fair comparison of area optimization resulting from the use of the binary HOG descriptor than the original HOG descriptor, we estimated the required resources for the hardware implementation of the original HOG descriptor using our proposed architecture for the binary HOG descriptor. To this end, we investigated the required resources to calculate one row of $G_x$ and one column of $G_y$ in parallel. For an image of $n \times m$ pixels, we need $m$ 9-bit subtractors for the original HOG descriptor and $m$ 2-bit subtractors for the binary HOG descriptor to calculate $G_x$ for one entire row in parallel. Similarly, we need $n$ 9-bit subtractors for the original HOG descriptor and $n$ 2-bit subtractors for the binary HOG descriptor to calculate $G_y$ for one entire column in parallel. Thus, there needs to be

*(m+n)* 9-bit subtractors for the original HOG descriptor and *(m+n)* 2-bit subtractors for the binary HOG descriptor, to calculate one row of $G_x$ and one column of $G_y$ in parallel. Moreover, gradients need to be stored for subsequent processing. Gradient storage for the binary HOG descriptor requires two matrices of size *(n × m)* 2 bits to store $G_x$ and $G_y$ values, as shown in Figure 3.2.(b), while the original HOG descriptor requires two matrices of size *(n × m)* 9 bits, as shown in Figure 3.2.(a).

In short, the gradient calculations and storage of the binary HOG descriptor are more area-efficient than in the original HOG descriptor. The original HOG descriptor would require 9-bit subtractors for computations, while the binary HOG descriptor requires 2-bit subtractors. Moreover, the binary HOG descriptor saves $2 \times n \times m \times 7$ bits compared to the original HOG descriptor for storage matrices. The gradients calculation of the binary HOG descriptor has no speed benefit compared to the original HOG descriptor because the computation is a simple subtraction in both of them.

- **Gradient Magnitude and Angle Generation:**

  The magnitude and angle of each pixel is calculated using $G_x$ and $G_y$ as follows:

$$Magnitude(r,c) = \sqrt{G_x(r,c)^2 + G_y(r,c)^2} \tag{3.3}$$

$$Angle(r,c) = |tan^{-1}\frac{G_y(r,c)}{G_x(r,c)}| \tag{3.4}$$

  It can be seen that the original HOG descriptor needs square, square root and arctangent operations for the computation of gradient magnitude and angle. These operations are not area-efficient in the hardware implementation. However, the gradient magnitude and angle computation for binary images can be implemented using two simple look-up tables (LUT), as shown in Table 3.1, similar to what [47] has done. The disadvantage of this hardware implementation simplification is loss of accuracy.

Table 3.1: Gradient magnitude and angle lookup table

| $G_x$ | $G_y$ | Magnitude | Angle |
|---|---|---|---|
| 00 | 00 | 00 | 00 |
| 00 | 01 | 01 | 10 |
| 00 | 11 | 01 | 10 |
| 01 | 00 | 01 | 00 |
| 01 | 01 | 10 | 01 |
| 01 | 11 | 10 | 11 |
| 11 | 00 | 01 | 00 |
| 11 | 01 | 10 | 11 |
| 11 | 11 | 10 | 01 |

The possible values for the gradient magnitudes of a binary pixel are 0, 1, and $\sqrt{2}$, which can be stored in 2-bit registers, which are designated as 00, 01, and 10, as shown in Table 3.1. The gradient magnitude values of a grayscale pixel are in the range of 0 to $255\sqrt{2}$ and need 9-bit registers for storage. Moreover, in the case of a grayscale image, gradient angles are real values between 0°and 180°and need 9 bits to represent (8 bits for the value + 1 bit for the sign). For a binary image, gradient angles have only four possible values: 0, +45, -45, and 90, designated as 00, 01, 10, and 11, as shown in Table

3.1. Therefore, the gradient magnitude and angle storage for the binary HOG descriptor requires two matrices of $n \times m$ 2 bits (one matrix for gradient magnitudes and one for gradient angles), as shown in Figure 3.2.(b), while the original HOG descriptor requires two matrices of $n \times m$ 9 bits, as shown in Figure 3.2.(a).

In summary, the binary HOG descriptor replaces complex operations of gradient magnitude and angle computation, including square, square root, and arctangent operations, with simple LUTs. This replacement removes complex operations which reduces hardware resource utilization and saves $2 \times n \times m \times 7$ bits for storage matrices. Moreover, utilizing LUTs for the gradient magnitude and angle computation speeds up the process compared to complex operations. We do not have any estimation of the computation acceleration and area optimization resulting from this replacement because the acceleration and area optimization depend on the hardware implementation of gradient magnitude and angle generation for the original HOG descriptor. As surveyed in Subsection 2.7.1, previous works utilized different techniques for hardware implementation of gradient magnitude and angle generation for the original HOG descriptor, but no one has reported resource utilization and processing time of gradient magnitude and angle generation. However, the overall acceleration and area optimization of the binary HOG descriptor compared to the original HOG descriptor will be discussed at the end of this subsection.

- **Cell Histogram Generation:**

  In order to generate the cell histogram, gradient magnitudes are categorized into orientation bins. The orientation bins are regions evenly spaced over 0°-180°. Gradient magnitudes are first assigned to one of the bins based on their angle value, and then the assigned magnitudes to each bin are accumulated to generate the histogram.

  As discussed in [4], for the original HOG descriptor, increasing the number of bins improves accuracy performance considerably up to nine bins, but it does not make much difference for a larger number of bins. So, most existing works use nine bins for cell histogram generation. However, the histogram for a binary image is computed based on four bins because gradient angles have only four possible values: 0, +45, -45, and 90, as discussed. The $CellHistogramSize$ is proportional to the number of bins, nine and four bins for original and binary HOG descriptors, respectively. Accumulating $2^n$ numbers adds $n$ bits to the sum. For a cell of $8 \times 8$ pixels, there are 64 magnitude values in each cell. In the worst case, all the 64 magnitudes are assigned to one bin; therefore, 64 values ($2^6$) should be accumulated. As mentioned, magnitude values in the original and binary HOG descriptors are represented by 9 and 2 bits, respectively. Therefore, to generate a cell histogram, the original HOG descriptor needs nine 15-bit accumulators (9+6), while the binary HOG descriptor needs four 8-bit accumulators (2+6). Details of the computation will be discussed in Chapter 5.

  A block histogram is the concatenation of the histograms of the block cells; thus, the block histogram size is $BlockSize \times CellHistogramSize$. Therefore, the block histogram of the original and binary HOG descriptors for a $BlockSize$ of $2 \times 2$ are $1 \times 36$ and $1 \times 16$, respectively, as shown in Figure 3.2. The final block histogram of the original HOG is the normalized block histogram which is a non-binary vector of size $1 \times 36$, as shown in Figure 3.2(a). However, the final block histogram of the binary HOG descriptor is a binary vector of size $1 \times 16$, as shown in Figure 3.2(b). As a result, the block histogram

vector size, which is proportional to the number of bins, in the binary HOG descriptor is 4/9 = 44% of the block histogram vector size in the original HOG descriptor.

To sum up, block histogram generation and storage of the binary HOG descriptor are more area-efficient than the original HOG descriptor. The original HOG descriptor would require 15-bit accumulators for computations, while the binary HOG descriptor requires only 8-bit accumulators. In addition, the binary HOG descriptor saves twenty 7 bits compared to the original HOG descriptor for storage matrices. There is no speed benefit in cell histogram generation of the binary HOG descriptor compared to the original HOG descriptor because both consist of the simple accumulation operation.

- **Normalization Replacement by Binarization:**

  Reference [4] evaluated four block normalization methods for the original HOG descriptor and showed that the L2-norm has the best accuracy performance. This process is performed using,

$$v = \frac{V_k}{\sqrt{||V_k||_2 + \epsilon}} \tag{3.5}$$

$$||V_k||_2 = \sqrt{V_1^2 + V_2^2 + ... + V_n^2} \tag{3.6}$$

  where $\epsilon$ is a small constant to prevent the denominator from ever being 0, and where $v$ and $V_k$ are the normalized and unnormalized feature vectors, respectively. As Equation 3.5 shows, the normalization process involves square, square root, and division operations. These complex operations are not area-efficient in the hardware implementation.

  According to the HOG algorithm, the output vector of the cell histogram generation is the accumulation of the gradient magnitudes. It can be seen in Figure 3.2 that the block histogram is a non-binary vector for both binary and original HOG descriptors. The only difference is in the values because the original HOG descriptor gradient magnitudes are in the range of 0 to $255\sqrt{2}$ while the binary HOG gradient magnitudes have only three possible values: 0, 1, $\sqrt{2}$. This difference in the gradient magnitude values results in the difference in the non-binary feature values and thus in the number of required bits for the storage, as shown in Figure 3.2. As a result, there is no difference in the computation complexity of the normalization process between the original HOG descriptor and binary HOG descriptor because of non-binary features. Since implementing square, square root, and division operations is not area-efficient in the hardware implementation, the block normalization process remains a constraint. We replaced the normalization process with a binarization process to overcome this constraint, as shown in Figure 3.2.(b). The binarization process simply compares each non-binary feature with a threshold to obtain the binary feature. We used the mean of block features as the threshold for each block histogram binarization, similar to what [49] has done.

  It will be shown in Subsection 6.1.2 that implementing binarization results in 19% and 55% reduction in the LUTs and processing time for the binary HOG descriptor module, respectively, compared to implementing normalization. Details of the computation will be discussed in Chapter 5.

It has been discussed how the binary HOG descriptor reduces the hardware resource utilization and speeds up the process by simplifying the computations compared to the original HOG descriptor. The hardware resource reduction resulting from each step has been analyzed theoretically. We compared the

FPGA implementation characteristics of our proposed binary HOG descriptor with existing solutions in Subsection 6.1.1. As hardware resource utilization depends on different parameters, such as the dimension of the feature space and image size, it is not straightforward to compare different architectures fairly. However, our hardware implementation of the binary HOG descriptor could achieve the fastest and the most area-efficient architecture among existing works because of the two steps of binarization.

As accuracy performance is application-dependent, making a fair comparison between different architectures is also not straightforward. Reference [47] investigated the accuracy performance of the binary HOG descriptor for car detection applications. They reported a 0.02 reduction in precision for the binary HOG descriptor compared to the original one. We provided a discussion on the accuracy performance of our binary classification system in Chapter 6. We employed the feature combination of HOG and LBP to improve accuracy performance.

The final HOG feature vector is the concatenation of the block histograms. Therefore, the size of the HOG feature vector is $NumberOfBlocks \times BlockHistogramSize$. It will be described in Section 5.1 that the number of blocks varies based on the image size, blocks overlap, and block size. The point is that the size of the final HOG feature vector is proportional to the block histogram size. As discussed, the size of the block histogram is proportional to the number of bins in the cell histogram. As a result, the size of the final HOG feature vector in the binary HOG descriptor is $4/9 = 44\%$ of the feature vector size in the original HOG descriptor. From the hardware point of view, the fewer the features, the lesser the processing time and the fewer the resources required in the subsequent module because the classification process is performed for every feature. Moreover, the binarization process generates a binary feature vector, thus, simplifying the SVM module's hardware implementation. This is discussed in more detail in Subsection 3.1.2.

### 3.1.2 SVM Classifier with Binary HOG Descriptor

As discussed in Subsection 2.3.1, the prediction phase of the SVM algorithm requires the multiplication of the input vector and SVM weights. In a feature-based classification system, the input vector is a high-dimensional feature vector extracted from the input image by the feature descriptor and fed to the classifier. Figure 3.3 illustrates the SVM weights multiplication data format for a non-binary feature vector and a binary feature vector. As shown in Figure 3.3(a), every feature vector element is multiplied by an SVM weight. Therefore, the hardware implementation of the prediction phase demands implementing many multipliers that impose a burden on the required resources. It has been discussed in the previous subsection that the feature vector size in the binary HOG descriptor is 44% of the feature vector size in the original HOG descriptor. Consequently, using the binary HOG descriptor results in 56% fewer operations in the SVM classifier module compared to the original HOG descriptor. Moreover, as shown in Figure 3.3(b), all the multiplications are replaced with additions for a binary feature vector. For example, for sizes shown in Figure 3.2, 36 multipliers are needed in the original HOG descriptor, while 16 adders are required in the binary HOG descriptor to classify one block feature set in parallel. To implement an $M \times N$-bit multiplier in hardware, we need $M \times N$ binary multiplications, equivalent to a logical AND, and $N - 1 \times M + 1$-bit addition operations [75]. Therefore, replacing one $N$-bit multiplication with $N$-bit addition saves $N - 2$ adders and $M \times N$ AND gates on the hardware implementation. As a result, feeding a binary feature set to the SVM classifier not only reduces the number of required operations in the SVM classifier but also simplifies the hardware implementation of the SVM classifier.

$$[x_1\ x_2\ x_3 \cdots x_n] \quad \times \quad \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} \quad = \quad x_1 \times w_1 + x_2 \times w_2 + x_3 \times w_3 + \cdots + x_n \times w_n$$

*Non-binary feature vector* *SVM weights*

**(a) SVM weights multiplication with a non-binary feature vector**

$$[1\ 0\ 1\ 0 \ldots 1] \quad \times \quad \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} \quad = \quad w_1 + w_3 + \cdots + w_n$$

*Binary feature vector* *SVM weights*

**(b) SVM weights multiplication with a binary feature vector**

Figure 3.3: SVM weights multiplication data format for (a) non-binary feature vector (b) binary feature vector

It will be shown in Subsection 6.1.2 that utilizing a binary feature set results in a 4.6 times reduction in the required number of LUTs for the SVM classifier in our hardware implementation. It will be discussed in detail in Chapter 5 that the SVM weights accumulation is performed in parallel with the HOG descriptor in our hardware implementation. Therefore, the SVM weights accumulation process has no effect on the processing time.

## 3.2 Binary Dual-feature Classification System

Although utilizing a binary HOG descriptor leads to an area-efficient architecture, it happens at the cost of accuracy loss. The presented literature review in Section 2.5 shows that HOG and LBP features are powerful complementary feature sets. Therefore, we proposed a binary dual-feature classification system that combines binary HOG and LBP feature sets to improve the classification accuracy. Accuracy improvement of the binary dual-feature system compared to the binary single-feature system is discussed in Chapter 6.

Figure 3.4 shows a system-level block diagram of the proposed binary dual-feature classification system. As can be seen, this system consists of the binary single-feature system and two additional blocks: LBP feature descriptor to extract LBP features and binarization for LBP features to generate a binary feature set. Besides, there are differences in the classifier module in combining HOG and LBP feature sets before the classification process.

In contrast to the HOG feature set, the LBP feature set is extracted from a grayscale image in our binary dual-feature classification system. Our experiments showed that extracting LBP features from binary images

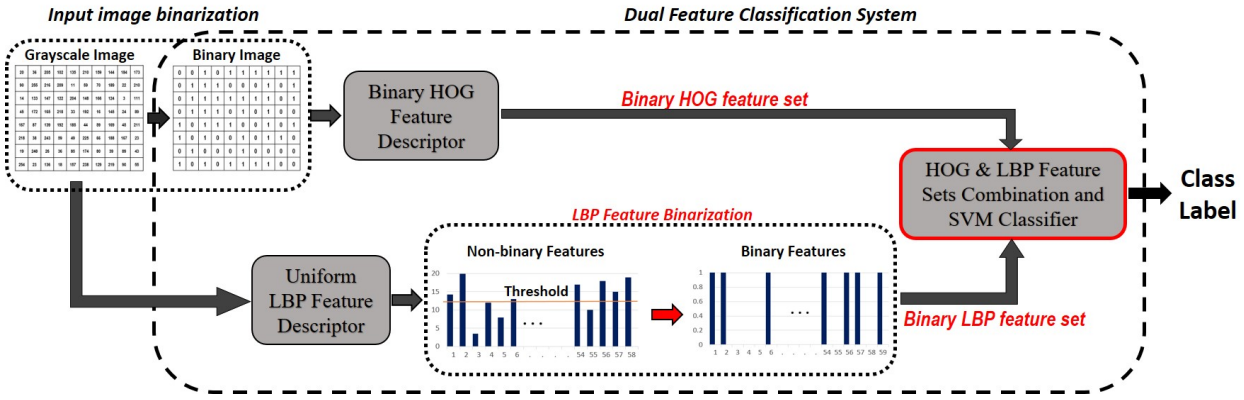



Figure 3.4: The proposed binary dual-feature classification system

does not improve the accuracy performance of the binary HOG descriptor in the binary dual-feature system. In addition, extracting LBP features from a binary image does not simplify hardware implementation as the binary HOG descriptor does. This is because of the simplicity of the LBP algorithm. The first step of the LBP is to compare a given pixel with a set of equally spaced neighbors. A comparator module performs this operation in the hardware implementation. It needs a 1-bit comparator for a binary image, while an 8-bit comparator is required for a grayscale image. This is the only difference between using binary and grayscale images in the hardware implementation. Therefore, in our proposed binary dual-feature classification system, HOG features are extracted from the binary image, and LBP features are extracted from the corresponding grayscale image.

We used the Uniform LBP algorithm in the LBP feature descriptor module of our binary dual-feature system. As discussed in Subsection 2.2.2, the main drawbacks of the non-uniform LBP algorithm are sensitivity to rotation and an exponential increase in the feature size with increasing number of neighbors. The Uniform LBP algorithm is a rotation invariant descriptor, and for an 8-bit LBP feature, the length of the feature vector is 59, instead of $2^8$ for non-uniform LBP. An LBP feature is considered uniform-2 if the number of transitions between 0's and 1's in the binary pattern is at most two 0-1 or 1-0 transitions. For example, 00001000 (2 transitions) is a uniform-2 pattern, but 01010001 (5 transitions) is not. For an 8-bit LBP feature, there are 58 uniform-2 patterns, as shown in Table 3.2. To form the LBP histogram, a separate bin is considered for every uniform-2 pattern, and there is one single bin for all non-uniform patterns. Therefore, the size of the LBP feature vector is 59 for an 8-bit LBP feature, regardless of the image size.

As reviewed in Subsection 2.2.1, a normalization step is necessary to scale the features before classification. We described in Subsection 3.1.1 that the normalization step of the HOG algorithm is replaced with binarization in our proposed binary HOG descriptor. The same replacement is applied to the LBP descriptor in our binary dual-feature system. To the best of our knowledge, no previous work utilized a binary LBP feature set. One of the contributions of this research is investigating the accuracy performance improvement by combining the binary LBP feature set with the binary HOG feature set. The binarization process for the LBP descriptor is similar to the HOG descriptor, as shown in Figure 3.4. The threshold value has been obtained empirically by averaging the LBP feature values for randomly selected images from datasets in our experiments. If the non-binary feature is greater than the threshold, the binary feature is set to 1; otherwise,

Table 3.2: The 58 possible uniform-2 patterns of the Uniform LBP descriptor for an 8-bit LBP feature

| | | | | |
|---|---|---|---|---|
| 00000000 | 00000001 | 00000010 | 00000011 | 00000100 |
| 00000110 | 00000111 | 00001000 | 00001100 | 00001110 |
| 00001111 | 00010000 | 00011000 | 00011100 | 00011110 |
| 00011111 | 00100000 | 00110000 | 00111000 | 00111100 |
| 00111110 | 00111111 | 01000000 | 01100000 | 01110000 |
| 01111000 | 01111100 | 01111110 | 01111111 | 10000000 |
| 10000001 | 10000011 | 10000111 | 10001111 | 10011111 |
| 10111111 | 11000000 | 11000001 | 11000011 | 11000111 |
| 11001111 | 11011111 | 11100000 | 11100001 | 11100011 |
| 11100111 | 11101111 | 11110000 | 11110001 | 11110011 |
| 11110111 | 11111000 | 11111001 | 11111011 | 11111100 |
| 11111101 | 11111110 | 11111111 | | |

it is set to 0. Details of the hardware implementation will be described in Subsection 5.2.2.

As reviewed in Section 2.5, in a HOG-LBP dual-feature system, the HOG and LBP features are first concatenated to create the dual-feature vector and then are fed to the classifier [52], [76], [56]. In our binary dual-feature system, we used the feature concatenation method to combine feature sets, but there is no HOG-LBP dual-feature vector in our proposed architecture. Instead, the extracted SVM weights vector is split into two separate weight vectors for the corresponding weights to the HOG and LBP features, as shown in Figure 3.5. The SVM weights accumulation is performed on the HOG and LBP features separately; then they are added together to combine the feature sets, as shown in Figure 3.6. That is why the HOG and LBP feature combination and SVM classifier are shown as one single module in Figure 3.4. Our proposed feature combination method in our binary dual-feature classification system implementation will be discussed in detail in Section 5.2.

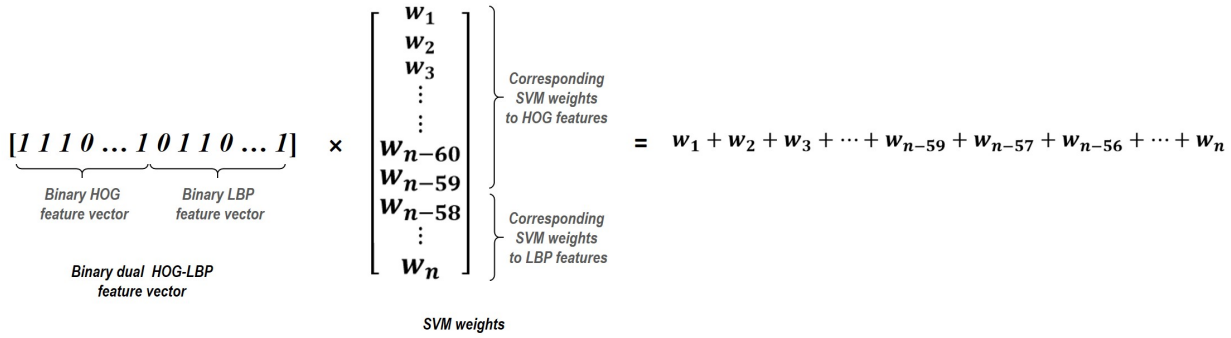



Figure 3.5: SVM weights multiplication with a binary HOG-LBP feature vector

As described above and shown in Figure 3.4, LBP features are extracted from grayscale images while HOG features are extracted from binary images. Therefore, two separate preprocessing modules are necessary for HOG and LBP feature descriptors in our binary dual-feature system. This increases hardware resource utilization for preprocessing compared to the single-feature system.

It will be discussed in detail in Chapter 5 that the processing time of our binary dual-feature system is the same as the binary single-feature system.

It is because of the parallel execution of the HOG and LBP feature descriptors and their preprocessing processes. Besides, feature combination is performed in parallel with the classification process in our proposed feature combination method, discussed above, thereby making no difference in the processing time between the two systems. It will be shown in Chapter 6 that combining the binary LBP feature set with the binary HOG feature set improves the accuracy performance by $3.5\% - 5.2\%$.

We introduced our binary single-feature and dual-feature classification systems in this chapter. As described in Section 2.3, the SVM classifier can be used to solve both binary (one-class) and multi-class classification problems. Since we used the SVM classifier in our system, both systems could be used as binary and multi-class classifiers. It will be described in Chapters 5 and 6 that we implemented and evaluated our binary single-feature and dual-feature classification systems for both binary classification for human detection and multi-class classification.

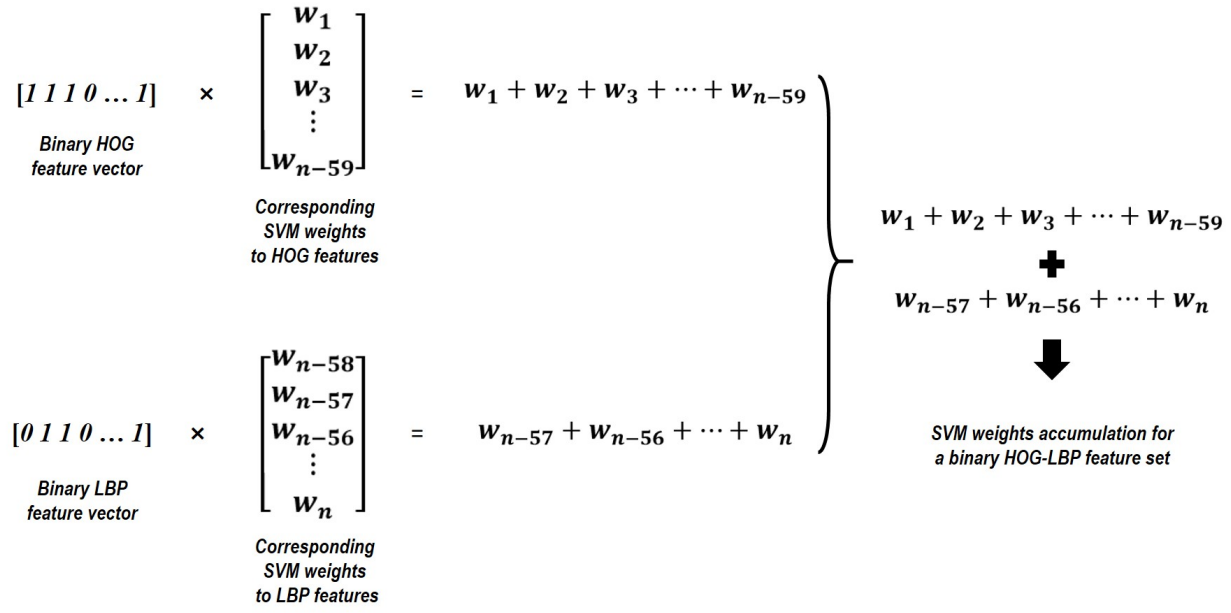



Figure 3.6: Our proposed feature combination method in our binary dual-feature classification system

# Chapter 4

# Accuracy Measurement Methodology and Overview of Results

As discussed in Section 2.1, there are many different applications in the field of computer vision, such as object detection and image classification. Each application has particular specifications and requirements from different points of view. Each application needs satisfactory and reliable accuracy performance. On the other hand, a hardware implementation of a system encounters hardware-related restrictions such as processing speed, resource utilization, and power consumption. There does not exist a perfect system that meets the requirements of all different applications. Therefore, a system should be customized for each application to satisfy its hardware-related and accuracy performance requirements.

Accuracy is the most critical requirement that a system must satisfy. The system is worthless if it does not meet the accuracy performance requirement of the target application regardless of how area-efficient or fast it is. Therefore, before presenting details of the hardware implementation in the following chapter, we present the methodology of our experiments and an overview of the accuracy performance of the proposed binary feature-based systems in this chapter. Since our proposed binary feature-based technique is a new approach to designing a feature-based system, we investigated its accuracy performance for two different applications and measured the accuracy on different datasets to find the application that is best fitted with the proposed systems' specifications.

First, we review the definition of various problems in the field of computer vision. "Object detection" refers to specifying the location of one or more similar or different objects in an image by drawing a bounding box around them. "Classification" categorizes images containing variant objects based on the predefined classes; one class is assigned to each object. Classification can be done as binary classification or multi-class classification.

We introduced our proposed binary single-feature and dual-feature systems in Chapter 3. Both proposed binary feature-based systems are evaluated for solving the "object detection" and "multi-class classification" problems in this research. Section 4.1 focuses on an experiment on the object detection problem. It describes the experiment's methodology to evaluate how our proposed binary single-feature and dual-feature systems perform for human detection. Section 4.2 focuses on experiments on the multi-class classification problem. It describes the methodology of experiments to evaluate the performance of our proposed binary single-

feature and dual-feature systems for multi-class classification. An overview of accuracy performance results is presented in this chapter, to clarify the reason that we only implemented multi-class classification in hardware, and details of the accuracy performance evaluation will be discussed in Chapter 6.

## 4.1 Object Detection System

Reference [4] studied the performance of the original HOG descriptor for human detection and showed that it considerably outperforms other feature sets. They used the SVM algorithm as the classifier in their human detection system. They produced the INRIA person dataset and used it to evaluate the accuracy performance of their proposed human detection system. INRIA person dataset is a popular dataset and the presented work in [4] is the primary reference of existing works to evaluate accuracy performance of the human detection system. As described in Chapter 3, we utilized the HOG feature descriptor and the SVM classifier in our proposed binary single-feature and dual-feature systems. Thus, [4] is a good reference, with the original form of the HOG descriptor and the SVM classifier, to evaluate and compare the accuracy performance of our binary single-feature and dual-feature systems. Therefore, we experimented with the INRIA person dataset and compared our system performance with reported results in [4].

In this section, we first introduce INRIA person dataset in Subsection 4.1.1. After that, Subsection 4.1.2 describes setting for the experiment. Then, Subsection 4.1.3 provides methodology of experiments. Finally, an overview of accuracy performance will be presented in Subsection 4.1.4.

### 4.1.1 Dataset

The INRIA person dataset [77] was introduced by Navneet Dalal in 2005 [4]. It contains separate positive and negative sets for training and testing. They collected images taken by a digital camera in various places and under various weather conditions. They also added some images from the Graz-01 dataset [78]. Positive images contain at least one person. People are always upright with some partial occlusions. There is a wide range of poses, clothing, appearance, background, and illumination for people. Figure 4.1 shows some examples of positive samples of the INRIA person dataset. The training set contains 614 positive samples, including 1,208 people and their right reflection (2,416 images in total). The positive test set contains 288 images, including 1,126 people.

The negative training and test sets include 1,218 and 453 person-free images, respectively. The negative samples include a wide range of natural scenes, such as mountains, beaches, buildings, animals, and also objects, such as motorbikes, cars, furniture, and utensils. Figure 4.2 shows some examples of negative samples of the INRIA person dataset.

Reference [4] claimed that the INRIA person dataset is more challenging than the often-used MIT dataset [79] for human detection because of a broad variety of backgrounds, poses, and clothing. For this reason, the INRIA person dataset is a popular and commonly-used dataset for accuracy performance evaluation of human detection [37], [49], [61], [62], [29].

### 4.1.2 Setting

**HOG Feature Descriptor:** Reference [4] investigated the impact of the original HOG descriptor parameter values on the overall accuracy of their human detection system. They reported that the following

Figure 4.1: Full image positive examples from the INRIA person dataset [4]

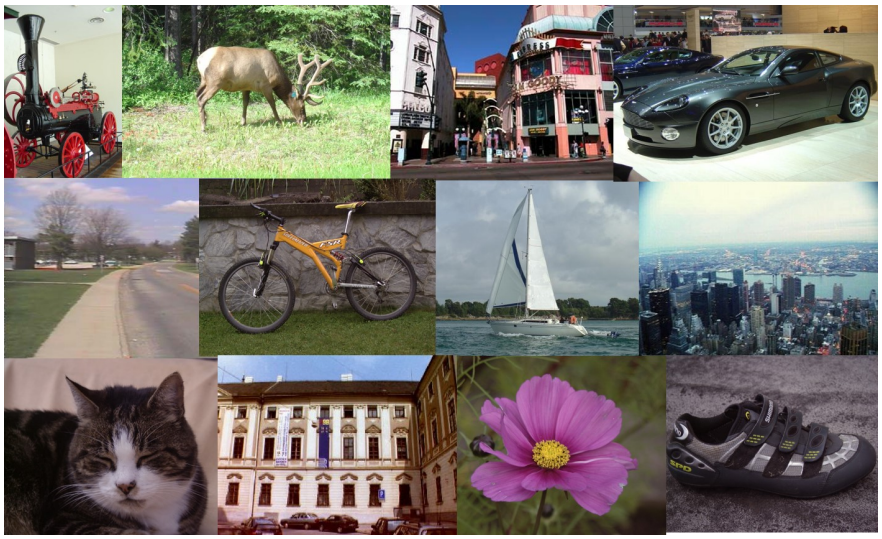


Figure 4.2: Person-free examples from the INRIA person dataset [4]

configurations have the best detection performance results: a [-1,0,1] gradient filter, nine orientation bins in $0° - 180°$ for linear gradient, a stride (block overlap) of 8 pixels, $64 \times 128$ detection window, and the L2-Hys normalization method, which is the L2-norm followed by clipping (limiting the maximum values of unnormalized vector to 0.2) and renormalizing. Regarding the cell and block size, 6 to 8 pixel wide cells and $2 \times 2$ and $3 \times 3$ block sizes show the best performance for human detection. To evaluate the accuracy performance of our binary systems in human detection, we attempted to reproduce the presented work in [4], as will be described in Subsection 4.1.5. To do so, we used the above-mentioned parameter values.

We evaluated the effect of different parameters on the accuracy performance of our binary single-feature and dual-feature systems for human detection. The various configurations of parameters that were tested is presented below.

- **Window size (pixels):** $64 \times 64$, $64 \times 128$, $128 \times 128$, $128 \times 256$.
- **Cell size (pixels):** $4 \times 4$, $6 \times 6$, $8 \times 8$, $10 \times 10$.

- **Block size (cells):** $2 \times 2$, $3 \times 3$, $4 \times 4$.
- **Block overlap:** 0, 1/2, 3/4.
- **Kernel type:** Linear, Radial Basis Function (RBF).

We observed that changing the parameter values had a minor effect on accuracy performance. The RBF kernel showed slightly better accuracy for the INRIA dataset than the linear kernel. However, we used a linear kernel in this work because it not only showed good accuracy performance but also it is more area-efficient in hardware implementation than the RBF kernel. Therefore, we only focus on that from now on. Table 4.1 shows the values obtained empirically that give us the highest accuracy for human detection. It is worth noting that the HOG features are extracted from binary images in our proposed systems.

Table 4.1: Binary HOG descriptor parameter values for the human detection system

| | |
|---|---|
| **Window size** | $64 \times 128$ pixels |
| **Cell size** | $8 \times 8$ pixels |
| **Block size** | $2 \times 2$ cells |
| **Block overlap (Step stride)** | $8 \times 8$ pixels (1/2) |
| **Kernel type** | Linear kernel |

**LBP Feature Descriptor:** As described in Section 3.2, the binary HOG feature vector is combined with the binary LBP feature vector in the binary dual-feature system. We used a block size of $3 \times 3$ pixels for the LBP feature descriptor, which is the minimum possible block size for the LBP descriptor, as described in Subsection 2.2.2. We tested larger block sizes and 0, 1/3, 2/3 block overlap for the LBP feature descriptor and concluded that a block size of $3 \times 3$ pixels with no block overlap has the best accuracy performance. As mentioned in Section 3.2, we used the Uniform LBP and the size of the final LBP feature vector is 59, independent of image size, block size, and block overlap. It is worth noting that the LBP features are extracted from grayscale images in our proposed systems. A binarization step is applied to generate binary features from the extracted non-binary features, as described in Section 3.2.

**SVM Classifier:** As described in Chapter 3, our proposed binary single and dual systems can be used as binary and multi-class classifiers. The human detection system needs a binary classifier to classify samples into two classes: object-like (containing a human) and non-object (person-free).

### 4.1.3 Methodology for Accuracy Measurement in Human Detection

We tested our proposed binary single-feature and dual-feature systems for human detection first because the HOG descriptor outperforms other descriptors for human detection purposes [4]. This experiment was only performed using a software implementation because our proposed binary single-feature and dual-feature systems could not achieve satisfying accuracy performance for human detection, as will be described later in this section. We used the MATLAB function *fitcsvm* to train the binary SVM classifier. For the prediction phase, both the proposed binary single-feature and dual-feature systems were implemented in MATLAB using parameter values described in Subsection 4.1.2.

For training the SVM model, we used the training set of the INRIA person dataset. The training set contains 1,208 cropped person windows and their right reflection (2,416 images in total) as the positive set

and 1,218 person-free images as the negative set. First, a preliminary SVM model was trained using 2,416 positive samples and 3,000 randomly selected windows from person-free images (initial negative samples). Then, the 1,218 negative training samples were searched exhaustively for false positives using our preliminary model with different window sizes. The false positives that the preliminary model predicts are known as "hard examples". Hard examples were added to the initial negative training set to increase the variety of negative training samples resulting in accuracy improvement of the final model. There are around 250,000 hard examples in our training set. The SVM model was then re-trained using 2,416 positive samples and "initial negative samples + hard examples" to create the final model.

As discussed in Subsection 3.1.1, the size of the final HOG feature vector is $NumberOfBlocks \times BlockHistogramSize$, and the $BlockHistogramSize$ for the original and binary HOG descriptor has 36 and 16 features, respectively. The $NumberOfBlocks$ varies based on the image size, block overlap, and cell size. It will be described in detail in 5.1.2 that for an image of $n \times m$ pixels, a cell of $c \times c$ pixels and 1/2 block overlap, there are $m/c - 1$ blocks per row and $n/c - 1$ blocks per column. Therefore, there are $(n/c - 1) \times (m/c - 1)$ blocks per image, for an image of $n \times m$ pixels. With the parameter values presented in Table 4.1, there are $(64/8 - 1) \times (128/8 - 1) = 105$ blocks per image. Thus the size of the binary HOG feature vector for our binary single-feature system is $105 \times 16 = 1680$ features. It was mentioned earlier in this section that the length of the feature vector for the Uniform LBP is 59 features. As a result, the size of our binary dual-feature vector is $1680 + 59 = 1739$. The size of the SVM weights vector is the same as the feature vector. Hence, the SVM weights vector size for our binary single-feature and dual-feature systems is 1680 and 1739, respectively.

### 4.1.4 Overview of Accuracy Performance of Human Detection Systems

In this experiment, the performance of the proposed binary single-feature and dual-feature systems was evaluated for human detection. It means that the location of humans is specified by drawing a bounding box around them. To this end, the sliding window classification followed by Non-Max Suppression (NMS) [80], [81] is utilized.

In the sliding window classification, an image is scanned using different window sizes. In our experiment, we increased the window size with a scale of 1.2 and moved the windows with a step stride of 8 pixels (one cell). Each window is classified independently as being either an object-like or non-object window. A likelihood score is given to each window by the classifier model. The likelihood score denotes the probability of the object's presence in the window. A threshold, obtained in the training phase, is utilized to classify the window as an object-like or non-object window. If the likelihood score is greater than the threshold, the window is predicted to be object-like, otherwise it is predicted to be non-object. Using the sliding window classification, several windows around the object in an image are predicted as object-like, as shown in Figure 4.3(a). These windows should be combined to predict the bounding box that indicates the object location, as shown in Figure 4.3(b). Non-Max Suppression (NMS) is used to find the bounding box.

The performance of the object detectors should be quantified for comparison. We used the Detection Error Tradeoff (DET) [82] curve to quantify the object detector performance, plotting the false rejection rate against the false acceptance rate. False rejection, known as "false negative", means wrongly predicting a positive sample (object-like) as a negative (non-object). False acceptance, known as "false positive", means wrongly predicting a negative sample as a positive. A DET curve plots the miss rate versus False Positives

Per Window (FPPW) on a log-log scale. The miss rate is defined as Equation 4.1:

$$MissRate = \frac{\#FalseNegatives}{\#TruePositive + \#FalseNegatives} \tag{4.1}$$

where $TruePositive$ means correctly predicting a positive sample. Lower miss rate values mean better classification performance.

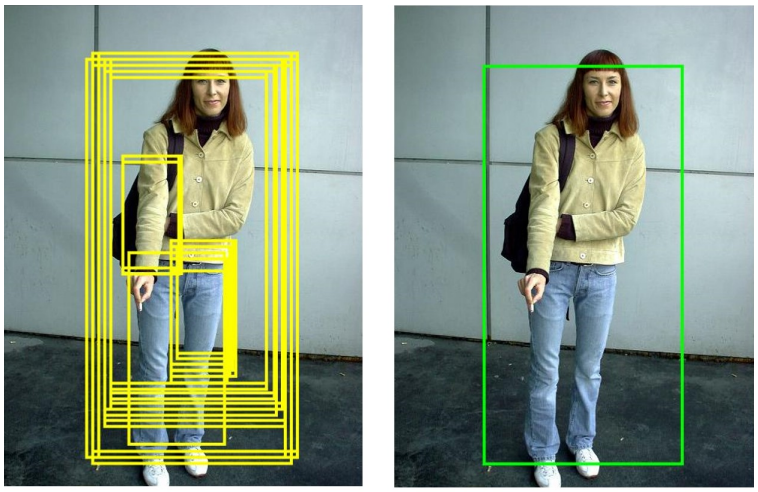



Figure 4.3: Bounding boxes: (a) Windows are predicted as object-like using the sliding window classification. (b) Final prediction is done using NMS

### 4.1.5 Summary of Human Detection System Performance

To evaluate the accuracy performance of our binary single-feature and dual-feature systems in human detection, we attempted to reproduce the results in [4] in MATLAB using the function "$hog\_feature\_vector$". However, we could not reproduce their reported results, therefore, we presented both of our reproduced curve and an estimated curve based on their reported results in Figure 4.4. Figure 4.4 compares the performance of our proposed binary single-feature and dual-feature systems and the presented work in [4]. All detectors were implemented in MATLAB according to the above-described settings. It can be seen that our binary dual-feature system outperforms the binary single-feature system. Nevertheless, the original work presented in [4] outperforms both of our proposed binary systems.

Our proposed binary model shows different accuracy performances for different samples in the INRIA person dataset, as will be discussed in detail in Chapter 6. Human detection is a challenging task because of variant appearance and a wide range of poses, clothing, background, and illumination. The two steps of binarization in our proposed binary model remove considerable detail during feature extraction and classification computation. This causes our proposed model to have high false positive and false negative rates for images where people are far from the camera and for samples with more complex backgrounds. Therefore,

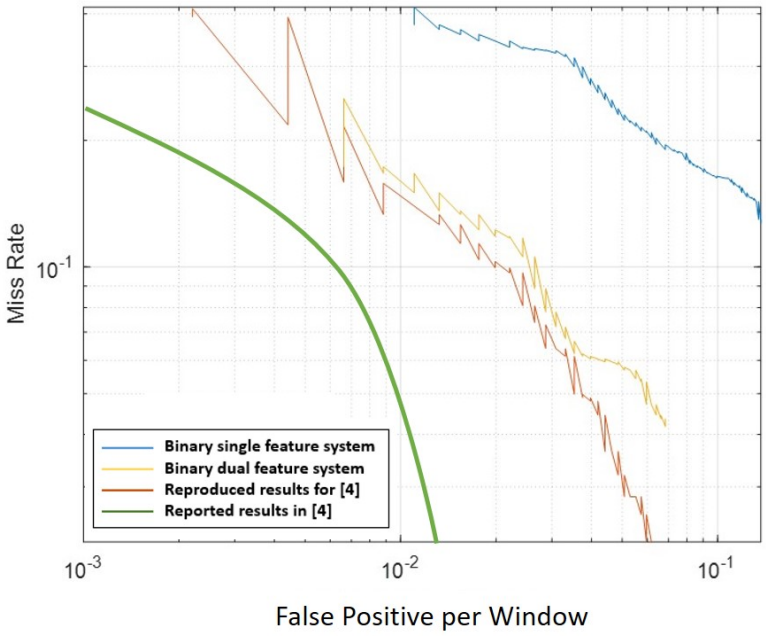



Figure 4.4: The performance of human detectors for the INRIA person dataset

our proposed model does not have a reliable performance for human detection.

As described at the beginning of this chapter, there is no perfect system that has reliable performance for all applications. Each system and application has particular specifications and requirements. As our proposed binary model fails for human detection, we examined the suitability of our model in solving the "multi-class classification" problem. We presented the methodology of our experiments and an overview of accuracy performance results for multi-class classification in Section 4.2.

## 4.2 Multi-class Classification System

In the second set of experiments to evaluate the accuracy performance of the proposed binary single-feature and dual-feature systems, we tested them as a multi-class classifier. To assess the multi-class classification accuracy performance, two commonly-used image classification datasets were used: the Caltech-256 dataset [83] and the KUL Belgium Traffic Sign dataset [84]. There are two reasons behind these selections. First, among literature reviews, the most similar work to ours, from an application point of view, is the work presented in [15]. Therefore, to have a fair classification accuracy comparison, we tested our proposed system using the same datasets and classes. Second, these datasets are regarded as challenging in image classification [15]. Therefore, achieving good accuracy performance in these two datasets can be evidence of a good performance on other datasets and a good indication of the accuracy in eventual deployment.

In this section, we first introduced the Caltech-256 and KUL Belgium Traffic Sign Classification datasets in Subsection 4.2.1. Subsection 4.2.2 describes setting for our experiments. Then, Subsection 4.2.3 provides the methodology of experiments on each dataset. Finally, an overview of accuracy performance will be

presented in Subsection 4.2.4.

### 4.2.1 Datasets

Caltech-256 is an object recognition dataset introduced by G. Griffin in 2007 [83]. It contains 30,607 real-world images in 256 object classes such as backpack, butterfly, goat, dog, airplane, and fireworks. Each class has at least 80 images.

The KUL Belgium Traffic Sign dataset contains more than 7,000 traffic signs from physically distinct traffic signs in Belgium. There are 62 different classes of traffic signs, such as stop, steep, side road, railroad, two-way traffic, and pedestrian crossing. Each image contains only one road sign that is not necessarily centered.

### 4.2.2 Setting

**HOG and LBP Feature Descriptors:** We investigated the accuracy performance of our binary single-feature and dual-feature systems for different parameter values of the HOG and LBP feature descriptors for multi-class classification. We repeated tests described in Subsection 4.1.2 for multi-class classification system. It is observed that changing parameter values has a minor effect on accuracy performance. Table 4.2 shows the binary HOG descriptor parameter values obtained empirically that give us the highest accuracy for multi-class classification. Similar to the object detection system, LBP features are extracted from blocks of $3 \times 3$ pixels with no block overlap. The size of the binary LBP feature vector is 59, independent of image size, block size, and block overlap. It is worth noting that the HOG and LBP features are extracted from binary and grayscale images, respectively. A binarization step is applied to generate binary features from extracted non-binary features, as described in Chapter 3.

**SVM Classifier:** As described in Chapter 3, our proposed binary single-feature and dual-feature systems can be used as one-class (binary) and multi-class classifiers. For the multi-class classification system, a multi-class SVM classifier is used that will be described in the following subsection.

Table 4.2: Binary HOG descriptor parameter values for the multi-class classifier

| | |
|---|---|
| **Input image** | $64 \times 64$ pixels |
| **Cell size** | $8 \times 8$ pixels |
| **Block size** | $2 \times 2$ cells |
| **Block overlap (Step stride)** | $8 \times 8$ pixels (1/2) |
| **Kernel type** | Linear kernel |

### 4.2.3 Methodology for Accuracy Measurement in Multi-class Classification

A One-Vs-One (OVO) multi-class SVM classifier with a linear kernel is implemented as the classifier in our binary single-feature and dual-feature multi-class classification systems. Reference [15] implemented a five-class classifier. Therefore, to have a fair classification accuracy comparison, we chose the same number of classes and the same datasets. As discussed in Subsection 2.3.1, the OVO multi-class SVM classifier consists

of $n(n-1)/2$ binary classifiers for $n$ classes, with one binary classifier for every possible pair of classes. Thereby, there needs to be ten independent binary classifiers for five classes.

To evaluate the accuracy performance of our binary multi-class classification systems, we tested both a software implementation and a hardware implementation. For both software and hardware implementations, the five-class SVM model is generated offline using the function *"fitcecoc"* in MATLAB. Then, the weights and bias values for the ten independent binary classifiers, and a coding design matrix were extracted from the model to implement the prediction phase. A sample of the SVM weights and bias values are provided in the appendix. A coding design matrix is a matrix that shows how a multi-class problem is split into a series of binary classification problems. Figure 4.5 shows the coding design matrix extracted from MATLAB for the five-class SVM classifier. Each column of the coding design matrix corresponds to a binary classifier, and each row corresponds to a distinct class. The first column corresponds to the binary classifier 1 and assigns the instance into one of two classes 1 or 2. If the score of the binary classifier 1 is greater than zero, it is assigned to class 1 (shown as 1 in Figure 4.5); otherwise, it is assigned to class 2 (shown as -1 in Figure 4.5). Similarly, the second column corresponds to the binary classifier 2 and assigns the instance into one of two classes 1 or 3. According to the coding design matrix, the binary classification process continues for the rest of the binary classifiers. Details of the multi-class classification computation will be discussed in Chapter 5.

| Binary Classifiers ➡ | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | | | | | | | Class 1 |
| | -1 | | | | 1 | 1 | 1 | | | | Class 2 |
| | | -1 | | | -1 | | | 1 | 1 | | Class 3 |
| | | | -1 | | | -1 | | -1 | | 1 | Class 4 |
| | | | | -1 | | | -1 | | -1 | -1 | Class 5 |

Figure 4.5: The coding design matrix for the five-class SVM classifier. In each column, if the score of the binary classifier is greater than zero, the sample is assigned to the class shown as 1; otherwise, it is assigned to the class shown as -1.

For the software implementation of the prediction phase, the binary HOG feature descriptor, as described in Subsection 3.1.1, the Uniform LBP feature descriptor, as described in Section 3.2, and the SVM classifier, as described in Subsection 3.1.2, were implemented in MATLAB. For the hardware implementation of the prediction phase, both the binary single-feature and dual-feature systems, described in Chapter 5, were simulated using VHDL code in Vivado [85]. The methodology of the experiments, discussed in the following paragraphs, were the same for software and hardware implementations. The HOG features are extracted from binary images and the LBP features are extracted from grayscale images.

As discussed in Subsection 4.1.3, the size of the binary HOG feature vector is $(n/c-1)\times(m/c-1)\times 16$, for an image of $n\times m$ and cell of $c\times c$ binary pixels. With the parameter values provided in Table 4.2, the size of the binary HOG feature descriptor in each binary classifier of our binary single-feature multi-class classification system is $(64/8-1)\times(64/8-1)\times 16 = 784$. Considering 59 features for the LBP feature vector, the size of the feature vector for each binary classifier of the dual-feature multi-class classification system is $784+59=843$. On the one hand, according to the Vapnik–Chervonenkis (VC) dimension [86], the minimum number of required training samples is the dimension of the feature vector plus one for a linear

SVM. Therefore, we need to have more than 844 training samples. On the other hand, we are limited to the number of available samples in each category of the Caltech-256 and KUL Belgium Traffic Sign datasets. Considering the above-mentioned conditions and limitations, our training set contains 1000 images, with 200 images per class. The number of testing samples was chosen based on [15] to have a fair comparison. Thus, in the testing phase of each experiment, each subset consists of 100 images, with 20 images from each of the five classes. In both the training and testing phase, color images are first converted to grayscale and then binary images.

We performed three experiments to evaluate the accuracy of our proposed binary single-feature and dual-feature systems for multi-class classification: one experiment on the Caltech-256 dataset and two experiments on the KUL Belgium Traffic Sign dataset. In each experiment, the performance of the proposed systems is evaluated in both software and hardware implementation.

***Experiment 1: Caltech-256***

For Experiment 1, ten randomly selected subsets from the Caltech-256 dataset are used. Each subset has five classes: airplane, face, horse, motorbike, and watch. To have a fair comparison, the chosen classes are the same as subset 1 of Experiment 1 in [15]. Some examples from the subsets are shown in Figure 4.6.


Figure 4.6: Example images from the Caltech-256 dataset [83]

***Experiment 2: KUL Belgium Traffic Sign - KUL-Set1***

In Experiment 2, ten randomly selected subsets from the KUL Belgium Traffic Sign dataset are used. Each subset has five classes: crossing for cyclists, danger, one way traffic, parking lot, and dead end. The chosen classes are the same as Experiment 2 in [15] for a fair comparison. Some examples from the subsets are shown in Figure 4.7.

***Experiment 3: KUL Belgium Traffic Sign - KUL-Set2***

Figure 4.8 shows the binary format of Figure 4.7. As can be seen, the binary format of "Crossing for Cyclists" and "Danger" classes have a similar pattern. Likewise, the "One Way Traffic" and "Dead End" have a similar pattern, especially in the binary format. It will be discussed in Chapter 6 that this similarity in the binary format of classes results in higher misclassification between them. Therefore, we performed Experiment 3 using another set of the KUL Belgium Traffic Sign dataset. To create KUL-Set2, we


Figure 4.7: Example images from the KUL Belgium Traffic Sign dataset [84] in RGB format - KUL-Set1

replaced "Crossing for Cyclists" and "Dead End" classes in KUL-Set1 with "Give Way" and "Speed Limit", respectively. Figure 4.9 and 4.10 illustrate some examples of KUL-Set2 in the RGB and binary format, respectively. It can be seen that the binary format of the five classes in KUL-Set2 has more distinctive patterns compared to KUL-Set1. We performed Experiment 3 using KUL-Set2 with the above-described replacements in KUL-Set1 to analyze and evaluate our binary model performance.

In short, for Experiment 3, each subset contains randomly selected samples from five classes, including give way, danger, one way traffic, parking lot, and speed limit. Similar to Experiments 1 and 2, there are ten subsets in the testing phase.

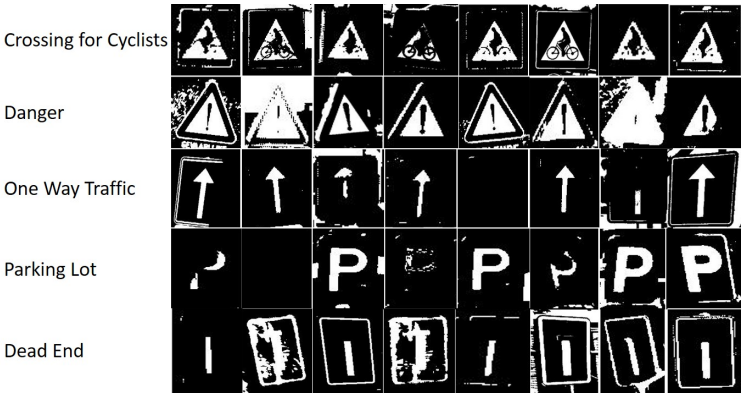



Figure 4.8: Example images from the KUL Belgium Traffic Sign dataset in binary format - KUL-Set1


Figure 4.9: Example images from the KUL Belgium Traffic Sign dataset [84] in RGB format - KUL-Set2

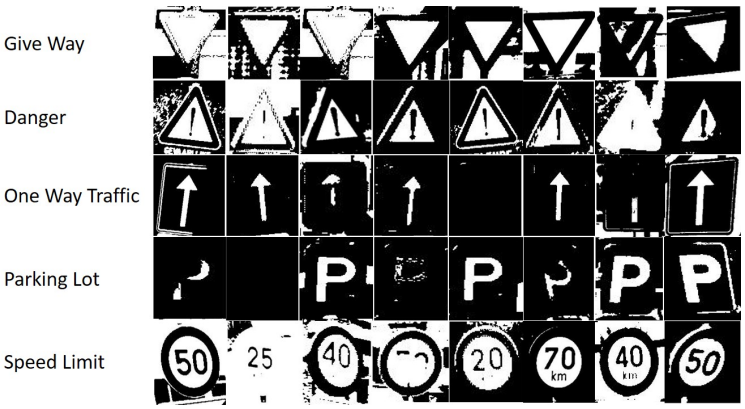



Figure 4.10: Example images from the KUL Belgium Traffic Sign dataset in binary format - KUL-Set2

### 4.2.4 Overview of Accuracy Performance of Multi-class Classification Systems

To quantify the performance of multi-class classifiers, we used the confusion matrix. The confusion matrix is a commonly-used method to describe the classification performance of a model. In a confusion matrix, the diagonal elements show the number of images that have been classified correctly, whereas other elements show misclassified images. As mentioned earlier in this section, there are ten subsets in the testing phase of each experiment. As the standard deviation is low, the average of the confusion matrices for the ten subsets is calculated to evaluate classification performance. An overview of the results for each experiment is presented in this subsection, and details of the classification performance evaluation will be discussed in Chapter 6.

Table 4.3 compares the classification rate of the presented work in [15] with our proposed binary single-feature and dual-feature classification systems in both hardware and software implementations for the Caltech-256 dataset. The classification accuracy dropped in the hardware implementation compared to the software implementation for all three systems. It is expected because hardware implementation uses fixed-point arithmetic with less precision than the floating-point arithmetic used in software implementation. Our single-feature classification system could achieve 3.1% (87.1% - 84%) and 4.4% (89.4% - 85%) higher classification rate compared to [15] in hardware and software implementation, respectively. However, it can be seen that combining LBP features with HOG features increases the classification accuracy further, by 3.5% (90.6% - 87.1%) in hardware and 2% (91.4% - 89.4%) in software , compared to the single-feature system.

Table 4.3: Multi-class classification rate comparison on the Caltech-256 dataset

| | **Single-feature** | **Dual-feature** | **[15]** |
|---|---|---|---|
| **Average classification rate in hardware implementation** | 87.1% | 90.6% | 84% |
| **Average classification rate in software implementation** | 89.4% | 91.4% | 85% |

Table 4.4 compares the classification rate of Experiments 2 and 3 for the KUL Belgium Traffic Sign dataset. Among reported classification rates for our proposed systems in Table 4.4, the lowest classification rate is related to our single-feature system for KUL-Set1. However, it is still 1.2% (79.2% - 78%) higher than [15] for KUL-Set1.

Table 4.4: Multi-class classification rate comparison on the KUL Belgium Traffic Sign dataset

| | **Single feature KUL-Set1** | **Dual feature KUL-Set1** | **Single feature KUL-Set2** | **Dual feature KUL-Set2** | **[15] KUL-Set1** |
|---|---|---|---|---|---|
| **Average classification rate in hardware implementation** | 79.6% | 84.8% | 88.3% | 91.8% | 78% |
| **Average classification rate in software implementation** | 84% | 88.5% | 90.1% | 94% | 80% |

As discussed earlier in this section, the binary format of “Crossing for Cyclists” and “Danger” classes,

and “One Way Traffic” and “Dead End” classes in KUL-Set1 are very similar. It will be shown in Chapter 6 that the misclassification of our binary single-feature and dual-feature systems for these classes is higher than other classes.

The high misclassification between some classes leads to a decrease in the overall classification rate for KUL-Set1. It can be seen in Table 4.4 that the hardware implementation of our binary single-feature multi-class classifier obtained a 8.7% (88.3% - 79.6%) higher classification rate for KUL-Set2 compared to KUL-Set1. This happens because of replacing “Crossing for Cyclists” and “Dead End” classes in KUL-Set1 with “Give Way” and “Speed Limit” classes in KUL-Set2, respectively. It will be shown in Chapter 6 that this replacement decreases the misclassification between classes of KUL-Set2 compared to KUL-Set1. Therefore, the overall classification rate of the system increases for KUL-Set2 compared to KUL-Set1.

Moreover, combining LBP features with HOG features increases the classification rate in both software and hardware implementations for both KUL-Set1 and KUL-Set2. The highest classification rate in Table 4.4 is related to our dual-feature system for KUL-Set2. It is 13.8% (91.8% - 78%) higher than [15] in hardware implementation. Our dual-feature system could achieve an 6.8% (84.8% - 78%) higher classification rate than [15] for KUL-Set1.

In summary, our proposed classification systems show reasonable accuracy compared to other works. We have then proceeded to hardware implementation, which details are given in the next chapter.

# Chapter 5

# Hardware Implementation

We proposed two image classification systems based on the binary single-feature set and the binary dual-feature set. The binary single-feature system is founded on a binary HOG feature set and an SVM classifier. In the binary dual-feature system, the HOG features are first combined with the LBP features, and then the same classification process as the binary single-feature system is applied to the combined features.

It is described in Chapter 4 that we evaluated our proposed binary single-feature and dual-feature systems for solving the "object detection" and "multi-class classification" problems. It is also shown that the software implementation of our proposed systems could not satisfy the required accuracy performance for human detection. However, our proposed binary single-feature and dual-feature systems could outperform the similar existing solution for multi-class classification. Therefore, we did not implement the binary classification system in hardware.

This chapter introduces hardware implementations for our proposed binary single-feature and dual-feature multi-class classification systems. Both systems consist of three main phases: preprocessing, feature extraction, and classification. We first describe the hardware implementation of the single-feature system in Section 5.1. Section 5.2 explains the modules that have been added to the single-feature system to build the dual-feature system. Section 5.3 summarizes our main contributions to the hardware implementation of the single-feature system and the dual-feature system.

## 5.1 Binary Single-feature Classification System

Figure 5.1 shows an algorithmic level block diagram of the binary single-feature system. It consists of three main phases: preprocessing, binary HOG descriptor, and SVM classification. The system input is a stream of grayscale pixels for an image, and the output is the class label. Although all steps show the input and output of each block sequentially, the system is implemented in a way so that some modules are working in a parallel or semi-parallel manner to speed up the process. This section describes details of the hardware implementation of each module and their parallel execution.

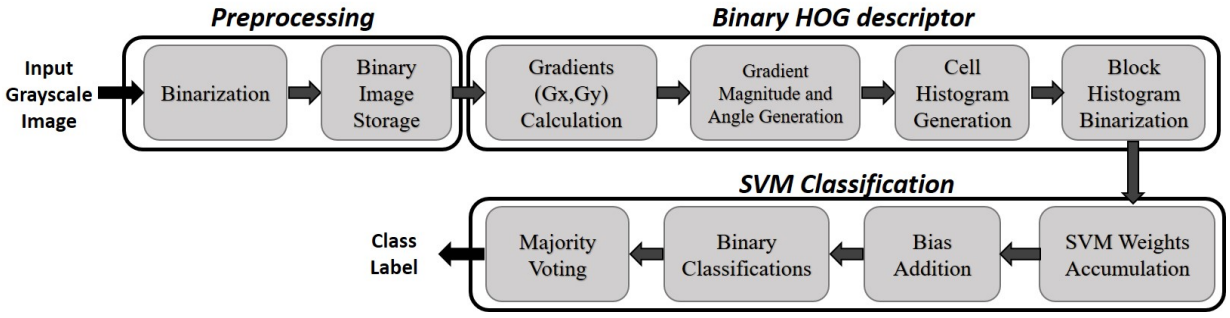



Figure 5.1: Algorithmic level block diagram of the binary single-feature classification system

### 5.1.1 Binary Single-feature System Preprocessing Phase

The preprocessing phase starts by performing the binarization process. The input to this module is a stream of pixels from the input grayscale image. Each pixel of the grayscale image is compared with a predefined threshold to generate the binary pixel. The threshold has been obtained empirically by averaging the grayscale pixel values for randomly selected images from the dataset. The threshold is defined as a constant in our hardware implementation, but can be set at run time. Our experiments show that customizing the threshold can improve accuracy up to 1%. If the grayscale pixel value is greater than the threshold, the binary pixel is set to 1; otherwise, it is set to 0.

Figure 5.2 shows the data flow of the preprocessing phase, including binarization and binary image storage, and the first two steps of the binary HOG descriptor: the gradient calculations and the magnitude and angle generation. As this subsection focuses on the preprocessing phase, we only describe details of the binarization process and binary image storage in Figure 5.2. Details of the gradient calculations and the magnitude and angle generation processes will be discussed in Subsection 5.1.2.

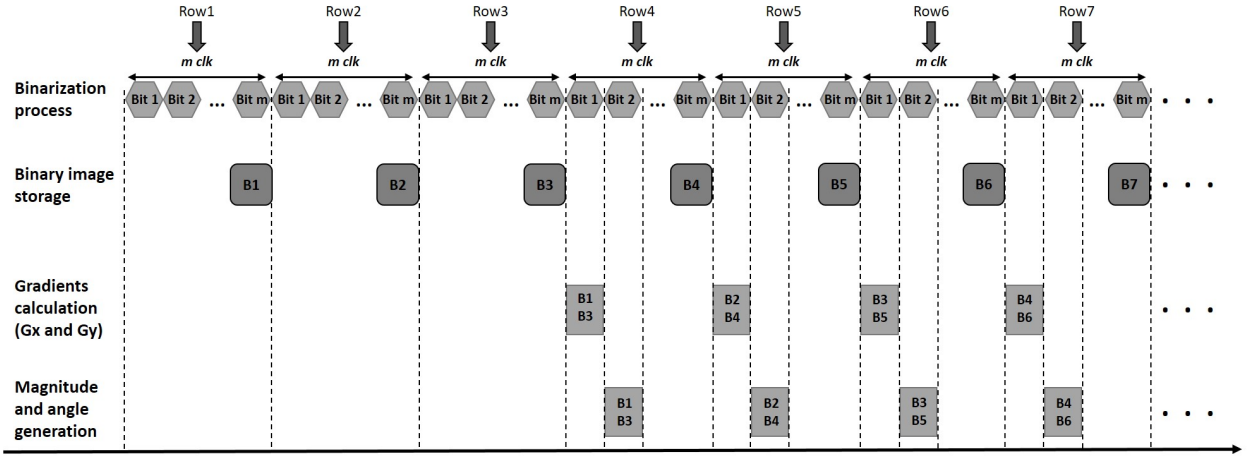



Figure 5.2: Data flow of the first four steps: input image binarization, binary image storage, gradients calculation, and magnitude and angle generation. The Bi blocks in the "Binary image storage" line show termination of the binarization process for row $i$. Blocks Bi and Bj in "Gradients calculation" and "Magnitude and angle generation" lines show the processing times on row $i$ and $j$ of the binary image.

The binarization process performs pixel-by-pixel using one comparator and generates one bit of the binary image at each clock cycle. Therefore, for an $n \times m$ image, the binarization process takes $m$ clock cycles for each row of the input grayscale image. This is shown in the "Binarization process" line of Figure 5.2.

As mentioned in Subsection 3.1.1, a [-1;0;1] gradient filter is utilized in this implementation for the gradients calculation. For each pixel of an $n \times m$ image, gradients on the $x$ and $y$ axis ($G_x$ and $G_y$) are calculated using the below formulas.

$$G_x(r, c) = I(r, c + 1) - I(r, c - 1) \tag{5.1}$$

where $r$, $c$ refer to rows and columns, respectively. In Equation 5.1, $r$ = 1,...,$n$ and $c$ = 2,...,$m - 1$.

$$G_y(r, c) = I(r + 1, c) - I(r - 1, c) \tag{5.2}$$

where in Equation 5.2, $r$ = 2,...,$n - 1$ and $c$ = 1,...,$m$.

Note that $c$ starts from 2 for $G_x$ and ends at $m - 1$. The first and last columns of $G_x$ are the same in the input image. Similarly, $r$ starts from 2 for $G_y$ and ends at $n - 1$. The first and last rows of $G_y$ are the same in the input image.

To process row $r$ of $G_y$, row $r - 1$ and $r + 1$ of the binary image are needed. Therefore, we need to finish the binarization process for the first three rows of the input image and then start the gradients calculation process. In the "Binary image storage" line of Figure 5.2, the Bi blocks appear at the termination of the binarization process for row $i$. As shown, gradient calculations for B1 and B3, which contain row 1 and row 3 of the binary image, starts after the binarization process ends for the third row (B3). Similarly, gradient calculations for B2 and B4, which contain row 2 and row 4 of the binary image, starts after the binarization process ends for the fourth row (B4), and so on.

According to Equation 5.2, each row of the binary image is used to calculate two different rows of $G_y$ and is no longer needed in subsequent computations. Thereby, the binary image only needs to be stored as long as it is required for gradient calculations. As a result, our hardware implementation needs a limited number of buffers to store the binary image rows temporarily. With regard to the above-described sequence of binarization and gradients calculation processes, and to facilitate the subsequent pipelined processes, we used six $m$-bit buffers in our hardware implementation to store the binary image rows temporarily, shown as R1, ..., R6 in subsequent subsections and figures. After filling R6 with row 6 of the binary image, R1 is filled with row 7, R2 is filled with row 8, and so on. The preprocessing phase, consists of binarization and binary image storage, is continued, and the gradients calculation and magnitude and angle generation are being executed in a pipelined manner with preprocessing, as shown in Figure 5.2.

### 5.1.2 Binary HOG Descriptor

The previous subsection describes the image binarization process and storage scheme of the binary image rows in six $m$-bit buffers. There are six possible pairs of $m$-bit buffers for the gradient calculations considering the storage scheme of the binary image. The possible pairs are (R1, R3), (R2, R4), (R3, R5), (R4, R6), (R5, R1), and (R6, R2).

**Gradients Calculation:** Figure 5.3 shows the implementation of gradient calculations and data flow between binary image storage and the gradient calculations of the binary HOG descriptor. As shown in the

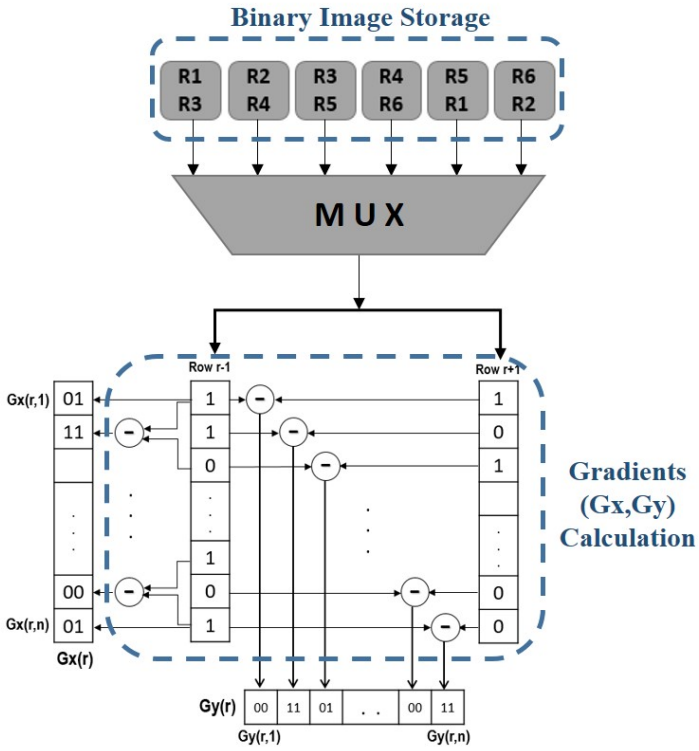



Figure 5.3: The gradients calculation implementation

figure, one multiplexer loads two $m$-bit registers with row $r+1$ and $r-1$ from one pair of the buffers. To perform Equation 5.2, $m$ subtractors are implemented to generate one row of the $G_y$ matrix at one clock cycle.

According to Equation 5.1, to calculate $G_x(r,c)$, bits $c+1$ and $c-1$ of row $r$ are needed. As shown in Figure 5.3, one row of $G_x$ is calculated using $m-2$ subtractors at one clock cycle. The first and last values of $G_x$ are sign-extended values of the binary pixels. Since the gradients are calculated from a binary pixel, the possible values for $G_x$ and $G_y$ are 0, 1, and -1, designated as 00, 01, and 11 in the $G_x$ and $G_y$ register arrays, as shown in Figure 5.3.

Figure 5.4 illustrates the data format after binarization and for each step of the binary HOG descriptor. For an image of $n \times m$ binary pixels, two separate $n \times m$ 2-bit arrays store the $G_x$ and $G_y$ matrices because $G_x$ and $G_y$ values are represented by two bits. The rest of this figure will be described later in this subsection, after describing each step of the binary HOG descriptor.

**Gradients Magnitude and Angle Generation:** As shown in Figure 5.1, the gradient calculations are followed by gradient magnitude and angle generation. We have discussed in Chapter 3 that the gradient magnitude and angle generation are implemented utilizing two lookup tables (LUTs) as shown in Table 5.1. As discussed in Subsection 3.1.1, for a binary image, gradient angles have four possible values: 0, +45, -45, and 90. These values are designated by 00, 01, 10, and 11. The possible gradient magnitudes are 0, 1, and $\sqrt{2}$, which we chose to encode as 00, 01, and 10, respectively. The gradient magnitude and angle calculation for one row of $G_x$ and $G_y$ are performed in one clock cycle. The $G_x$ and $G_y$ values for each matrix entry are compared with LUT values, and the gradient magnitude and angle are generated. As shown in Figure

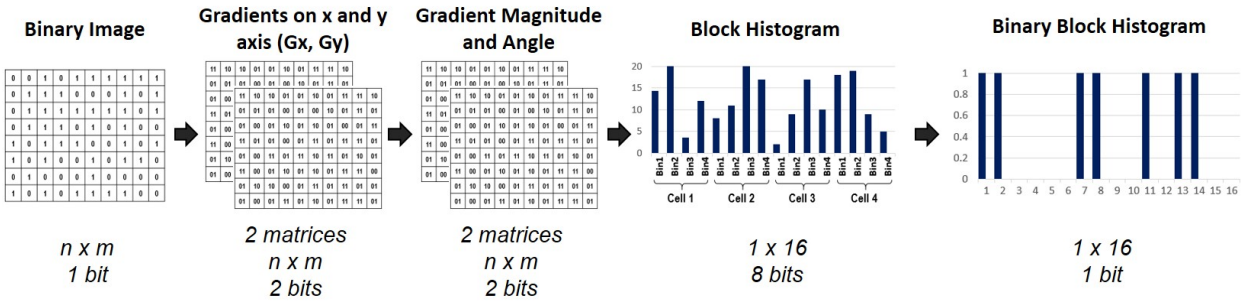



Figure 5.4: Format of data after binarization and each step of the binary HOG descriptor

5.4, two separate $n \times m$ 2-bit arrays are implemented to store the gradient magnitude and angle matrices because the input image size is $n \times m$ pixels, and the gradient magnitude and angle values are represented by two bits.

Table 5.1: Gradient magnitude and angle lookup table (same as Table 3.1, reproduced here for convenience)

| $G_x$ | $G_y$ | **Magnitude** | **Angle** |
|---|---|---|---|
| 00 | 00 | 00 | 00 |
| 00 | 01 | 01 | 10 |
| 00 | 11 | 01 | 10 |
| 01 | 00 | 01 | 00 |
| 01 | 01 | 10 | 01 |
| 01 | 11 | 10 | 11 |
| 11 | 00 | 01 | 00 |
| 11 | 01 | 10 | 11 |
| 11 | 11 | 10 | 01 |

So far, we have discussed how the preprocessing steps, including binarization and binary image storage, and the first two steps of the binary HOG descriptor are being executed in a pipelined manner. Before describing the implementation of the rest of the binary HOG descriptor, we formulized the processing time for the part of the system we have discussed so far. As Figure 5.2 shows, the binarization process for one $m$-pixel row takes $m$ clock cycles. Therefore, it takes $n \times m$ clock cycles to complete the binarization process for an image of $n \times m$ grayscale pixels. As can be seen in Figure 5.2, the gradient calculations process starts after completing the binarization of the third row and takes only one clock cycle. Gradient magnitude and angle generation for each row of the $G_x$ and $G_y$ matrices is performed right after gradient calculations and takes only one clock cycle. As mentioned before, the last row of the gradients is the same as the input image, so it is copied directly from the binary image, and there is no need to calculate. Therefore, after the $n \times m$ clock cycles to complete binarization, it takes three more clock cycles to process the gradients calculation, and gradient magnitude and angle generation for the entire image.

In our hardware implementation, the binarization process, gradient calculations, and gradient magnitude and angle generation are executed in a pipelined manner, taking $n \times m + 3$ clock cycles for an image of $n \times m$ grayscale pixels. Given a binary image, there is no need for binarization, which is the preprocessing step. To obtain the processing time of the gradient calculations and gradient magnitude and angle generation, we

assume that the input of the system is a stream of binary pixels. There needs one clock cycle to load two rows of the binary image into the designated registers. According to the above discussion, one clock cycle is required to generate each row of $G_x$ and $G_y$. The gradient magnitude and angle generation for each row of $G_x$ and $G_y$ also take one clock cycle. Therefore, the gradient magnitude and angle generation for each row of the binary image need three clock cycles. For an $n$-row binary image, $3n$ clock cycles are required to the generate gradient magnitude and angle.

**Cell Histogram Generation:** The control flow of the rest of the binary HOG descriptor is shown in Figure 5.5. As illustrated in the algorithmic level block diagram in Figure 5.1, the feature extraction process is continued with cell histogram generation after gradient magnitude and angle generation. As shown in Figure 5.5, cell histogram generation starts by selecting the corresponding gradient magnitude and angle to the first cell. As the cell size is $c \times c$ binary pixels, $c$ rows of $c$ values should be selected. This is done by repeating two states: "Cell Selection" and "Cell Rows Counting". In each iteration of this two-state loop, the $c$ values corresponding to one row of the cell are selected. Therefore, $c$ repetitions are required. This process takes $2c - 1$ clock cycles for a cell of $c \times c$ bits.

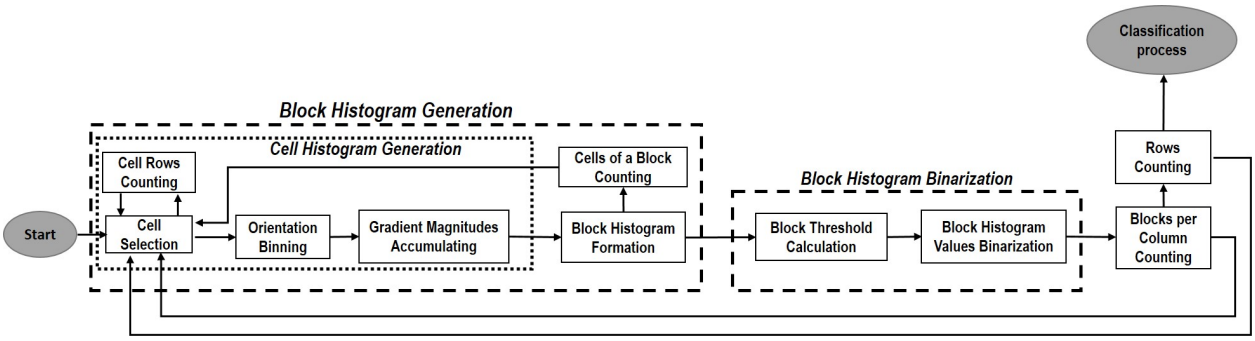



Figure 5.5: Control flow of the binary HOG descriptor

The number of histogram bars in the cell depends on the number of bins. As described in Subsection 3.1.1, the number of bins in the binary HOG descriptor is four. After that, orientation binning is performed by classifying gradient magnitudes into four groups based on their angle values, as shown in Table 5.2. This is shown as "Orientation Binning" in Figure 5.5. Following that, assigned gradient magnitudes to each bin are accumulated using the circuit shown in Figure 5.6, and illustrated as "Gradient Magnitudes Accumulating" in Figure 5.5. This module consists of $c \times c - 1$ adders for a cell of $c \times c$ binary pixels. Cell histograms are available after one clock cycle by utilizing four replicates of this circuit (four bins per histogram).

Table 5.2: Orientation binning lookup table

| Magnitude | Angle | Bin |
|---|---|---|
| xx | 00 | 1 |
| xx | 01 | 2 |
| xx | 10 | 3 |
| xx | 11 | 4 |

**Block Histogram Generation:** Since the concatenation of cell histograms forms the block histogram, the cell histogram calculation is repeated for all four cells of the current block. The process of concatenating

the cell histograms is performed using two states, "Block Histogram Formation" and "Cells of a Block Counting", see Figure 5.5. For four cells of a block, considering $2c-1$ clock cycles for cell selection and four clock cycles for the rest of the block histogram generation process shown in Figure 5.5, the block histogram generation takes $4 \times ((2c-1)+4)$ clock cycles for a cell of $c \times c$ bits. The block histogram data format can be found in Figure 5.4 as a vector of $1 \times 16$ 8-bit values, consisting of four cell histograms.

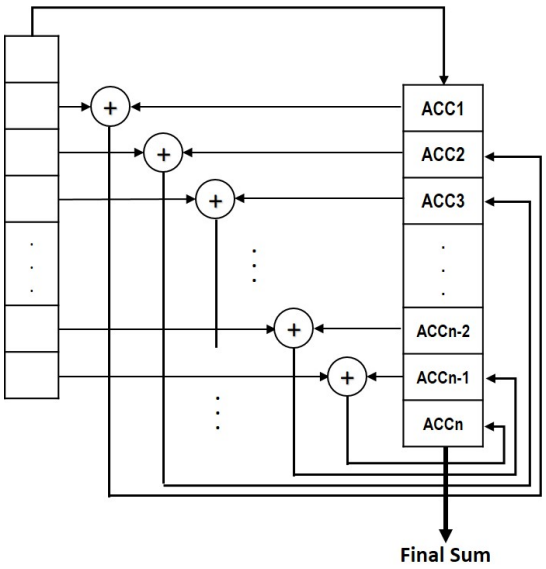



Figure 5.6: Accumulator architecture with $n-1$ adders to accumulate $n$ values

**Block Histogram Binarization:** After completing the block histogram generation, block histogram binarization should be performed. The mean of the block histogram is used as the threshold for the block histogram binarization. An accumulator, similar to the circuit in Figure 5.6 with 15 adders, is utilized to accumulate 16 values of the block histogram. To calculate the threshold, the sum of the block histogram is shifted to the right by four bits to perform a division by 16. The "Block Threshold Calculation" in Figure 5.5 illustrates the calculation of the mean of the block histogram.

The block histogram values are compared with the threshold to create the binary block histogram. We employed 16 comparators to complete this process for 16 block histogram values in one clock cycle, shown as "Block Histogram Values Binarization" in Figure 5.5. If the block histogram value is greater than the threshold, the binary feature is set to 1; otherwise, it is set to 0. Figure 5.4 shows the block histogram data format after binarization and the data that is stored in a 16-bit register in our hardware implementation. At this point, the feature extraction process for one block is completed.

It is described earlier in this section that "Cell Histogram Generation" takes $(2c-1)+2$ clock cycles. Two more clock cycles are required to form the block histogram and keep the count of the cells in a block, resulting in $(2c-1)+4$ clock cycles. To generate the block histogram, cell histogram generation should be repeated four times for four block cells, so it needs $4 \times ((2c-1)+4)$ clock cycles. As shown in Figure 5.5 and described above, "Block Histogram Binarization" needs two clock cycles. Therefore, our hardware implementation needs $4 \times ((2c-1)+4)+2$ clock cycles to generate the binary block histogram.

In our system, the input image is scanned in a raster manner, with a step stride (block overlap) of one

cell, as shown in Figure 5.7. There are two counters in our hardware implementation to control the feature extraction process for the entire image. One counter counts the blocks per column, and the other counts the rows, to perform the raster scan shown in Figure 5.7. The raster scan is controlled using two states shown as "Blocks per Column Counting" and "Rows Counting" in Figure 5.5. After completing feature extraction for one block, one clock cycle is required to count the block ("Blocks per Column Counting" in Figure 5.5) and then return to "Cell Selection" to repeat the feature extraction process for the next block. Therefore, one clock cycle is added to the required clock cycles to generate the binary block feature, resulting in $4 \times ((2c-1)+4)+3$ clock cycles.

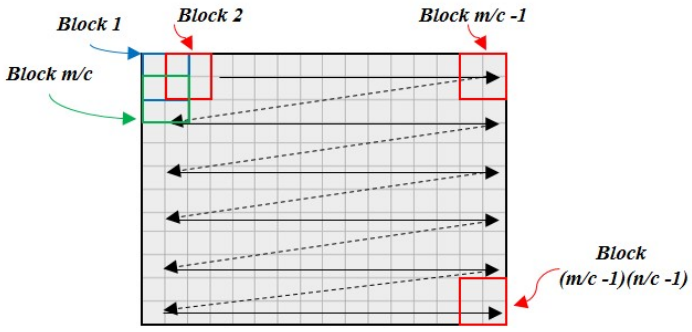



Figure 5.7: Input image blocks scan scheme for the binary HOG descriptor for an image of $n \times m$ pixels, cell of $c \times c$ pixels, and block of $2 \times 2$ cells

It is described earlier in this subsection that the three modules of binarization, gradient calculations, and gradient magnitude and angle generation, executing in a pipelined manner, need $n \times m+3$ clock cycles in our hardware implementation. It is also discussed that it takes $4\times((2c-1)+4)+3$ clock cycles to generate a binary block histogram for one block. For an image of $n\times m$ and a cell of $c\times c$ binary pixels, with a block overlap of one cell, there are $m/c-1$ blocks per row and $n/c-1$ blocks per column. Therefore, there are $(n/c-1)\times(m/c-1)$ blocks per image. As a result, the binarization process and the binary HOG feature extraction process for an image of $n \times m$ grayscale pixels take $n \times m+3+((n/c-1) \times (m/c-1)) \times (4 \times ((2c-1)+4)+3)$ clock cycles in our hardware implementation.

It is discussed earlier in this subsection that under the assumption of having a stream of binary pixels as the input and hence no need for the binarization process, gradient calculations and gradient magnitude and angle generation need $3n$ clock cycles for the entire image of $n \times m$ binary pixels. Therefore, the processing time of the binary HOG feature descriptor in our hardware implementation is $3n+((n/c-1) \times (m/c-1)) \times (4 \times ((2c-1)+4)+3)$ clock cycles.

After completing the feature extraction process for the entire image, the final feature vector, which is a concatenation of the block histograms, is ready. At this point, the control flow transits to the classification process.

### 5.1.3 Binary Single-feature System Classification Phase

A One-Vs-One (OVO) multi-class SVM classifier with a linear kernel is implemented as the classifier in our system. The number of object classes in our system was chosen to be five. Among literature that was reviewed, the most similar work to ours from the application point of view is the presented work in

[15]. As discussed in Chapter 4, M. Qasaimeh et al. in [15] did two experiments using five classes of two commonly-used image classification datasets: Caltech-256 dataset [83] and KUL Belgium Traffic Sign dataset [84]. Therefore, to have a fair classification accuracy evaluation, we chose the same number of classes and same datasets. Details of the accuracy evaluation will be discussed in Chapter 6.

As discussed in Section 2.3, the OVO multi-class SVM classifier consists of $n(n-1)/2$ binary classifiers for $n$ classes, one binary classifier for every possible pair of classes. Thereby, there needs to be ten independent binary classifiers for five classes. In our system, the five-class SVM model is generated offline using MATLAB, then the weights and bias values for ten independent binary classifiers and a coding design matrix are extracted from the model to implement the prediction phase in hardware. A sample of the SVM weights and bias values are provided in the appendix. A coding design matrix is a matrix that shows how a multi-class problem is split into a series of binary classification problems. This section describes our hardware implementation of the SVM classifier.

Figure 5.8 shows the general scheme of the five-class SVM classifier implementation. Ten independent binary classifiers run in parallel, and each classifier assigns a sample between two classes. The sample is then classified based on a majority voting scheme. In this subsection, the implementation of the binary classifiers is first described, and then details of the decision-making scheme based on the coding matrix is presented. Finally, the multi-class classification process based on the majority voting is explained.

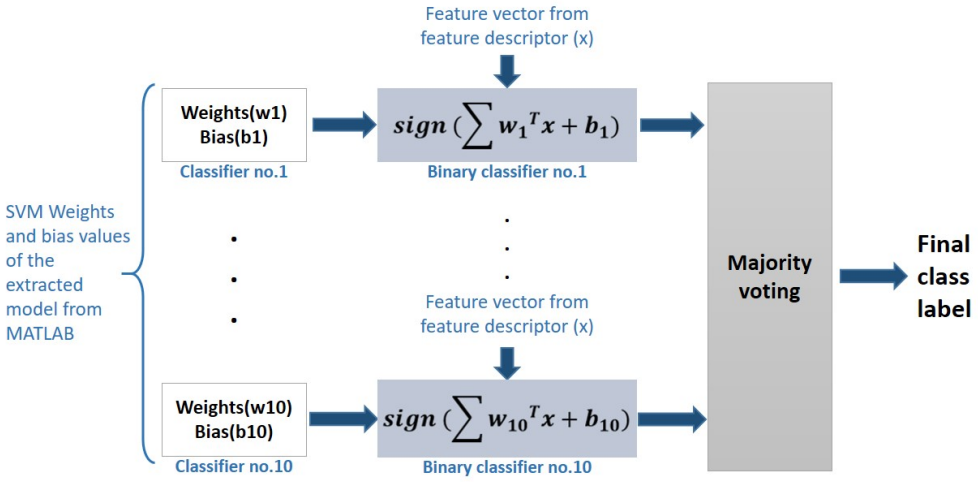



Figure 5.8: General scheme of the five-class SVM classifier

As described in Subsection 3.1.2, all multiplications of the binary SVM classifier are replaced by additions due to the binary features, which is shown as "SVM Weights Accumulation" in the first step of the SVM classification process in Figure 5.1. To the best of our knowledge, in all existing works for a feature-based image classification system, the final HOG feature vector, which is the concatenation of the block histograms (block features), is first generated, and then the SVM weights are multiplied to the final HOG feature vector. One of our main contributions to the hardware implementation is that the "SVM Weights Accumulation" is performed on the block features rather than the final HOG feature vector. This provides the capability of parallel execution of the "SVM Weights Accumulation" and the binary HOG feature extraction process. Details of the hardware implementation of the "SVM Weights Accumulation" module and its

parallel execution with other modules are described in the following paragraphs.

As mentioned in the previous subsection, for an image of size $n \times m$ and a cell of $c \times c$ pixels with the block overlap of one cell, there are $(n/c-1)\times(m/c-1)$ blocks per image. We will define $(n/c-1)\times(m/c-1)$ as the value of $k$ for simplicity from now on. It has also been discussed in Subsection 3.1.1 that the size of the final HOG feature vector is $NumberOfBlocks \times BlockHistogramSize$ with $BlockHistogramSize$ being a binary vector of length $1 \times 16$. Therefore, the size of the final HOG feature vector and SVM weights are $1 \times 16k$. In our design, the SVM weights are stored in 10-bit registers. We determined that 10 bits of precision is a good balance for accuracy-area trade-off, through trial and error. Therefore, there is a $1 \times 16k$ array of 10-bit registers for each binary classifier to store SVM weights. Ten sets of weights for ten binary classifiers are extracted from the MATLAB model and stored into the ten arrays in hardware.

In the hardware implementation of the SVM classifier, we benefit from the fact that the SVM weights vector can be seen as $k$ batches of $1 \times 16$ weights where each batch corresponds to the features of a block, as shown in Figure 5.9. Therefore, each weight vector is split into $k$ batches, and based on the block number, each batch is loaded into the weight array shown in Figure 5.10. In this way, the SVM weights accumulation can be performed in parallel with the binary HOG descriptor process. Figure 5.10 shows the architecture that calculates the sum of the SVM weights. Ten replications of this circuit are implemented for ten binary classifiers. As can be seen in the figure, block features are used as the selectors of the multiplexers to implement the multiplier behaviour for binary features. The block features are loaded into the corresponding registers as soon as they are generated in the binary HOG descriptor module.

We employed an array of 15 adders to complete the SVM weights accumulation process for all the block features in one clock cycle. If the block feature is 1, the corresponding weight is added to the accumulation; otherwise, accumulation is passed to the next adder. In Figure 5.10, depending on the value of *feature(1)*, the first adder *sum1* is loaded with zero or *weight(1)* for the first block. For all the subsequent blocks, *sum16*, which contains the partial sum of the SVM weights, is loaded to *sum1*. In this way, *sum1* contains the partial sum of the SVM weights corresponding to all the previous blocks. Using the above-described architecture, block features are processed right after they are generated, and they are no longer needed. Therefore, there is no need to concatenate block features and store the entire final HOG feature. This benefits the hardware implementation by reducing the required memory resources for the final HOG feature vector storage. A $16k$ 1-bit array is needed for the final HOG feature vector storage. In our design, the only intermediate data needs to be saved for the rest of the classification process is the total sum of the SVM weights.

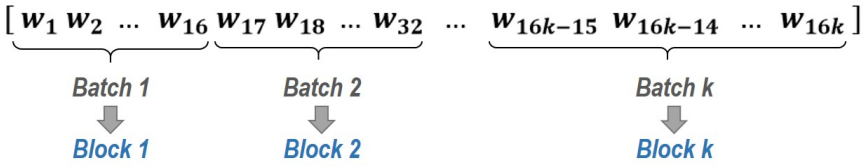



Figure 5.9: SVM weights vector splits into batches

As described above, computation of the SVM weights accumulation for each block is done for all ten binary classifiers in one clock cycle right after the block features are generated. The block features are ready after the "Block Histogram Binarization" step, shown in Figure 5.5. Therefore, the SVM weights accumulation process is completed for each block in parallel with the "Rows Counting" or "Block per Column

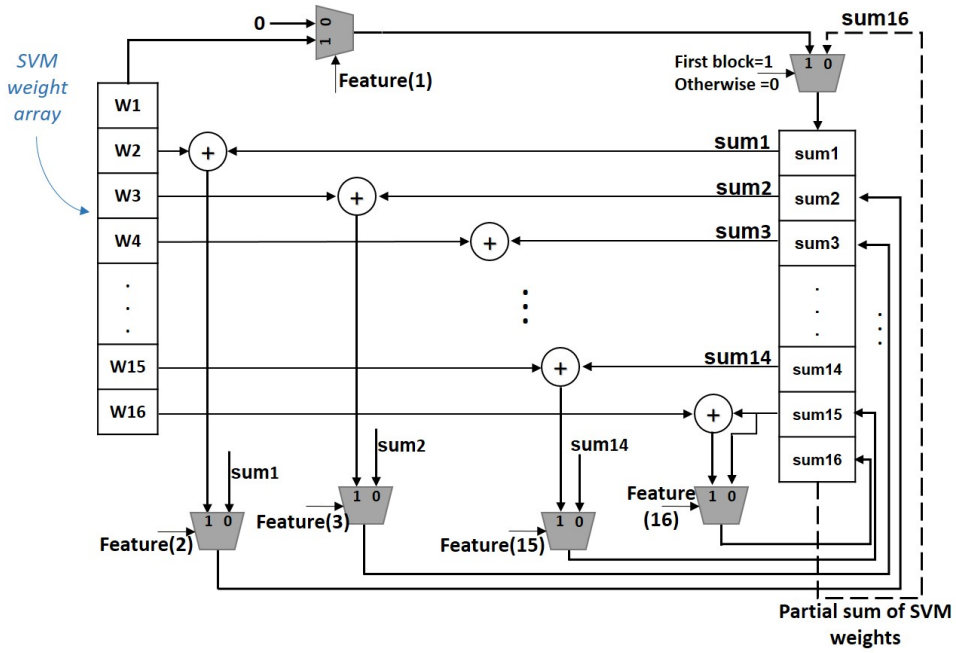



Figure 5.10: SVM weights accumulation module architecture

Counting" step. It can be concluded that after completing the binary HOG feature extraction process for the entire image, the *sum16* register of each binary classifier contains the total sum of the SVM weights for the corresponding binary classifier. At this point, the next step of the binary SVM classification process, which is bias addition, should be applied. Figure 5.11 shows the control flow for the rest of the multi-class SVM classifier.

As mentioned above, the only intermediate data that needs to be passed from the SVM weights accumulation modules (Figure 5.10) to the next processing step is the total sum of the SVM weights. The *sum16* register of each binary classifier contains the total sum of the SVM weights for the corresponding binary classifier. Therefore, the bias value for each binary classifier must be added to the content of the register of that binary classifier. Ten adders are utilized to complete "Bias Addition", as shown in Figure 5.11, for all the ten binary classifiers independently. At this stage, each binary classifier finishes the calculation of its own score for the sample, and the system is ready to start the decision-making process to classify the sample. The rest of Figure 5.11, which are binary classification and majority voting, will be discussed in the following paragraphs.

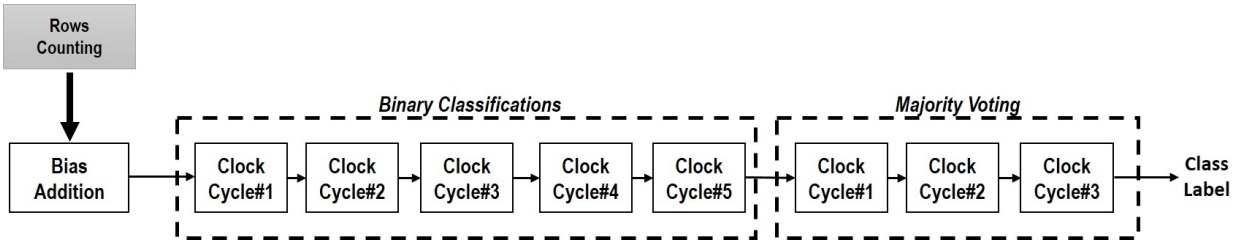



Figure 5.11: Control flow of the five-class SVM classifier

Figure 5.12 depicts the multi-class classification scheme based on the coding design matrix extracted from MATLAB. The coding design is a matrix that shows how a multi-class problem is split into a series of binary classification problems. Each column of the table corresponds to a binary classifier which is implemented by a 2-input comparator, and each row corresponds to a distinct class. Therefore, in our circuit, there are ten comparators to implement ten binary classifiers and five counters to count votes for the five classes.

In clock cycle 1, comparator 1 corresponds to the binary classifier 1, and it assigns the instance into one of two classes 1 or 2; and comparator 2 assigns it into one of two classes 4 or 5. If the score of comparator 1 is greater than zero (shown as 1 in Figure 5.12), it is assigned to class 1; otherwise (shown as -1 in Figure 5.12), it is assigned to class 2. The binary classification process continues for the rest of the comparators according to the coding design matrix. As an example, comparator 1 only works in clock 1, and class 1 can only be voted by comparators 1, 3, 5, and 7. Since each counter can only count one vote every clock cycle, only two comparators can count assigned class label every clock cycle. Therefore, five clock cycles are required to complete the binary classifications for the ten classifiers, as shown in Figure 5.11 and 5.12.

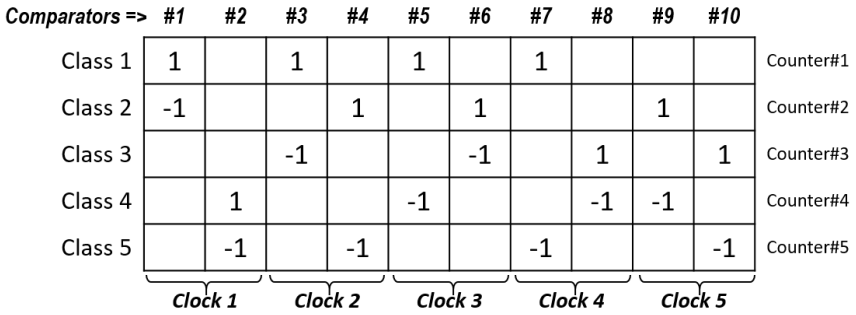



Figure 5.12: The five-class SVM classification scheme based on the coding design matrix

Figure 5.13 shows an example of the multi-class classification process. Corresponding scores to the binary classifiers are shown at the top of the columns. As described earlier, these scores are obtained by adding the bias value for each binary classifier to the content of the register of that binary classifier. In other words, these scores are output of "Bias Addition" in Figure 5.11. According to the coding design matrix shown in Figure 5.12, comparator 1 assigns the sample to class 1 because its score is positive, and comparator 2 votes for class 5 because its score is negative. At the end of the fifth clock cycle, the counters contain the number of votes for each class. In this example, counter 1 to 5 count 2, 0, 2, 2, 4, respectively. Therefore, in the last step of the classification process which is vote counting, this example is classified as class 5.

To find the final class label, the highest value among the counters should be determined. Figure 5.14 shows the hardware implementation of the majority voting. Four comparators are utilized to compare the votes for the five classes. As shown in the figure, this process is performed in three clock cycles to find the final class label. Therefore, three clock cycles are considered in Figure 5.11 for the majority voting process.

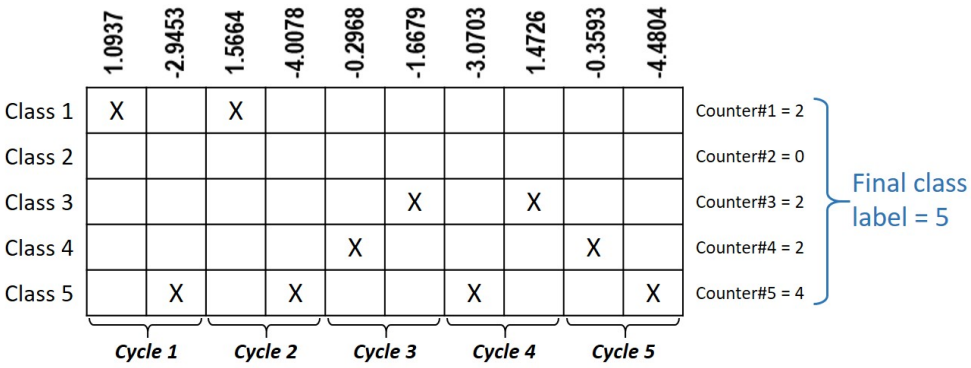



Figure 5.13: A multi-class classification example

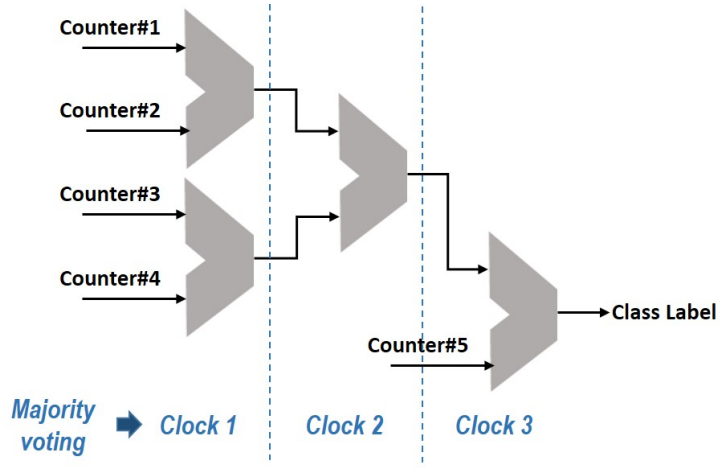



Figure 5.14: The majority voting architecture

## 5.2 Binary Dual-feature Classification System

The algorithmic level block diagram of the proposed binary dual-feature system is shown in Figure 5.15. It can be seen that the binary dual-feature system consists of the binary single-feature system plus the LBP feature descriptor module. As can be seen, the HOG and LBP feature extraction processes execute in parallel. In addition, the HOG and LBP feature combination is performed at the same clock cycle that bias addition is performed.

Similar to the binary single-feature system, the system input is a stream of pixels from the grayscale image, and the output is the class label. The major part of the binary dual-feature system is the same as the binary single-feature system. Therefore, in this section, we only described details of the additional modules, indicated with a red dashed box in Figure 5.15, added to the binary single-feature system to create the binary dual-feature system. The differences between the two systems are also addressed.

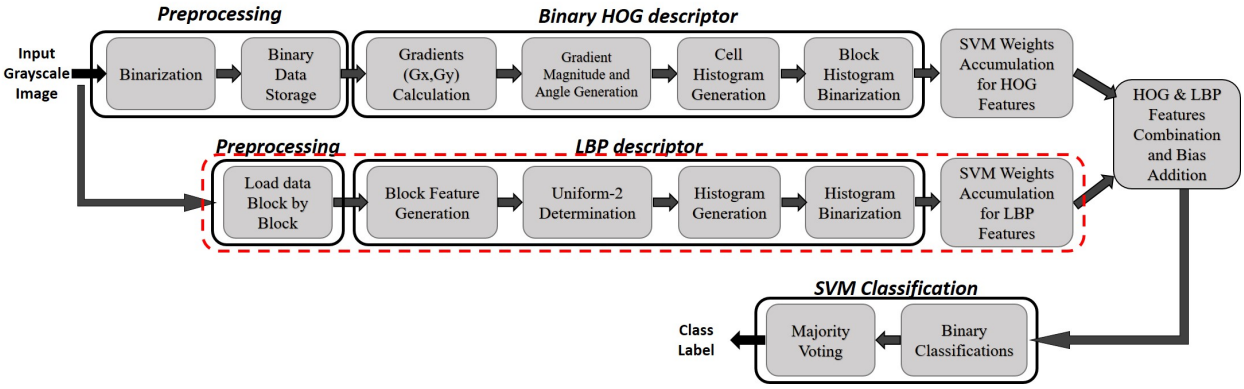



Figure 5.15: Algorithmic level block diagram of the proposed binary dual-feature classification system

### 5.2.1 Binary Dual-feature System Preprocessing Phase

The input to the system is a stream of grayscale pixels. Similar to the binary single-feature system, HOG features are extracted from binary pixels. Therefore, it is the same binarization process as the binary single-feature system in the preprocessing phase of the binary HOG descriptor of the binary dual-feature system. As described in Subsection 4.1.2, the LBP features are extracted from blocks of $3 \times 3$ grayscale pixels. Therefore, the first three rows of the input image are needed to start the LBP feature extraction process. As shown in Figure 5.16, we used three $m \times 8$-bit First-In First-Out (FIFO) memories to store input grayscale pixels and transfer them block-by-block to the LBP feature descriptor module; where $m$ is the number of pixels per row of the image. The control module manages to load FIFOs with the input grayscale image row-by-row. It takes $3m$ clock cycles for the initial load of the first three rows of the input image into the FIFOs. The control module also manages to transfer grayscale pixels to the "Block Feature Generation" sub-module.

### 5.2.2 Binary LBP Features Extraction Phase

As mentioned in Section 3.2, we implemented the Uniform LBP algorithm as the LBP feature descriptor in our binary dual-feature system. Figure 5.16 shows four LBP feature descriptor sub-modules, including Block Feature Generation, Uniform-2 Determination, Histogram Generation, and Histogram Binarization. First, a block feature is generated by comparing the center grayscale pixel with its eight surrounding grayscale pixels in the "Block Feature Generation" sub-module. After that, the block feature is categorized based on the 58 possible uniform-2 patterns, as shown in Section 3.2, in the "Uniform-2 Determination" sub-module. The block histogram is then generated by accumulating block features assigned to each uniform-2 pattern in the "Histogram Generation" sub-module. Finally, in the "Histogram Binarization" sub-module, a binarization process is applied to the histogram to generate the binary LBP feature set. This subsection describes details of the LBP descriptor sub-modules that work in a pipeline manner.

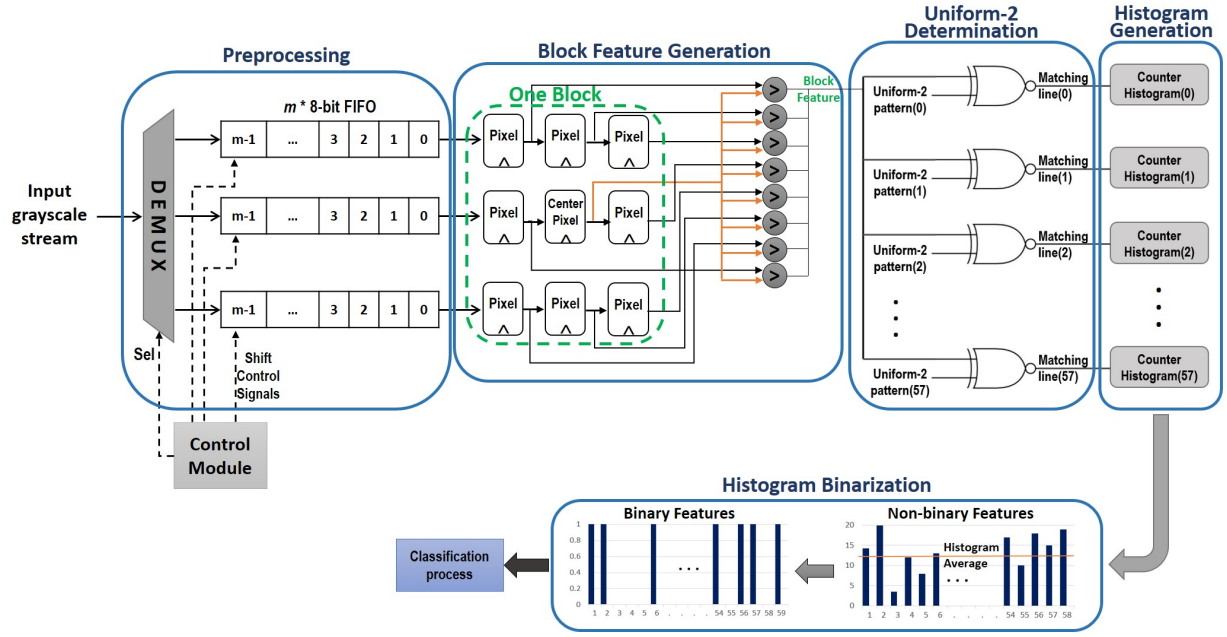



Figure 5.16: Preprocessing and LBP feature descriptor module: the three FIFOs in the "Preprocessing" sub-module and nine registers in the "Block Feature Generation" sub-module are loaded with grayscale pixels.

**Block Feature Generation:** The LBP features are extracted from blocks of $3 \times 3$ grayscale pixels; each grayscale pixel is 8 bits. Thereby, the "Block Feature Generation" sub-module comprises nine 8-bit registers and eight 8-bit comparators. The control module applies three 8-bit shifts to the FIFOs to load one block of grayscale pixels into the nine registers. Following this, the comparators compare the center pixel with its eight surrounding pixels. The comparison results are combined as an 8-bit resultant to generate the block feature and pass it to the next sub-module.

**Uniform-2 Determination:** In the "Uniform-2 Determination" sub-module, there is an array of 58 8-bit registers to store the uniform-2 patterns. The block feature should be compared with all the uniform-2 patterns. This is performed using 58 XNOR gates; each gate has two 8-bit inputs. The hardware implementation of the "Uniform-2 Determination" sub-module is similar to the one in [31]. We used the presented architecture in [31] because it could achieve at least a 37.12% higher speed and 7.7 times fewer LUTs in

uniform-2 pattern recognition compared to similar existing works, as reviewed in Subsection 2.7.2.

**Histogram Generation:** The next sub-module is "Histogram Generation", which consists of 58 counters to count the number of block features that have been matched with each of the uniform-2 patterns. The output of the XNOR gates, labeled as matching lines in Figure 5.16, are used as enable signals for the counters. Whenever a block feature is matched with any uniform-2 patterns, the corresponding counter is incremented by one. The maximum possible value for counters happens when all block features are matched with one uniform-2 pattern. Therefore, the maximum possible value for counters is the number of blocks per image. As discussed in Subsection 4.2.2, for the LBP descriptor, block features are extracted from non-overlapped blocks of $3 \times 3$ grayscale pixels. Thus, there are $[m/3]$ blocks per three rows and $[n/3]$ blocks per three columns ( where [] is the floor function), as shown in Figure 5.17. For an image of $n \times m$ and cell of $3 \times 3$ pixels, there are $[n/3] \times [m/3]$ blocks per image. Hence, the maximum possible value of the counters is $[n/3] \times [m/3]$. As the image size in our binary dual-feature system is $64 \times 64$, the maximum possible value of the counters is $21 \times 21 = 441$. Thus, we used 9-bit counters ($441 = (110111001)_2$) in our hardware implementation.

At this point, the computation of one block feature is completed. The block feature extraction process that has been described above should be continued in a raster manner, as shown in Figure 5.17. The block feature generation, uniform-2 determination, and histogram generation take four clock cycles for each block. As there are $[m/3]$ blocks per three rows, the block feature extraction process for three rows takes $4 \times [m/3]$ clock cycles.

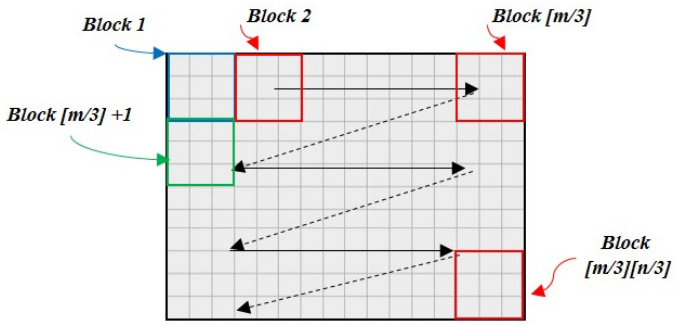



Figure 5.17: Input image blocks scanning scheme for LBP descriptor for an image of $n \times m$ pixels and blocks of $3 \times 3$ pixels (Note: [] is the floor function)

**Histogram Binarization:** After completing the block feature extraction for the entire image, the counters contain the histogram of the LBP feature set. In the final step, binarization should be applied to the histogram to generate binary features. If the histogram value is greater than the threshold, the binary feature is set to 1; otherwise, it is set to 0. The threshold value has been obtained empirically by averaging LBP feature values for randomly selected images from the dataset. The threshold value is hard-coded in the hardware implementation for the histogram binarization because our experiments show that the threshold value does not considerably affect accuracy performance. The "Histogram Binarization" sub-module consists of one 9-bit comparator and 58 registers, as counters in the "Histogram Generation" sub-module are 9 bits and non-binary features (block histogram) have 58 values. The "Histogram Binarization" sub-module requires 58 clock cycles to complete the binarization task.

As described in Subsection 2.2.2, all non-uniform-2 LBP features are considered as one pattern, which

is the 59th value in the histogram. It is described earlier in this subsection that the histogram average is used as the threshold for binarization. According to our experiments, each image contains a considerable number of non-uniform-2 LBP features, so the number of non-uniform-2 LBP features (the 59th value in the histogram) is always greater than the histogram average. Thereby, there is no need to count them, and the 59th binary LBP feature is hard-coded as 1 in our hardware implementation.

Before ending this subsection, we formulated the processing time of the LBP feature extraction for an image of $n \times m$ grayscale pixels. After completing the block feature extraction for the first three rows, the FIFOs should be loaded with new samples from the input image to continue block feature extraction for the next three rows of the image. In our hardware implementation, the first FIFO is loaded with new grayscale pixels during the LBP feature extraction process of the previous three rows. Loading the second and third FIFOs with new grayscale pixels takes $2 \times (m + 1)$ clock cycles. As described earlier in this subsection, the block feature extraction process for every three rows of the image takes $4 \times [m/3]$ clock cycles. Therefore, the block feature extraction process for every three rows of the image and loading of new grayscale pixels to process the next three rows, require $4 \times [m/3] + 2 \times (m + 1)$ clock cycles. We also discussed earlier in this subsection that the initial load of the first three rows of the input image takes $3m$ clock cycles, the histogram binarization requires 58 clock cycles, and there are $[n/3]$ blocks per three columns. As a result, it takes $3m + [n/3] \times (4 \times [m/3] + 2 \times (m + 1)) + 58$ clock cycles to complete the LBP feature extraction process for an image of $n \times m$ grayscale pixels.

### 5.2.3 Binary Dual-feature System Classification Phase

For the binary dual-feature system's classification phase, an OVO multi-class SVM classifier with a linear kernel is implemented. The five-class SVM model is generated offline using MATLAB, similar to the binary single-feature system. The weights and bias values for ten independent binary classifiers and a coding design matrix are extracted from the model to implement the prediction phase in hardware. Samples of SVM weights and bias values are provided in the appendix. As the LBP feature set is binary, all multiplications of the SVM are replaced by additions in the binary dual-feature system, similar to the binary single-feature system.

As described in Section 3.2, one of the contributions of this research is that there is no HOG-LBP dual-feature vector in our implemented dual-feature system despite all similar existing systems. Instead, the extracted SVM weights vector from MATLAB is split into two separate weight vectors for corresponding weights to the HOG and LBP feature sets. This provides the capability for the parallel execution of the SVM weights accumulation for the HOG and LBP feature sets. That is the reason for having two separate modules for SVM weights accumulation of HOG and LBP features, as shown in Figure 5.15. The "SVM Weights Accumulation for HOG Features" module is the same as the one in the binary single-feature system. The "SVM Weights Accumulation for LBP Features" module consists of one adder and two arrays of 59 registers to store LBP features and the corresponding SVM weights for LBP features, as shown in Figure 5.18. Ten arrays are considered in each SVM weights accumulation module to store weights for ten binary classifiers. The corresponding SVM weights for LBP features are loaded into the SVM weights array based on the binary classifier number. Then, the accumulation process is performed during 59 clock cycles for each binary classifier.

The SVM weights accumulation of the SVM classifier is performed on HOG and LBP features separately;

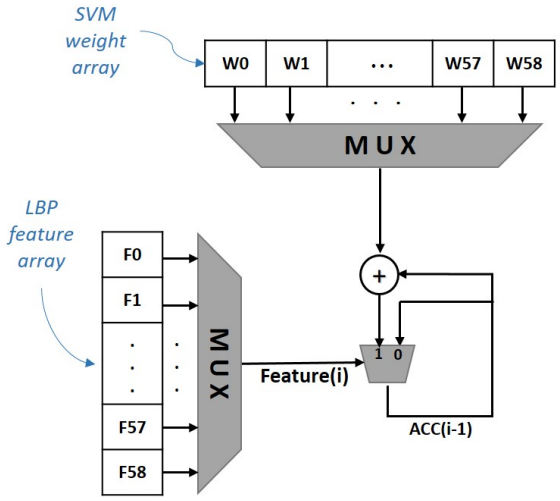



Figure 5.18: SVM weights accumulation module for LBP features

then, they should be added to implement the combination of HOG and LBP feature sets. These addition operations need ten adders to be performed in one clock cycle. Besides, at this point, bias addition should be performed to complete the binary classification that needs ten adders to be completed in one clock cycle. Therefore, 20 independent adders are required to complete these addition operations for all ten binary classifiers in one clock cycle. As described in Subsection 5.1.3, we used 10-bit precision for SVM weights. It is discussed in Subsection 4.2.3 that the size of our binary dual-feature vector is 843 (784 HOG features + 59 LBP features). To add 843 values with 10-bit precision, we need 20-bit adders. Thus, 20 independent 20-bit adders are implemented to perform HOG and LBP features combination and bias addition of the SVM classifier in one clock cycle, as have shown in Figure 5.15. The rest of the classification process is the same as in the binary single-feature system, described in Subsection 5.1.3.

It is mentioned earlier in this subsection that the SVM weights accumulation process for every binary classifier requires 59 clock cycles. Since we performed this process using one adder for all ten binary classifiers in our hardware implementation, it takes $10 \times 59 = 590$ clock cycles to complete the SVM weights accumulation process for ten binary classifiers. As described in Subsection 5.1.3 and shown in Figure 5.11, our hardware implementation needs nine clock cycles for the rest of the SVM classification computation. In total, the processing time of the classification phase of the implemented binary dual-feature system is 599 clock cycles.

The question may arise of why SVM weights accumulation for LBP features is done using one adder while SVM weights accumulation for HOG features is performed using ten accumulators, each one containing 15 adders. The answer is in the processing time difference between HOG and LBP feature descriptors. We provided formulas for the processing time of our implemented the HOG feature descriptor, the LBP feature descriptor, and the SVM classifier based on the image size, block size, and cell size, in previous subsections of this chapter.

Figure 5.19 shows an overall scheme of the processing time for preprocessing, HOG and LBP feature extraction, and SVM classification. As described earlier in this chapter, SVM weights accumulation for the HOG features is performed in parallel with HOG feature extraction. The processing time of these two

processes is shown together in Figure 5.19. Using the formula for our binary HOG descriptor in Subsection 5.1.2, for a cell of $8 \times 8$ binary pixels, preprocessing, feature extraction, and SVM weights accumulation require around $6.5 \times (n \times m)$ clock cycles. For the binary LBP descriptor, preprocessing and feature extraction process take around $3m + n \times m$ clock cycles. It can be inferred that the binary LBP descriptor is faster than the binary HOG descriptor. Therefore, there is no need to accelerate SVM weights accumulation for LBP features. As discussed earlier in this subsection, the SVM weights accumulation is performed one-by-one for ten binary classifiers using one adder in 590 clock cycles to minimize resource utilization. Even with the sequential processing for LBP features, the preprocessing, feature extraction and SVM weights accumulation for the binary HOG descriptor take longer than the same processes for the binary LBP feature descriptor, as shown in Figure 5.19. This is because the HOG algorithm is more compute-intensive than the Uniform LBP algorithm. Thus, there is a wait time after binary LBP feature extraction, for the binary HOG feature set generation to complete. After that, the rest of the SVM classification process can be performed on the HOG-LBP feature set, as shown in Figure 5.11.

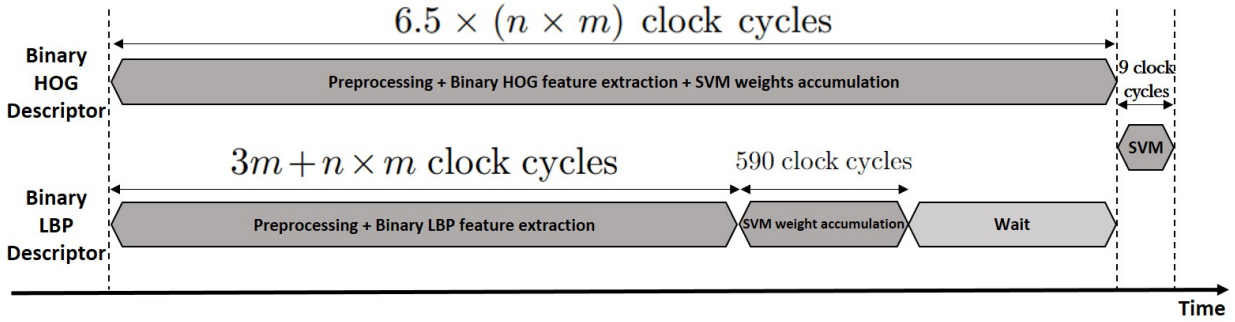



Figure 5.19: Processing time difference between HOG and LBP descriptors

It can be concluded from the above discussion that the processing time of the binary dual-feature system is the same as the binary single-feature system that includes processing time of preprocessing, binary HOG feature extraction and SVM classification. The processing time of the binary HOG feature descriptor is formulated in Subsection 5.1.2, which is $n \times m + 3 + ((n/c - 1) \times (m/c - 1)) \times (4 \times ((2c - 1) + 4) + 3)$ clock cycles. The SVM classification requires 9 clock cycles, as described in Subsection 5.1.3. As a result, our proposed architectures for binary multi-class classification systems require $n \times m + 3 + ((n/c - 1) \times (m/c - 1)) \times (4 \times ((2c - 1) + 4) + 3) + 9$ clock cycles to classify an image of $n \times m$ grayscale pixels, with cell of $c \times c$ pixels for the HOG descriptor.

## 5.3 Summary of the Hardware Implementation

This chapter presents our Register-transfer Level (RTL) implementation for two proposed binary single-feature and dual-feature multi-class classification systems. An RTL implementation can be automatically generated using High Level Synthesis (HLS). HLS is a design process that compiles a functional description of a design into an RTL implementation. However, our proposed architectural level techniques for hardware acceleration can not be realized and satisfied by HLS. This section summarizes our main contributions to the hardware implementation.

- The preprocessing phase of the HOG feature descriptor consisting of binarization and binary image storage, is executed in a pipelined manner with the gradient calculations and magnitude and angle generation. This pipelined processing accelerates these processes in the hardware implementation.

- We proposed a new method to implement feature concatenation without creating a dual HOG-LBP feature set. In our method, the classifier weight vector is split into two separate vectors for corresponding weights to the HOG and LBP features. The SVM weights accumulation is performed on the HOG and LBP features separately; then, they are added together to combine feature sets. The proposed method is an algorithmic level technique for hardware acceleration. This method accelerates the classification process by parallelizing a significant portion of the classification process for the two feature sets. Moreover, the processing time of our binary single-feature system and binary dual-feature system is the same due to the use of this method. The acceleration from this method depends on the feature vector size; the longer the feature vector, the more acceleration compared to not using this method.

- Due to the use of binary features, all multiplications of the SVM classifier are replaced with additions. This replacement is an algorithmic level technique that accelerates the SVM classification process and also results in area efficiency in the implemented SVM classifier module. The theoretical analysis of this replacement's acceleration and area reduction is presented in Subsection 3.1.2. Acceleration and area reduction resulting from this replacement depends on the feature vector size; the longer the feature vector, the more acceleration compared to non-binary features. The hardware implementation results, which concur with the theoretical analysis, are presented in Subsection 6.1.2.

- In our hardware implementation, the SVM weights accumulation is performed on the block HOG features rather than the final HOG feature vector. This way, SVM weights accumulation can be performed in parallel with the HOG feature descriptor. As the HOG feature vector is long (784 features) compared to the Uniform LBP feature vector (59 features), this parallel processing accelerates the classification process. Both our binary single-feature and dual-feature systems benefit from this method; however, the dual-feature system gains more advantages due to having a longer feature vector. The acceleration resulting from this algorithmic level enhancement method depends on the HOG feature vector size; the longer the HOG feature vector, the more acceleration compared to not using this method.

- We implemented the OVO five-class SVM classifier with parallel execution of the SVM weights accumulation for all the ten binary classifiers with the HOG feature extraction process. The proposed parallel processing approach is an architectural level enhancement. This parallel processing makes our

proposed architecture flexible to be easily extended for a different number of object classes with a minor effect on processing time. Changing the number of object classes can be done by repeating the SVM weights accumulation module to implement more binary classifiers, and minor changes in the majority voting architecture to support more classes.

# Chapter 6

# Experimental Results, Performance Evaluation and Comparison

In Chapter 4, we introduced the methodology of our experiments to evaluate the performance of our proposed binary single-feature and dual-feature systems for human detection and multi-class classification. Details of the training phase are presented in Chapter 4. In this chapter, we present details of our experiments and comparison with existing works for hardware and software implementations. For both software and hardware implementations, the SVM model is generated offline in MATLAB; then, the parameter values are extracted to implement the prediction phase. The software implementation of the prediction phase is performed in MATLAB. The hardware implementation of the prediction phase is simulated using VHDL code in Vivado [85]. Details about the implementation of the prediction phase in hardware and software are provided in this chapter.

Since the main focus of this research is hardware implementation, we focused on the experimental results of the hardware implementation in this chapter. The performance of the hardware implementation can be measured in terms of two metrics: accuracy performance, and hardware-related characteristics such as resource utilization and processing speed. To evaluate hardware resource utilization, we used a Xilinx Kintex FPGA [87] to implement our proposed architectures for binary single-feature and dual-feature systems, as presented in Chapter 5. We provide FPGA implementation characteristics of our proposed architectures, including LUTs, Slice Registers, DSP modules, Block RAM (BRAM), and maximum operating frequency. In general, processing speed means the time interval between getting input and producing output. In this work, processing speed for a feature descriptor means time to extract a feature vector from an image, for a classifier module means time to classify a feature vector, and for a classification system means time to classify an image (includes both feature extraction and classification).

As described in Section 3.1, the feature space dimension affects the resource utilization and processing speed of a feature-based classification system because the dimension of the feature vector determines the number of required operations in both the feature descriptor and classifier modules. However, the required number of features for a reliable accuracy performance is application-dependent. Moreover, the required hardware resources directly relate to the number of classes in a multi-class classification system. Therefore, it is not straightforward to have a fair comparison of hardware implementations targeting different applications,

using different feature descriptors, different number of classes, and kernel type.

To make a comprehensive comparison, we present our experimental results and compare our proposed systems with existing solutions in two sections. Section 6.1 provides FPGA implementation characteristics of every single module of our binary classification systems as well as the one-class and multi-class classification systems in five subsections. For accuracy performance comparison, Section 6.2 presents results in two subsections for human detection and multi-class classification applications. Section 6.3 summarizes FPGA implementation characteristics and accuracy performance of our implemented binary systems for human detection and multi-class classification applications.

## 6.1 FPGA Implementation Characteristics

To investigate the hardware-related characteristics of our proposed binary model, we compared FPGA implementation characteristics of our binary single-feature and dual-feature systems with existing works using five different comparisons. Subsection 6.1.1 compares FPGA implementation characteristics of our binary HOG module with existing works to investigate area optimization of our binary HOG descriptor compared to the original HOG descriptor. In Subsection 6.1.2, we investigate area and speed optimization resulted from replacing normalization with binarization in our binary HOG feature descriptor. Subsection 6.1.3 presents comparison of FPGA implementation characteristics of our multi-class SVM classifier module with similar existing works. FPGA implementation characteristics of our LBP descriptor module is compared with existing solutions in Subsection 6.1.4. In Subsection 6.1.5, FPGA implementation characteristics of our binary dual-feature system are compared with existing human detection systems. Subsection 6.1.6 provides comparison of FPGA implementation characteristics of our binary multi-class classification system with existing solutions.

### 6.1.1 FPGA Implementation Characteristics of the Binary HOG Descriptor Module

As described in Subsection 5.1.2, the binary HOG feature extraction process for an image of $n \times m$ and a cell of $c \times c$ binary pixels takes $3n + ((n/c-1) \times (m/c-1)) \times (4 \times ((2c-1)+4)+3)$ clock cycles in our hardware implementation. Therefore, it takes 4,063 clock cycles to complete the computation of the binary HOG descriptor for an image of $64 \times 64$ and a cell of $8 \times 8$ binary pixels. Therefore, at a 44 MHz operating frequency, the processing time of our binary HOG descriptor module is 89.4 $\mu s$. Table 6.1 compares FPGA characteristics of our binary HOG descriptor module with [29], [37], [62], [88], and [46]. Table 6.1 depicts that our binary HOG descriptor implementation is the most area-efficient architecture in four area-related parameters including: utilized LUTs, slice registers, DSP blocks, and block RAM (BRAM); compared to the other presented works in the table.

Processing speed of the HOG feature descriptor depends on the image size and operating frequency. As existing solutions have different frame sizes, we used pixels per clock period unit as suggested in [88] to have a fair processing speed comparison. Pixels per second unit is calculated using Equation 6.1. On this scale, our binary HOG descriptor implementation achieves the highest processing speed, as shown in Table 6.1.

$$PixelsPerSecond = \frac{FrameSize}{ProcessingTime \times MaximumOperatingFrequency} \tag{6.1}$$

Our hardware implementation for the binary HOG descriptor could achieve the most area-efficient and fastest architecture among existing works in Table 6.1. This is because of two steps of binarization in our proposed binary HOG descriptor, including input image binarization and replacing normalization with binarization, as described in Section 3.1. In Subsection 3.1.1, it is described theoretically how extracting HOG features from a binary image makes each processing step of the HOG descriptor faster and more area-efficient. Presented experimental results in Table 6.1 concur with the analysis is provided in Subsection 3.1.1.

Table 6.1: Comparing FPGA implementation characteristics of the proposed binary HOG descriptor with existing solutions

| | Proposed | [29] | [37] | [62] | [88] | [46] |
|---|---|---|---|---|---|---|
| **Dimension of Feature Space** | 784 | 864 | 1,980 | 3,780 | 3,840 | 34,596 |
| **Device** | Xilinx Kintex | ALTERA Stratix II | Spartan 3 | Virtex 5 | Cyclone 5 | Zynq 7000 |
| **LUT** | 2,192 | 3,794 | 28,616 | 3,924 | 8,610 | 7,226 |
| **Slice Register** | 1,511 | 6,699 | Not Provided | 3,642 | 17,697 | 12,462 |
| **DSP Blocks** | 0 | 12 | 18 | 12 | 74 | 26 |
| **BRAM (kbits)** | 0 | Not Provided | 1800 | 936 | 326 | 432 |
| **Operating Frequency (MHz)** | 44 | 127 | 63 | 270 | 162 | 125 |
| **Processing Time** | 89.4 µs | 5.2 µs | 312 µs | 150 µs | 1.9 ms | 16.67ms |
| **Frame Size** | $64 \times 64$ | $16 \times 32$ | $48 \times 96$ | $64 \times 128$ | $640 \times 480$ | $256 \times 256$ |
| **Scan Stride (pixels)** | 8 | 4 | 8 | 8 | 8 | 8 |
| **Cell Size (pixels)** | $8 \times 8$ | $4 \times 4$ | $8 \times 8$ | $8 \times 8$ | $8 \times 8$ | $8 \times 8$ |
| **Block Size (cells)** | $2 \times 2$ | $3 \times 3$ | $2 \times 2$ | $2 \times 2$ | $1 \times 1$ | $2 \times 2$ |
| **Pixels per Clock Period** | 1.04 | 0.775 | 0.234 | 0.202 | 0.997 | 0.031 |

### 6.1.2 Area and Speed Optimization of Replacing Normalization with Binarization

It is theoretically described in Subsections 3.1.1 and 3.1.2 that replacing normalization with binarization in the binary HOG feature descriptor results in reduction in hardware resource utilization and speed increase in both the binary HOG descriptor and the SVM classifier modules. As described in Section 4.2, one of the contributions of this research is utilizing binary LBP features, with the normalization step replaced by

binarization in the LBP descriptor. Therefore, it is important to investigate how this replacement affects the hardware implementation.

In order to investigate hardware resource reduction and speed increase resulted from replacing the normalization process of the original HOG descriptor with binarization, we implemented two binary HOG descriptors. In the first implementation, the binary HOG descriptor, described in Subsection 5.1.2 is implemented, followed by L2-norm normalization, described in Subsection 3.1.1. In the second implementation, normalization is replaced by the binarization process, described in Subsection 5.1.2. The image size is $64 \times 128$ binary pixels in the hardware implementation.

Characteristics of both FPGA implementations are presented in Table 6.2. Reported results for each implementation consist of two cases: the binary HOG descriptor module and the binary object classification system. The binary object classification system consists of the binary HOG descriptor module followed by a one-class SVM classifier, described in Subsection 5.1.3.

As can be seen in Table 6.2, the binary HOG descriptor utilized 4,992 LUTs in the first implementation with normalization, and 4,053 LUTs in the second implementation with binarization. Therefore, replacing normalization with binarization results in a 19% (4,992 - 4,053 = 940 LUTs) reduction in the hardware resources for the binary HOG descriptor. In addition, one-class SVM classifier occupies only 396 LUTs (4,449 - 4,053 = 396) in the second implementation, while it needs 1,827 LUTs (6,819 - 4,992 = 1,827) in the first implementation. It can be inferred that the one-class SVM classifier of the first implementation requires 4.6 times more hardware resources than the second implementation. This resource reduction in the SVM classifier is the outcome of replacing normalization with binarization in the HOG descriptor, which results in replacing multiplications with additions in the SVM classifier, as described in Subsection 3.1.2.

Besides, comparing the required clock cycles for processing one window shows a 55% (11,556 - 5,157 = 6,399 clock cycles) reduction in the processing time of the second implementation compared to the first implementation.

It can be concluded that replacing normalization with binarization leads to a more area-efficient and faster architecture. That is because all multiplications of the SVM classification algorithm are replaced by additions due to the binary features, as described in Subsection 3.1.2. Moreover, the binarization process is faster and more area-efficient on hardware implementation than the normalization process, as described in Subsection 3.1.1. As described in Section 3.2, the normalization step of the LBP descriptor is replaced with a binarization step, similar to the HOG descriptor. It can be concluded that there is a similar area and speed optimization, as described above, in our hardware implementation of the LBP descriptor.

Table 6.2: Area and speed optimization of replacing normalization with binarization in HOG descriptor

| | **Implemented Architecture** | **LUT** | **Slice Register** | **#Clock Cycle per Window** |
|---|---|---|---|---|
| **First Implementation (Normalization)** | Binary HOG | 4,992 | 1,664 | 11,556 |
| | Binary object detection system | 6,819 | 1,822 | 11,557 |
| **Second Implementation (Binarization)** | Binary HOG | 4,053 | 1,640 | 5,157 |
| | Binary object detection system | **4,449** | 1,676 | **5,159** |

### 6.1.3 FPGA Implementation Characteristics of the Multi-class SVM Module

As described in Subsection 5.1.3, we split the multi-class SVM classification process into two parts in our hardware implementation: SVM weights accumulation and the rest of the SVM computation, including bias addition, decision making of binary classifications, and majority voting. In our hardware implementation, SVM weights accumulation is executed in parallel with the HOG feature extraction process to speed up the classification process. The rest of the SVM computation takes only nine clock cycles for five-class classification, regardless of the frame size. Therefore, our hardware implementation needs only nine clock cycles for five-class classification since SVM weights accumulation does not affect the processing time of the classification process.

However, to have a fair comparison of our multi-class SVM classifier speed with existing works, the processing time of the SVM weights accumulation should be considered; thus, we calculated it here. As described in Subsection 5.1.3, the proposed architecture for SVM weights accumulation requires one clock cycle to complete the process for one block. We need one additional clock per block to load data to the implemented SVM weights accumulation module. Therefore, the SVM weights accumulation processing time depends on the number of blocks per image; two clock cycles for each block. It is described in Subsection 5.1.2 that for an image of $n \times m$ and cell of $c \times c$ pixels, there are $(n/c-1) \times (m/c-1)$ blocks per image. As a result, for an image of $64 \times 64$ and cell of $8 \times 8$ pixels, there are 49 blocks per image. It can be concluded that our five-class SVM classifier module needs $2 \times 49 = 98$ clock cycles for the SVM weights accumulation and nine clock cycles for the rest of the SVM classification, thereby $98 + 9 = 107$ clock cycles in total. The maximum clock frequency of our multi-class SVM classifier module is 103 MHz. Therefore, at 103 MHz operating frequency, the processing time of the implemented multi-class SVM classifier module is 1.03 $\mu s$.

Table 6.3 compares FPGA implementation characteristics of our multi-class SVM classifier module with similar existing works in [15], [41], [68], [73], [42], and [71]. All the previous works used for the comparison in Table 6.3 are based on a non-binary feature set because there is no previous multi-class SVM classifier that used a binary feature set. Therefore, the presented comparisons in this subsection show area and speed optimization resulting from feeding a binary feature set to the multi-class SVM classifier.

The processing time of a multi-class SVM classifier module directly relates to the number of features and number of classes. As existing solutions have a different number of features and number of classes, to have a fair processing speed comparison, we proposed to use time per class per feature unit. This unit is calculated as shown in Equation 6.2.

$$TimePerClassPerfeature = \frac{ProcessingTime}{\#Classes \times \#Features} \tag{6.2}$$

On this scale, Table 6.3 depicts that our FPGA implementation for multi-class SVM classifier is far faster than existing solutions. This is because we used ten replications of the SVM weights accumulation modules, as described in Subsection 5.1.3, to speed up the SVM computation.

The SVM weights and bias values are stored as arrays using LUTs in our FPGA implementation instead of RAM or ROM. The FPGA resource utilization of a multi-class SVM classifier directly relates to the number of features, kernel type, and the number of classes. Targeting various applications affects the required number of features, implementing different types of the kernel, and a different number of classes by existing solutions makes the FPGA resource utilization comparison difficult.

Replacing multiplications with additions in our SVM classifier module resulted from binary features

leads to hardware resource reduction of our multi-class SVM classifier. As can be seen in Table 6.3, our proposed system is the third most area-efficient architecture after [41] and [42]. The area utilization is highly dependent on the feature space's dimension because the feature vector's dimension determines the number of required operations in the SVM module, as described in Subsection 3.1.2. Presented work in [41] is the most area-efficient architecture because it has the fewest number of features among existing works and the fewest number of classes with the linear kernel. Presented work in [42] is the second most area-efficient architecture among other works in Table 6.3. There is no information about the processing time of their system; therefore, we can not make a fair comparison. However, there is 6.5 times fewer features in [42] compared to our architecture. Therefore, it is reasonable to have fewer resource utilization compared to ours.

Table 6.3: Comparing FPGA implementation characteristics of the proposed binary multi-class SVM classifier module with existing solutions

| | Proposed | [15] | [41] | [68] | [73] | [42] | [71] |
|---|---|---|---|---|---|---|---|
| **Dimension of Feature Space** | 784 | 500 | 7 | 51 | 1,024 | 120 | 8 |
| **Device** | Xilinx Kintex | Virtex 5 | Virtex 4 | ML510 | Cyclone 2 | Virtex 6 | Virtex 2 |
| **LUT** | 5,595 | 38,179 | 748 | 6,511 | 14,064 | 1,072 | 11,943 |
| **Slice Register** | 477 | 9,646 | 1,422 | 2,026 | Not Provided | 922 | 1,576 |
| **DSP Blocks** | 0 | 52 | 27 | 24 | 20 | 7 | 64 |
| **BRAM (kbits)** | 0 | 576 | 496 | 144 | 64 | 0 | 14,600 |
| **Processing Time** | 1.03 $\mu s$ | 0.25 ms | 1.4 ms | Not Provided | 2 ms | Not Provided | 14.18 ms |
| **Number of Classes** | 5 | 5 | 3 | 3 | 4 | 6 | 3 |
| **SVM Type** | OVO * | OVA * | pairwise | OVA * | OVO * | Not Provided | OVO * |
| **Kernel Type** | Linear | RBF | Linear | Linear | Hardware friendly | Polynomial | RBF |
| **Dataset** | Caltech-256 | Caltech-256 | Persian handwritten digits | Facial expressions | COIL dataset | Facial expressions | TIMIT corpus |
| **Time per Class per Feature** | 0.26 ns | 0.1 $\mu s$ | 66.7 $\mu s$ | Not Provided | 4.8 $\mu s$ | Not Provided | 590 $\mu s$ |

* OVO: One-Vs-One
* OVA: One-Vs-All

### 6.1.4 FPGA Implementation Characteristics of the Binary LBP Descriptor Module

Comparing the FPGA implementation characteristics of our LBP descriptor module with existing works is not straightforward because of the difference in reported results among existing works. Presented works in [65], [28], [53], and [64] reported hardware utilization for the LBP descriptor followed by a classifier or a prediction module. A face detection system using the LBP descriptor is presented in [16]. They reported the hardware resource utilization for the entire face detection system. As described in Subsection 2.7.2, authors in [31] proposed integrating LBP descriptor with content-addressable memory (CAM). They reported FPGA resource utilization and matching speed of their system for one LBP pattern, not the entire image. The only work reported hardware resource utilization for the LBP descriptor is presented in [30]; however, it does not generate binary LBP features.

As described in Subsection 5.2.2, our hardware implementation needs $3m + [n/3] \times (4 \times [m/3] + 2 \times (m + 1)) + 58$ clock cycles to complete the LBP feature extraction process for an image of $n \times m$ grayscale pixels. Therefore, our hardware implementation requires 4,744 clock cycles to generate the binary LBP feature set of an image of $64 \times 64$ grayscale pixels. At operating frequency of 416 MHz, the processing time of our LBP descriptor is 11.4 $\mu s$.

Table 6.4 shows FPGA characteristics of our implemented LBP descriptor and [30]. It can be seen that our LBP descriptor is faster and more area-efficient than [30]. This comparison shows area and speed optimization resulting from binarization in our LBP feature descriptor.

Table 6.4: Comparing FPGA implementation characteristics of the proposed LBP descriptor module with a similar existing solution

| | Proposed | [30] |
|---|---|---|
| **Dimension of Feature Space** | 59 | 59 |
| **Device** | Xilinx Kintex | Virtex-6 |
| **LUT** | 738 | 971 |
| **Slice Register** | 710 | 1,022 |
| **DSP Blocks** | 0 | 4 |
| **BRAM (kbits)** | 0 | 262 |
| **Operating Frequency (MHz)** | 416 | 393 |
| **Processing Time of Feature Extraction** | 11.4 $\mu s$ | 0.7 ms |

### 6.1.5 FPGA Implementation Characteristics of the Binary One-class Classification System

Table 6.5 compares FPGA resource utilization of our proposed binary single-feature one-class classification system with existing works in [47], [49], [62], [37], [29], and [36]. We compared FPGA characteristics of our binary single-feature system because the purpose of Table 6.5 is hardware resource utilization comparison, and other existing works used a single-feature set. The image size is $64 \times 128$ binary pixels to have a fair comparison with existing works. As mentioned in Section 4.1, our proposed binary single-feature and dual-feature systems could not achieve reliable accuracy performance for human detection. A system that can not satisfy the accuracy requirement is worthless. However, we compared FPGA resource utilization to evaluate the area efficiency of our binary one-class classification system in the case that other researchers may use this system for other applications.

As can be seen, presented works in [47], and [49] utilized fewer LUTs compared to our proposed architecture. As reviewed in Subsection 3.1.1, presented works in [47], and [49] proposed using binary images to extract HOG features and replacing normalization with binarization, respectively, that both techniques result in a reduction in hardware resource utilization. Therefore, they achieved a fewer number of LUTs compared to others. Since our implemented binary HOG descriptor is the combination of proposed works in [47] and [49], it is expected to result in a more area-efficient design than both of [47] and [49]. It is described in Chapter 5 that there are several parallel processing units in our architecture, such as an array of adders to perform SVM weights accumulation for one image block in one clock cycle, an array of adders for gradient magnitudes accumulation, an array of subtractors for gradients generation on x and y axis ($G_x$ and $G_y$) for one row in one clock cycle. These parallel processing units lead to increasing hardware resource utilization.

All of the works in Table 6.5, except [47], used grayscale images. It can be observed that while our binary single-feature system implementation shows a fairly average clock frequency and processing rate, it could achieve at least five times and eight times fewer LUTs and slice registers, respectively, compared to existing works that used grayscale images. Moreover, unlike the other presented works, our binary single-feature implementation does not require any DSP block and BRAM.

Table 6.5: Comparing FPGA implementation characteristics of the proposed binary one-class classification system with existing solutions (all used grayscale images except [47], [49] replaced normalization with binarization)

| | Proposed | [47] | [49] | [62] | [37] | [29] | [36] |
|---|---|---|---|---|---|---|---|
| **Dimension of feature space** | 1,680 | 1,680 | 3,780 | 3,780 | 1,980 | 864 | 3,780 |
| **Device** | Xilinx Kintex | Artix 7 | Spartan3 | Virtex5 | Spartan 3 | Altera Stratix 2 | Virtex7 |
| **LUT** | 4,449 | 3,500 | 3,379 | 38,535 | 28,616 | 37,940 | 19,241 |
| **Slice Register** | 1,676 | 2,988 | 2,602 | 42,987 | Not Provided | 66,990 | 14,312 |
| **DSP Blocks** | 0 | Not Provided | Not Provided | 357 | 18 | 120 | 19 |
| **BRAM (kbits)** | 0 | Not Provided | 108 | 7,128 | 1,800 | Not Provided | 36 |

### 6.1.6 FPGA Implementation Characteristics of the Binary Multi-class Classification System

As reviewed in Section 2.1, CNN-based multi-class classification systems show high accuracy performance for image classification. References [25] and [26] surveyed accelerating CNN inference on FPGAs and showed FPGA resource utilization of CNN inference is far more than feature-based systems. It can be concluded that comparing CNN-based classification systems with feature-based classification systems is not fair. Therefore, this section compares our binary multi-class dual-feature classification system with similar existing feature-based multi-class classification systems in [89], [15], and [52] in Table 6.6. Table 6.6 includes FPGA implementation characteristics and some other specifications of a multi-class classification system, including: feature descriptor type, number of classes, classifier and kernel type.

As mentioned earlier, it is not straightforward to compare different works that used different number of features, different feature descriptors, different number of classes, and kernel type. As can be seen in Table 6.6, our hardware implementation for binary dual-feature multi-class classification system is the most area-efficient architecture among the presented works in the table, although we have the largest feature space and number of classes.

To evaluate processing speed of our multi-class classification system, we used the time of classifying one image. As described in Section 5.2, classification time of our binary dual-feature multi-class classification system for an image of $n \times m$ and cell of $c \times c$ grayscale pixels is $n \times m + 3 + ((n/c - 1) \times (m/c - 1)) \times (4 \times ((2c - 1) + 4) + 3) + 9$ clock cycles. Therefore, our hardware implementation requires 7,979 clock cycles to classify one image of $64 \times 64$ grayscale pixels, with cell of $8 \times 8$ grayscale pixels. At an operating frequency of 44 MHz, it takes 181.3 $\mu s$ to classify one image. Table 6.6 depicts that our binary dual-feature multi-class classification system is far faster than existing works.

In short, our hardware implementation for the binary dual-feature multi-class classification system could achieve the most area-efficient and fastest architecture among existing works in Table 6.6. As described in Subsection 6.1.1, it is because of two steps of binarization that results in area and speed optimization in both HOG and LBP feature descriptors and SVM classifier.

Table 6.6: Comparing FPGA implementation characteristics of the proposed binary dual-feature multi-class classification system with existing solutions

| | Proposed dual-feature system | [89] | [15] | [52] |
|---|---|---|---|---|
| **Dimension of Feature Space** | 843 | Not Provided | 500 | 4,932 |
| **Device** | Xilinx Kintex | Virtex 6 | Virtex 5 | Stratix IV |
| **LUT** | 10,293 | 113,359 | 59,821 | 14,189 |
| **Slice Register** | 4,376 | 75,071 | 23,770 | 10,662 |
| **DSP Blocks** | 0 | 72 | 64 | Not Provided |
| **BRAM (kbits)** | 0 | 119 | 128 | 554.25 |
| **Operating Frequency (MHz)** | 44 | 25 | 60 | 130 |
| **Classify One Image Time** | 181.3 $\mu s$ | 16.6 ms | 6.4 ms | 16.6 ms |
| **Feature Descriptor** | HOG+LBP | HOG | SIFT | HOG+LBP |
| **Number of Classes** | 5 | 3 | 5 | 4 |
| **Classifier** | SVM | SVM | SVM | Softmax |
| **Kernel Type** | Linear | Linear | RBF | Not Applicable |

## 6.2 Accuracy Performance Evaluation and Comparison

For accuracy performance comparison, we presented results in two subsections for human detection and multi-class classification applications. We presented the experimental results of our proposed binary human detection system in Subsection 6.2.1. Subsection 6.2.2 provides details of classification accuracy performance measurement and comparison of our binary single-feature and dual-feature multi-class classification systems with similar existing work that used the same datasets [15].

### 6.2.1 Detection Accuracy Evaluation and Comparison

As mentioned in Chapter 4, our proposed binary single-feature and dual-feature systems could not achieve reliable accuracy performance for human detection, although, they could outperform previous works for multi-class classification. This section presents more details of how our binary single-feature and dual-feature systems perform for human detection. As described in Section 4.1, our binary dual-feature system outperforms the binary single-feature system. Therefore, to evaluate the accuracy performance for human detection, we focused on the binary dual-feature system in this section.

It is described in Section 4.1 that this experiment is only performed using software implementation because our proposed binary single-feature and dual-feature systems could not achieve satisfying accuracy performance for human detection. The proposed binary single-feature and dual-feature systems, described in Chapter 3, have been simulated in MATLAB using parameter values described in Subsection 4.1.2. We used the MATLAB function *fitcsvm* to train the one-class SVM classifier.

As described in Section 4.1, the INRIA dataset includes people with various poses, clothing, appearance, background, and illumination. Our experiments show that the accuracy of our binary dual-feature system for human detection depends on different factors in images, as described in the following paragraphs.

Figure 6.1 shows some examples of the INRIA dataset in RGB and binary format. Green and yellow rectangles in Figure 6.1 show detection bounding boxes according to the ground truth dataset and detection results of our binary dual-feature system, respectively. The ground truth dataset is the annotation included in the dataset to determine correct detection bounding boxes. The purpose of Figure 6.1 is to illustrate the accuracy performance for binary images. The RGB format of images are shown because binary format does not represent details. If a detected bounding box ($BB_{dt}$) and a ground truth bounding box ($BB_{gt}$) overlap sufficiently, the detection is counted as a true positive. According to the Intersection over Union (IoU) unit [90], the overlap area, defined as Equation 6.3, must exceed 50%.

$$OverlapArea = \frac{area(BB_{dt} \cap BB_{gt})}{area(BB_{dt} \cup BB_{gt})} \tag{6.3}$$

It can be seen that the detection result of our binary dual-feature system for examples of the INRIA person dataset shown in Figure 6.1 is counted as true positive detection. False positive and false negative rates are zero for these examples. Common features of these examples are that each image includes one person close to the camera, and the background has a simple pattern.

Figure 6.2 shows some other examples of the INRIA person dataset in RGB and binary format. These examples include one person per image, similar to Figure 6.1 with a more complex background. It can be seen that the detection accuracy performance of our binary dual-feature system for examples shown in Figure 6.2 decreased as compared to the examples shown in Figure 6.1. False positive and false negative rates are

Figure 6.1: Examples of the INRIA person dataset include one person in RGB and binary format. Green and yellow rectangles show detection bounding boxes according to the ground truth dataset and detection results of our binary dual-feature system, respectively. The false positive and false negative rates are zero. The RGB format of images is only shown to represent details that are not clear in the binary format.

not zero in these examples.

Figure 6.2: Examples of the INRIA person dataset include one person in RGB and binary format. Green and yellow rectangles show detection bounding boxes according to the ground truth dataset and detection results of our binary dual-feature system, respectively. False positive and false negative rates are not zero in these examples. The RGB format of images is only shown to represent details that are not clear in the binary format.

Figure 6.3 shows some examples of the INRIA dataset in RGB and binary format, including images with more than one person. It can be seen that the detection accuracy performance of our binary dual-feature system for examples shown in Figure 6.3 degraded as compared to the examples shown in Figure 6.2. It

can be inferred that the accuracy performance of our binary system is highly dependent on the number of persons per image, distance from camera, background complexity, and illumination.

Figure 6.3: Examples of the INRIA person dataset include more than one person in RGB and binary format. Green and yellow rectangles show detection bounding boxes according to the ground truth dataset and detection results of our binary dual-feature system, respectively. The RGB format of images is only shown to represent details that are not clear in the binary format.

To evaluate accuracy performance of our binary single-feature and dual-feature systems in human detection, we attempted to reproduce presented work in [4] in MATLAB using the function "*hog_feature_vector*". However, we could not reproduce their reported results, therefore, we presented both of our reproduced curve and an estimated curve based on their reported results in Figure 4.4. Figure 6.4 compares the performance of our proposed binary single-feature and dual-feature systems and the presented work in [4]. As described in Chapter 4, presented work in [4] is the primary reference of existing works for accuracy performance evaluation for human detection due to its reliable accuracy performance. As described in Section 4.1, we only performed software algorithmic implementation for human detection because our proposed binary single-feature and dual-feature systems could not achieve satisfying accuracy performance for human detection. It can be seen that the binary dual-feature system outperforms the binary single-feature system, but the original work presented in [4] outperforms both of our proposed binary systems. The reason is that presented work in [4] used grayscale images and non-binary features, as described in Chapter 3. In contrast, we used binary images and binary features that eliminate some information, as will be discussed in detail in the following paragraphs.

A binary pixel can include only two levels of colors, black and white, and is represented by one bit, while a grayscale pixel can illustrate 256 shades of gray, between black and white, and represented by 8 bits. Therefore, a binary image includes fewer details than a grayscale image. Thus, features extracted from a binary image contain less information than features extracted from a grayscale image. As a result, the first step of binarization in our binary HOG descriptor, utilizing binary images instead of grayscale images to extract features, removes some details during feature extraction computation.

A non-binary feature is represented by at least eight bits, while a binary feature is represented by one bit. The second binarization step in our binary HOG descriptor, replacing normalization with binarization, converts non-binary features to binary features and thus eliminates some information. Feeding binary fea-

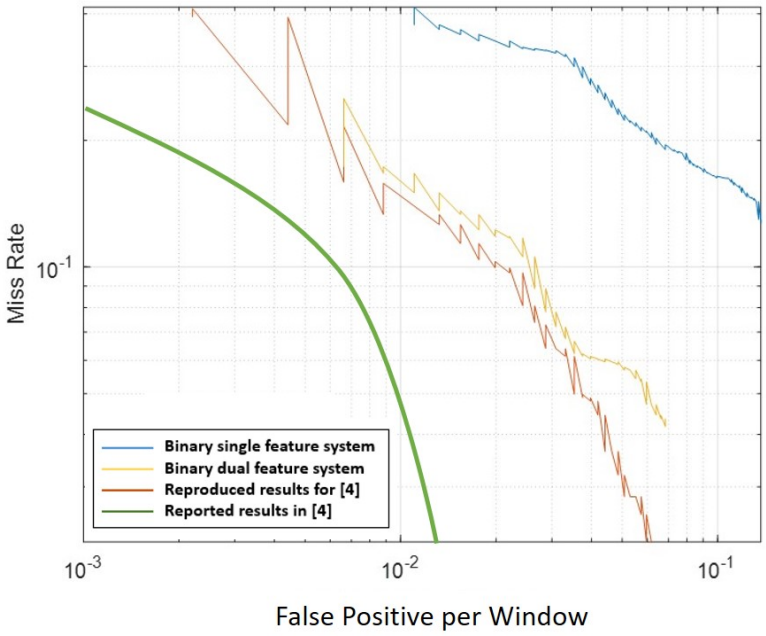



Figure 6.4: The performance of human detection systems on INRIA person dataset

tures to the SVM classifier degrades classification accuracy due to less information. In short, two steps of binarization in our proposed binary model remove a considerable amount of details during feature extraction and classification computation.

As shown above, our binary model could only achieve reliable detection performance for images, including one person who is close to the camera in simple background. Variant appearance, a broad range of poses, background, clothing, and illumination make human detection a challenging problem to solve. Two steps of binarization cause our proposed binary model to have high false positive and false negative rates for a challenging application like human detection. Therefore, our proposed binary model fails for human detection in the INRIA person dataset.

### 6.2.2 Classification Accuracy Evaluation and Comparison

To assess the accuracy performance of our binary single-feature and dual-feature systems for multi-class classification, two commonly-used image classification datasets are used: the Caltech-256 dataset [83], and the KUL Belgium Traffic Sign dataset [84]. The setting of the multi-class classification systems and the methodology for the accuracy measurement and comparison are discussed in Section 4.2. This section presents details of the classification accuracy evaluation of our proposed single-feature and dual-feature systems for hardware and software implementations.

For both software and hardware implementations, the five-class SVM model is generated offline using the function "fitcecoc" in MATLAB, then the weights and bias values for ten independent binary classifiers and a coding design matrix are extracted from the model to implement the prediction phase. Details of the training phase is presented in Subsection 4.2.3, and SVM weights and bias values are provided in the appendix. For the software implementation of the prediction phase, the binary HOG feature descriptor as described in Subsection 3.1.1, the Uniform LBP feature descriptor as described in Section 3.2, and the SVM classifier as described in Subsection 3.1.2, are implemented in MATLAB. For the hardware implementation of the prediction phase, the binary single-feature and dual-feature systems, as described in Chapter 5, are simulated using VHDL code in Vivado. Methodologies of the experiments, discussed in Section 4.2, are the same for software and hardware implementations. The HOG features are extracted from binary images and LBP features are extracted from grayscale images.

In order to evaluate the classification accuracy performance, we used the confusion matrix. In a confusion matrix, the diagonal elements show the number of images that have been classified correctly, whereas other elements show misclassified images. In each experiment, the average confusion matrix is computed by averaging the values for five classes over ten subsets which values are very close. We provided the average confusion matrix for software and hardware implementations for each experiment. For accuracy performance evaluation, we used classification rate, the sum of diagonal elements (correctly classified samples), divided by the sample size of the test data, for the 100 samples in our experiments. Since there are 20 samples per class in each experiment, each diagonal element shows how many images, out of 20, are classified correctly for the corresponding class. It is worth emphasizing that we presented the average confusion matrix for each experiment.

The experimental results are reported for both single-feature and dual-feature systems in each subsection. As described in Section 4.2, three experiments are performed to evaluate the accuracy, and results are compared with the presented work in [15]. Reference [52] utilized a HOG-LBP feature set similar to ours, but they did not provide details of their experiment. Therefore, we could not compare the accuracy of our proposed binary model with their system. Subsection 6.2.2.1 provides experimental results of Experiment 1, on the Caltech-256 dataset, for software and hardware implementations and their comparison with presented work in [15]. The classification accuracy comparison on the Caltech-256 dataset is summarized at the end of this subsection. Experimental results of Experiment 2, on Set 1 of the KUL Belgium Traffic Sign dataset (KUL-Set1), for software and hardware implementations and their comparison with presented work in [15] are presented in Subsection 6.2.2.2. Subsection 6.2.2.3 describes experimental results of Experiment 3, on Set 2 of the KUL Belgium Traffic Sign dataset (KUL-Set2), on software and hardware implementations and their comparison with presented work in [15]. Moreover, the classification accuracy comparison on KUL-Set1 and 2 of the KUL Belgium Traffic Sign dataset is summarized at the end of this subsection.

#### 6.2.2.1 Experiment 1: Caltech-256 Dataset

As described in Section 4.2, for Experiment 1, ten subsets from the Caltech-256 dataset are used. There are five classes in each subset: airplane, face, horse, motorbike, and watch, shown as C1, C2, C3, C4, and C5 in confusion matrices in this subsection. Some examples from the subsets are shown in Figure 6.5. The classification rate of the binary single-feature and dual-feature systems are presented in two separate parts in this subsection.


Figure 6.5: Example images from the Caltech-256 dataset [83]

***Part 1: Single-feature System Evaluation - Caltech-256 Dataset***

In the first classification accuracy evaluation experiment, we measured the classification rate of our single-feature system. The average confusion matrices of our binary single-feature system for hardware and software implementations are presented in Table 6.7 and Table 6.8, respectively. The average classification rate (sum of the diagonal elements) over ten subsets for hardware implementation is 87.1% and 89.4% for the software implementation.

Table 6.7: Hardware average confusion matrix of our binary single-feature system on the Caltech-256 dataset

| | **C 1** | **C 2** | **C 3** | **C 4** | **C 5** |
|---|---|---|---|---|---|
| **C 1** | **14.9** | 0.1 | 1.2 | 2.2 | 1.6 |
| **C 2** | 0.4 | **18.5** | 0 | 0.9 | 0.2 |
| **C 3** | 0.1 | 0 | **18.6** | 0.5 | 0.8 |
| **C 4** | 1 | 2 | 0 | **17** | 0 |
| **C 5** | 0.7 | 0.2 | 0.3 | 0.7 | **18.1** |

Table 6.8: Software average confusion matrix of our binary single-feature system on the Caltech-256 dataset

| | **C 1** | **C 2** | **C 3** | **C 4** | **C 5** |
|---|---|---|---|---|---|
| **C 1** | **16.1** | 0.1 | 0.4 | 2.1 | 1.3 |
| **C 2** | 0.5 | **17.4** | 0 | 1.7 | 0.4 |
| **C 3** | 0.1 | 0 | **18.4** | 0.5 | 1 |
| **C 4** | 0 | 1 | 0 | **19** | 0 |
| **C 5** | 0.3 | 0.1 | 0.1 | 1 | **18.5** |

***Part 2: Dual-feature System Evaluation - Caltech-256 Dataset***

In the second part of the experiment, we measured the classification rate of our binary dual-feature system. The average confusion matrices of our binary dual-feature system for hardware and software implementations are presented in Table 6.9 and Table 6.10, respectively. The average classification accuracy over ten subsets

on hardware is 90.6%, and 91.4% for the software implementation.

Table 6.9: Hardware average confusion matrix of our binary dual-feature system on the Caltech-256 dataset

| | **C 1** | **C 2** | **C 3** | **C 4** | **C 5** |
|---|---|---|---|---|---|
| **C 1** | **16.3** | 0.1 | 0.6 | 1.6 | 1.4 |
| **C 2** | 0.6 | **18.5** | 0.1 | 0.7 | 0.1 |
| **C 3** | 0.1 | 0 | **18.7** | 0.7 | 0.5 |
| **C 4** | 0 | 1 | 0 | **19** | 0 |
| **C 5** | 0.7 | 0.3 | 0.3 | 0.6 | **18.1** |

Table 6.10: Software average confusion matrix of our binary dual-feature system on the Caltech-256 dataset

| | **C 1** | **C 2** | **C 3** | **C 4** | **C 5** |
|---|---|---|---|---|---|
| **C 1** | **17.3** | 0 | 0.1 | 2.1 | 0.5 |
| **C 2** | 1 | **17.9** | 0 | 1.1 | 0 |
| **C 3** | 0.1 | 0 | **18.6** | 0.7 | 0.6 |
| **C 4** | 0 | 0 | 0 | **20** | 0 |
| **C 5** | 0.5 | 0 | 0.3 | 1.6 | **17.6** |

***Classification Accuracy Comparison on the Caltech-256 dataset***

Table 6.11 summarizes the average classification rate of the presented work in [15] and our binary single-feature and dual-feature classification systems for hardware and software implementations. As seen in Table 6.11, the average classification rate dropped in the hardware implementation compared to the software implementation for the three systems. It is expected because hardware implementation uses fixed-point arithmetic with less precision than the floating-point arithmetic used in the software implementation. Our single-feature classification system could achieve 3.1% (87.1% - 84%) and 4.4% (89.4% - 85%) higher classification rate compared to [15] in hardware and software implementations, respectively. However, it can be seen that combining LBP features with HOG features in our binary dual-feature system increases the classification rate by 3.5% (90.6% - 87.1%) in hardware and 2% (91.4% - 89.4%) in software, compared to the binary single-feature system.

Table 6.11: Multi-class classification rate comparison on the Caltech-256 dataset

| | **Single-feature** | **Dual-feature** | **[15]** |
|---|---|---|---|
| **Average classification rate in hardware implementation** | 87.1% | 90.6% | 84% |
| **Average classification rate in software implementation** | 89.4% | 91.4% | 85% |

#### 6.2.2.2 Experiment 2: KUL Belgium Traffic Sign Dataset - KUL-Set1

As described in Section 4.2, for Experiment 2, ten subsets from the KUL Belgium Traffic Sign dataset are used. There are five classes in each subset: crossing for cyclists, danger, one way traffic, parking lot, and dead end, shown as C1, C2, C3, C4, and C5 in confusion matrices in this subsection. Figure 6.6 and 6.7 illustrate some examples of our KUL-Set1 from the KUL Belgium Traffic Sign dataset in the RGB and binary format, respectively. The classification rate of the binary single-feature and dual-feature systems are presented in two separate parts in this subsection.


Figure 6.6: Example images from the KUL Belgium Traffic Sign dataset [84] in RGB format - KUL-Set1

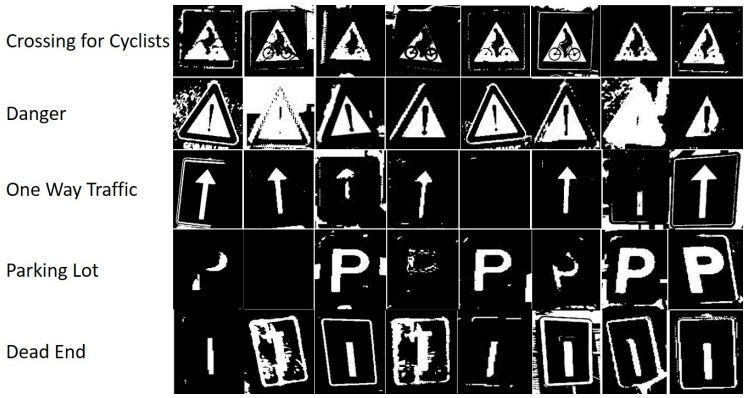



Figure 6.7: Example images from the KUL Belgium Traffic Sign dataset in binary format - KUL-Set1

***Part 1: Single-feature System Evaluation - KUL Belgium Traffic Sign Dataset - KUL-Set1***

In the first part of Experiment 2, we measured the classification rate of our single-feature system for KUL-Set1. The average confusion matrices of our binary single-feature system for hardware and software implementations are presented in Table 6.12 and Table 6.13, respectively. The average classification rate (sum of the diagonal elements) over ten subsets for hardware implementation is 79.6% and 84% for the software implementation.

As seen in Table 6.12, the hardware implementation of our binary single-feature system achieved an

Table 6.12: Hardware average confusion matrix of our binary single-feature system on the KUL-Set1

| | **C 1** | **C 2** | **C 3** | **C 4** | **C 5** |
|---|---|---|---|---|---|
| **C 1** | **18.5** | 0.3 | 0 | 1.2 | 0 |
| **C 2** | 3.1 | **11.4** | 1.4 | 4.1 | 0 |
| **C 3** | 0.5 | 0.5 | **14.9** | 3.2 | 0.9 |
| **C 4** | 0.9 | 0 | 2.9 | **16.2** | 0 |
| **C 5** | 0 | 0.2 | 0.5 | 0.7 | **18.6** |

Table 6.13: Software average confusion matrix of our binary single-feature system on the KUL-Set1

| | **C 1** | **C 2** | **C 3** | **C 4** | **C 5** |
|---|---|---|---|---|---|
| **C 1** | **18.8** | 0.8 | 0.4 | 0 | 0 |
| **C 2** | 1.5 | **14.7** | 2.3 | 0.5 | 1 |
| **C 3** | 0 | 0 | **16.6** | 0 | 3.4 |
| **C 4** | 0.1 | 0.1 | 0.3 | **19.2** | 0.3 |
| **C 5** | 0.9 | 0 | 4.4 | 0 | **14.7** |

average of 11.4 correctly classified samples out of 20 test samples for C2, which is the lowest diagonal element. In Table 6.12, the highest average misclassification for C2 is between C2 and C4 (4.1) and C2 and C1 (3.1). C1 and C2 correspond to "Crossing for Cyclists" and "Danger" object classes with triangle shapes. The reason for the worse classification performance for C1 and C2, compared to other classes, is the similarity of their shapes which leads to misclassification easily. Our binary model uses binary images containing fewer details than grayscale images. Therefore, it shows worse classification performance for objects with a similar shape.

Table 6.12 shows that the hardware implementation of our binary single-feature system obtained an average of 14.9 correctly classified samples out of 20 test samples for C3, which is the second lowest diagonal element. In Table 6.12, the highest average misclassification for C3 is between C3 and C4 (3.2). C3 and C4 correspond to "One Way Traffic" and "Parking Lot" object classes, with close shapes in their binary patterns, as shown in Figure 6.7.

As seen in Table 6.13, the software implementation of our binary single-feature system achieved an average of 14.7 correctly classified samples out of 20 test samples for C2 and C5, which is the lowest diagonal element. In Table 6.13, the highest average misclassification for C2 is between C2 and C3 (2.3), and the highest average misclassification for C5 is between C5 and C3 (4.4). C3 and C5 correspond to "One Way Traffic" and "Dead End" object classes with similar shapes in their binary patterns, as shown in Figure 6.7.

The above discussion shows that our binary single-feature system's accuracy in identifying the classes strongly depends on how distinctive object shapes are, especially in their binary format. The similarity in shapes of the objects has a tangible effect on the chance of misclassification. With that said, higher misclassification occurs when we have maximum similarity between object classes. Therefore, the model behaviour is reasonable considering the mentioned deviation in different classes.

***Part 2: Dual-feature System Evaluation - KUL Belgium Traffic Sign Dataset - KUL-Set1***

In the second part of Experiment 2, we measured the classification rate of our binary dual-feature system. The average confusion matrices of our binary dual-feature system on hardware and software implementations are presented in Table 6.14 and Table 6.15, respectively. The average classification accuracy over ten subsets for the hardware implementation is 84.8%, and 88.5% for the software implementation.

As seen in Table 6.14, the hardware implementation of our binary dual-feature system achieved an average of 15.5 correctly classified samples out of 20 test samples for C3, which is the lowest diagonal element. Table 6.14 shows that C3 is only misclassified as C5 (4.5). C3 and C5 correspond to "One Way Traffic" and "Dead End" object classes with close patterns in their binary pattern, as shown in Figure 6.7.

As seen in Table 6.15, the software implementation of our binary dual-feature system achieved an average

Table 6.14: Hardware average confusion matrix of our binary dual-feature system on the KUL-Set1

| | C 1 | C 2 | C 3 | C 4 | C 5 |
|---|---|---|---|---|---|
| **C 1** | **17.6** | 1 | 0.9 | 0 | 0.5 |
| **C 2** | 1 | **16** | 1.6 | 0 | 1.4 |
| **C 3** | 0 | 0 | **15.5** | 0 | 4.5 |
| **C 4** | 0 | 0.2 | 0.2 | **19.2** | 0.4 |
| **C 5** | 0.4 | 0.5 | 2.2 | 0.4 | **16.5** |

Table 6.15: Software average confusion matrix of our binary dual-feature system on the KUL-Set1

| | C 1 | C 2 | C 3 | C 4 | C 5 |
|---|---|---|---|---|---|
| **C 1** | **19.2** | 0.8 | 0 | 0 | 0 |
| **C 2** | 1.5 | **16.3** | 1.4 | 0.5 | 0.3 |
| **C 3** | 0 | 0 | **17.5** | 0 | 2.5 |
| **C 4** | 0 | 0.4 | 0.2 | **19.2** | 0.2 |
| **C 5** | 0.9 | 0 | 2.8 | 0 | **16.3** |

of 16.3 correctly classified samples out of 20 test samples for C2 and C5, which is the lowest diagonal element. In Table 6.15, the highest average misclassification for C2 is between C2 and C1 (1.5), with triangle shapes. The highest average misclassification for C5 is between C5 and C3 (2.8), with similar shapes in their binary pattern, as shown in Figure 6.7.

#### 6.2.2.3 Experiment 3: KUL Belgium Traffic Sign Dataset - KUL-Set2

Experiment 2 shows that our binary model has higher misclassification between object classes with similar patterns than object classes with distinct patterns. For example, the binary format of "Crossing for Cyclists" and "Danger" classes in KUL-Set1 have a similar triangle pattern. Likewise, the "One Way Traffic" and "Dead End" have a similar pattern in their binary format, as shown in Figure 6.7. Therefore, we evaluated the classification accuracy performance of our binary model for another set of the KUL Belgium Traffic Sign dataset to investigate its performance further. In our KUL-Set2, we replaced "Crossing for Cyclists" and "Dead End" classes of KUL-Set1 with "Give Way" and "Speed Limit" object classes, respectively, to eliminate similarity between object classes. Figure 6.8 and 6.9 illustrate some examples of our KUL-Set2 from the KUL Belgium Traffic Sign dataset in the RGB and binary format, respectively. It can be seen that the binary format of the five classes in KUL-Set2 (Figure 6.9) has more distinctive patterns compared to KUL-Set1 (Figure 6.7).

As described in Section 4.2, Experiment 3 has the same condition as Experiment 2, using KUL-Set2. Ten randomly subsets are selected from five classes including: give way, danger, one way traffic, parking lot, and speed limit, shown as C1, C2, C3, C4, and C5 in confusion matrices in this subsection. The classification rate of the binary single-feature and dual-feature systems are presented in two separate parts in this subsection.


Figure 6.8: Example images from the KUL Belgium Traffic Sign dataset [84] in RGB format - KUL-Set2

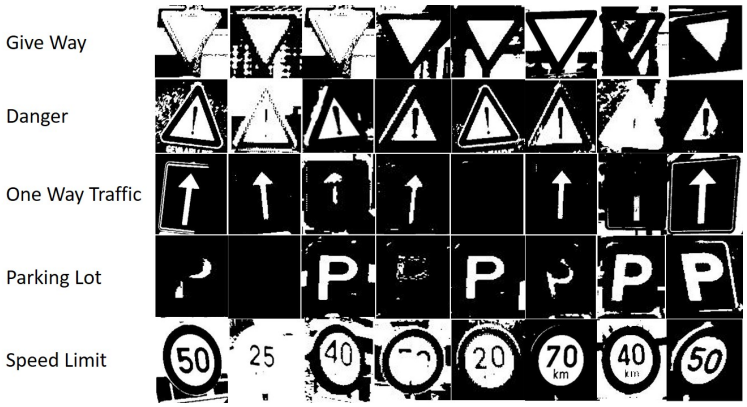



Figure 6.9: Example images from the KUL Belgium Traffic Sign dataset in binary format - KUL-Set2

***Part 1: Single-feature System Evaluation - KUL Belgium Traffic Sign Dataset - KUL-Set2***

In the first part of Experiment 3, we measured the classification rate of our binary single-feature system. The average confusion matrices of our binary single-feature system on hardware and software implementations are presented in Table 6.16 and Table 6.17, respectively. The average classification rate of our binary single-feature system over ten subsets for the hardware implementation is 88.3%, and 90.1% for the software implementation.

Table 6.16: Hardware average confusion matrix of our binary single-feature system on the KUL-Set2

| | **C 1** | **C 2** | **C 3** | **C 4** | **C 5** |
|---|---|---|---|---|---|
| **C 1** | **19.3** | 0 | 0 | 0.3 | 0.4 |
| **C 2** | 0.4 | **14.4** | 4.5 | 0.7 | 0 |
| **C 3** | 0.4 | 0.2 | **18.5** | 0.8 | 0.1 |
| **C 4** | 0 | 0.4 | 1.3 | **18.3** | 0 |
| **C 5** | 0.2 | 0.6 | 0.9 | 0.5 | **17.8** |

Table 6.17: Software average confusion matrix of our binary single-feature system on the KUL-Set2

| | **C 1** | **C 2** | **C 3** | **C 4** | **C 5** |
|---|---|---|---|---|---|
| **C 1** | **19.4** | 0.1 | 0.2 | 0.1 | 0.2 |
| **C 2** | 0 | **16.8** | 2.5 | 0.7 | 0 |
| **C 3** | 0.3 | 1.2 | **17.3** | 0.6 | 0.6 |
| **C 4** | 0 | 1.2 | 0.4 | **18.4** | 0 |
| **C 5** | 0.1 | 1.2 | 0.1 | 0.4 | **18.2** |

As seen in Table 6.16, the hardware implementation of our binary single-feature system achieved an average of 14.4 correctly classified samples out of 20 test samples for C2, which is the lowest diagonal element. In Table 6.16, the highest average misclassification for C2 is between C2 and C3 (4.5).

Table 6.17 illustrates that the software implementation has a similar accuracy performance to the hardware implementation. As seen in Table 6.17, the software implementation of our binary single-feature system achieved an average of 16.8 correctly classified samples out of 20 test samples for C2, which is the lowest diagonal element. In Table 6.17, the highest average misclassification for C2 is between C2 and C3 (2.5). C2 and C3 correspond to the "Danger" and "One Way Traffic" object classes. As shown in Figure 6.9, among five object classes in KUL-Set2, the "One Way Traffic" object class has a faded and vague pattern in its binary format. Therefore, our binary single-feature system had higher misclassification when compared to the other object classes in KUL-Set2.

***Part 2: Dual-feature System Evaluation - KUL Belgium Traffic Sign Dataset - KUL-Set2***

The average confusion matrices of our binary dual-feature system on KUL-Set2 for hardware and software implementations are presented in Table 6.18 and Table 6.19, respectively. The average classification rate of our binary dual-feature system over ten subsets for the hardware implementation is 91.8%, and 94% for the software implementation.

As seen in Table 6.18, the hardware implementation of our binary dual-feature system achieved an average of 16 correctly classified samples out of 20 test samples for C2, which is the lowest diagonal element. In Table 6.18, the highest average misclassification for C2 is between C2 and C3 (2.5).

Table 6.19 illustrates that the software implementation has a similar accuracy performance to the hardware implementation. As seen in Table 6.19, the software implementation of our binary dual-feature system achieved an average of 16.4 correctly classified samples out of 20 test samples for C2, which is the lowest diagonal element. In Table 6.18, the highest average misclassification for C2 is between C2 and C3 (2.4). C2 and C3 correspond to "Danger" and "One Way Traffic" object classes. As can be seen in Figure 6.9, among five object classes in KUL-Set2, "One Way Traffic" object class has a faded and vague pattern in its binary format. Therefore, our binary dual-feature system shows higher misclassification when compared to the other object classes in KUL-Set2.

Table 6.18: Hardware average confusion matrix of our binary dual-feature system on the KUL-Set2

| | **C 1** | **C 2** | **C 3** | **C 4** | **C 5** |
|---|---|---|---|---|---|
| **C 1** | **19.8** | 0 | 0 | 0.1 | 0.1 |
| **C 2** | 0.6 | **16** | 2.5 | 0.7 | 0.2 |
| **C 3** | 0.1 | 0.3 | **19.4** | 0.1 | 0.1 |
| **C 4** | 0.1 | 0.3 | 1.3 | **18.3** | 0 |
| **C 5** | 0.3 | 0.3 | 0.6 | 0.5 | **18.3** |

Table 6.19: Software average confusion matrix of our binary dual-feature system on the KUL-Set2

| | **C 1** | **C 2** | **C 3** | **C 4** | **C 5** |
|---|---|---|---|---|---|
| **C 1** | **20** | 0 | 0 | 0 | 0 |
| **C 2** | 0 | **16.4** | 2.4 | 0 | 1.2 |
| **C 3** | 0.2 | 0.1 | **19.3** | 0.4 | 0 |
| **C 4** | 0 | 0 | 0.7 | **18.5** | 0.8 |
| **C 5** | 0.1 | 0 | 0 | 0.1 | **19.8** |

***Classification Accuracy Comparison for KUL-Set1 and KUL-Set2***

Table 6.20 summarizes the classification rate of Experiment 2 and Experiment 3, performed using KUL-Set1 and KUL-Set2 collected from the KUL Belgium Traffic Sign dataset. As can be seen in Table 6.20, the classification rate dropped in the hardware implementation compared to the software implementation for all systems. It is expected because hardware implementation uses the fixed-point arithmetic with less precision than the floating-point arithmetic used in the software implementation. It can be seen that our proposed binary systems outperform the architecture in [15] in all the cases.

Table 6.20 illustrates that our binary single-feature and dual-feature systems could achieve higher classification rate for KUL-Set2 compared to KUL-Set1. The binary single-feature system achieved 8.7% (88.3% - 79.6%) and 6.1% (90.1% - 84%) higher classification rate for KUL-Set2 compared to KUL-Set1 in hardware and software implementations, respectively. Similarly, the classification rate of the binary dual-feature system for KUL-Set2 is higher than KUL-Set1 by 7% (91.8% - 84.8%) and 5.5% (94% - 88.5%) for hardware and software implementations, respectively.

Moreover, combining LBP features with HOG features increases the classification accuracy in both software and hardware for both KUL-Set1 and KUL-Set2. For KUL-Set1, the classification rate of the binary dual-feature system compared to the single-feature system is increased by 5.2% (84.8% - 79.6%) and 4.5%

(88.5% - 84%) for hardware and software implementations, respectively. For KUL-Set2, the classification rate of the binary dual-feature system compared to the single-feature system is increased by 3.5% (91.8% - 88.3%) and 3.9% (94% - 90.1%) for hardware and software implementations, respectively.

In short, Experiments 2 and 3 show that the classification accuracy performance of our proposed binary model depends on how distinctive the object class patterns are in their binary format. Among reported classification rates for our proposed binary single-feature and dual-feature systems in Table 6.20, the lowest classification rate is related to our single-feature system for KUL-Set1. However, it is still 1.2% higher than presented work in [15] for KUL-Set1. Our dual-feature system could achieve an 6.8% higher classification rate than [15] for KUL-Set1.

Table 6.20: Multi-class classification rate comparison on the KUL Belgium Traffic Sign dataset

| | **Single-feature KUL-Set1** | **Dual-feature KUL-Set1** | **Single-feature KUL-Set2** | **Dual-feature KUL-Set2** | **[15] KUL-Set1** |
|---|---|---|---|---|---|
| **Average classification rate in hardware implementation** | 79.6% | 84.8% | 88.3% | 91.8% | 78% |
| **Average classification rate in software implementation** | 84% | 88.5% | 90.1% | 94% | 80% |

## 6.3 Summary

This chapter provides details of FPGA implementation characteristics and an accuracy performance evaluation of our binary single-feature and dual-feature systems. There is a trade-off between hardware-related characteristics and accuracy performance where higher accuracy needs more hardware resources and processing time. Moreover, the area, processing speed, and accuracy requirements are application-dependent. Therefore, it is not straightforward to compare different existing solutions fairly.

We studied the accuracy performance of our binary model for human detection because the original HOG feature descriptor with SVM classifier showed high accuracy performance for human detection among existing works [4]. It has been shown that our binary model could not perform reliably for human detection on the INRIA person dataset due to the two steps of binarization that remove significant details during feature extraction and classification processes.

As our proposed binary model could not achieve reliable human detection performance, we examined our binary model's suitability for multi-class classification using two challenging datasets, including the Caltech-256 dataset and the KUL Belgium Traffic Sign dataset. Experimental outcomes show that our binary single-feature and dual-feature systems could perform more accurately than the similar existing solution in [15]. It is shown that the accuracy of our binary systems in identifying the classes strongly depends on how distinctive the object shapes are, especially in their binary format. The similarity in shapes of the objects has a tangible effect on the chance of misclassification. The higher misclassification occurs when we have maximum similarity between object classes.

We compared the FPGA implementation characteristics of every single module of our binary classification systems as well as the one-class and multi-class classification systems with existing solutions to make a

fair and comprehensive evaluation of our hardware implementation. Comparing FPGA implementation characteristics of our binary HOG feature descriptor with existing works shows that two steps of binarization make the binary HOG feature descriptor module faster and more area-efficient than existing architectures. Besides, employing a binary feature set leads to area reduction and processing speed increase in the SVM classifier due to replacing all multiplications with additions. Moreover, experimental results demonstrate that combining binary HOG and LBP feature sets improves accuracy performance.

In short, our proposed architecture for the binary single-feature and dual-feature systems is an area-efficient and fast architecture with comparable accuracy for multi-class classification applications.

# Chapter 7

# Conclusions and Future Work

## 7.1 Conclusions

Image classification is a fundamental task in computer vision with many applications, such as pedestrian detection, face detection, surveillance, and robotics. Real-time processing rate and reliable accuracy performance are two primary requirements for a classification system. The feature-based technique is a popular approach to design classification systems. Commonly-used feature extraction and classification algorithms in a feature-based classification system are often computationally expensive. Furthermore, these algorithms have to process a large amount of data, which makes it hard to achieve a real-time processing rate using software implementation [7], [8], [9], [10]. Therefore, researchers have been motivated to employ hardware implementation capabilities to accelerate the classification process. Many works have shown that the FPGA platform is well-suited to implement real-time classification and object detection systems due to its reasonable power consumption and reconfigurable capability [18], [19], [17].

This research addressed the image classification acceleration problem based on a binary feature classification system. Two classification systems are proposed, based on the binary single-feature set and the binary dual-feature set. Area-efficient and fast FPGA-based architectures are implemented for binary single-feature and dual-feature systems, while achieving comparable accuracy for multi-class classification.

The unique contribution of this research is utilizing a *binary dual* feature set. To the best of our knowledge, no previous work employs a binary dual-feature set. Using *binary* features decreases hardware resource utilization in both the feature descriptor and classifier modules, and a *dual* feature set improves the accuracy compared to a single-feature set.

The other major contributions and experimental results of this dissertation can be summarized as follows:

- We implemented an area-efficient and fast architecture for a binary HOG feature descriptor. The FPGA implementation of our binary HOG feature descriptor module occupies 2,192 LUTs and 1,511 registers. Its processing time for an image of $64 \times 64$ and cell of $8 \times 8$ binary pixels is 4,063 clock cycles, with a maximum operating frequency of 44 MHz. The FPGA implementation of our binary HOG feature descriptor module could achieve the most area-efficient and fastest architecture among similar existing solutions reviewed in this research. This is because our implemented binary HOG feature descriptor employs two steps of binarization. It extracts features from binary images and generates a

binary HOG feature set by applying feature binarization. The size of the final feature vector in the binary HOG descriptor is 44% of the feature vector size in the original HOG descriptor, resulting in resource reduction in both the HOG feature descriptor and the SVM classifier.

- We implemented an area-efficient and fast architecture for the binary Uniform LBP feature descriptor. The FPGA implementation of our LBP feature descriptor module occupies 738 LUTs and 710 registers. Its processing time for an image of $64 \times 64$ grayscale pixels is 4,686 clock cycles, with a maximum operating frequency of 416 MHz. The implemented binary Uniform LBP feature descriptor extracts features from grayscale images and generates a binary LBP feature set by applying feature binarization. To the best of our knowledge, our classification system is the first work utilizing the binary LBP feature set.

- We developed a new method to implement feature concatenation without creating a dual HOG-LBP feature set. Instead, the classifier weight vector is split into two separate vectors for corresponding weights to the HOG and LBP features. The SVM weights accumulation is performed on the HOG and LBP features separately; then, they are added together to combine the feature sets. This method accelerates the classification process by parallelizing a significant portion of the classification process for the two feature sets. The acceleration resulting from using this method depends on the feature vector size; the longer the feature vector, the more acceleration compared to not using this method.

- Due to the use of binary features, all multiplications of the SVM classifier are replaced with additions. This replacement accelerates the SVM classification process and also results in area efficiency in the implemented SVM classifier module. Acceleration and area reduction resulting from this replacement depends on the feature vector size; the longer the feature vector, the more acceleration compared to non-binary features.

- In our hardware implementation, the SVM weights accumulation is performed on the block HOG features rather than the final HOG feature vector. This way, SVM weights accumulation can be performed in parallel with the HOG feature descriptor and accelerate the classification process. The acceleration resulting from using this method depends on the HOG feature vector size; the longer the HOG feature vector, the more acceleration compared to not using this method.

- Our implementation for the One-Vs-One multi-class SVM classifier can be easily extended for a different number of object classes with a minor effect on processing time. This is because SVM weights accumulation, the most time-consuming part of the SVM classification, is performed in parallel with the HOG feature extraction process.

- Experimental results show that hardware implementation of our binary single-feature classification system could achieve a comparable classification rate on the Caltech-256 dataset, 87.1%, and the KUL Belgium traffic sign dataset, 88.3%. Moreover, combining binary HOG and LBP feature sets improves the classification rate of our binary multi-class classification system by 3.5%-5.2% for different datasets.

## 7.2 Discussion

As reviewed in Chapter 2, existing works developed different methods to simplify the complex operations of the classification algorithms in hardware implementation. In this research, we proposed a new approach to simplify hardware implementation. Utilizing the binarization approach in various steps of the classification process, we could achieve an area-efficient and fast architecture, which is the main goal of this research. Due to the trade-offs between area utilization and accuracy performance, area optimization is gained at the cost of accuracy loss. However, we showed that our proposed binary approach could achieve comparable accuracy performance with existing solutions. Although our proposed approach could satisfy the classification accuracy requirement, it is unsuitable for applications that require high accuracy. This is because of the three steps of binarization that remove many details from the features.

In this research, we did not perform preprocessing except the binarization because the main goal was to investigate how using binary images and binary features affects the classification performance. Of course, preprocessing may improve the overall classification accuracy. As reported results show, our implemented classification system could achieve comparable classification accuracy with existing solutions without any preprocessing.

Our proposed classification system can process each window of size $64 \times 64$ pixels in 181 $\mu s$. This processing rate is faster than the typical real-time processing rate of 30 fps. On the other hand, as discussed in Chapter 6, our proposed binary approach could not detect objects satisfying for images with complex backgrounds. Therefore, our proposed classification system is suitable for applications that require high-speed processing rate with a solid background. An application that would match our system characteristics is text scanning to recognize text from an image.

The proposed architecture in this research can be implemented in both FPGA and ASIC (Application-Specific Integrated Circuit). In order to evaluate the hardware resource utilization of our proposed architecture, we used Xilinx Kintex FPGA. Furthermore, the flexible parallel processing capability of FPGAs, which is commonly used in hardware acceleration, is utilized.

The power consumption of an FPGA implementation highly depends on the FPGA technology. Therefore, comparing the power consumption of existing works that used different FPGA families is not fair. Hence, power analysis was not a concern in this research. However, an area-efficient architecture often results in power consumption efficiency. This conclusion is supported by the power consumption of our proposed dual-feature system as shown in Table 7.1: 78% of the power consumption is static power that is independent of the implemented circuit; while the dynamic power consumption is 132 mW.

Table 7.1: Power consumption of our proposed binary dual-feature system

| | | Power (mW) | Percentage |
|---|---|---|---|
| **Static** | | 479 | 78% |
| **Dynamic** | Clocks | 8 | 6% |
| | Signals | 28 | 21% |
| | Logic | 32 | 24% |
| | I/O | 65 | 49% |

## 7.3 Future Work

Here are some suggestions for future work of this research:

- **Preprocessing:** In this research, we did not perform preprocessing, such as data cleaning and image normalization, before input image binarization. Performing such preprocessing may result in a different binary format of images that affects the extracted feature vector, thus giving a better overall classification accuracy performance.

- **Using HOG and LBP Features with Variant Feature Vector Sizes:** In this research, LBP features are extracted from non-overlapped blocks of $3 \times 3$ grayscale pixels, and HOG features are extracted from blocks of $2 \times 2$ binary pixels with 2/3 block overlap. Reference [60] extracted two LBP feature sets and two HOG feature sets with block sizes of $3 \times 3$ and $5 \times 5$ grayscale pixels. They investigated the accuracy performance of the concatenated feature vectors including $LBP_{3\times3}$ + $LBP_{5\times5}$ and $HOG_{3\times3}$ + $HOG_{5\times5}$. They showed that concatenated feature vectors outperform single-feature vectors. One future orientation can be using this type of feature combination with *binary* feature sets.

- **Other Feature Combination Methods:** In this work, the concatenation method is used to combine features. However, there are other types of feature combination, such as Multiple Kernel Learning (MKL) and boosting [56], as reviewed in Section 2.5. Combining *binary* feature sets, proposed in this dissertation, by using other feature combination methods could be an interesting future work.

- **Using Other Feature Descriptors:** As described in Chapter 2, we utilized HOG and LBP feature descriptors because this work is the first attempt at utilizing two binarization steps in a feature descriptor. Therefore, we used algorithms with reliable accuracy performance for different applications that do not need to be specifically designed for the application. Haar-like features are popular features that have shown accuracy improvement in their combination with HOG features in classification systems. One future direction can be replacing the binary LBP feature descriptor in our binary dual-feature system with Haar-like features.

- **Using Other Classifiers:** In this research, we used the SVM classifier that has reliable accuracy performance for different applications and does not need to be designed specifically for an application. AdaBoost is a popular classifier in classification systems. As reviewed in Chapter 2, AdaBoost is a popular classifier for enhancing Haar-like feature's performance [32]. One future direction can be replacing the SVM classifier with the AdaBoost classifier, especially if Haar-like features are utilized as one of the feature sets.

- **Changing the Hardware Implementation of the SVM Classifier Module:** In the implemented five-class SVM module, SVM weights accumulation is performed in parallel for the ten binary classifiers using ten accumulators. This architecture results in high speed at the cost of greater hardware resource utilization. SVM weights accumulation can be performed using one accumulator for the ten binary classifiers, similar to our implemented LBP descriptor module, to further decrease resource utilization. However, the processing speed may decrease in such architecture.

# Appendix A

# Multi-class SVM Classifier Model

As described in Chapter 4, the five-class SVM model is generated offline for the software and hardware implementations using the function *"fitcecoc"* in MATLAB. Then weights and bias values for ten independent binary classifiers and a coding design matrix are extracted from the model to implement the prediction phase. Due to space limitations, we can not present parameter values for all studies performed. In this appendix, the SVM weights and bias values are provided for our Experiment 1 for the binary single-feature system on the Caltech-256 dataset.

Table A.1: Bias values for the binary single-feature system on Caltech-256

| **Binary classifier number** | **Bias value** |
|---|---|
| Binary classifier 1 | 0.5652729049 |
| Binary classifier 2 | 2.1601498243 |
| Binary classifier 3 | 1.5222892706 |
| Binary classifier 4 | 0.4968358890 |
| Binary classifier 5 | 0.8532886266 |
| Binary classifier 6 | 0.1614396790 |
| Binary classifier 7 | -0.0406305753 |
| Binary classifier 8 | -1.1041080079 |
| Binary classifier 9 | -2.1246997520 |
| Binary classifier 10 | -0.3780250137 |

Table A.2: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 1

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1-7 | 0.0183254483 | 0.0446911474 | -0.044915737 | 0.0154467626 | 0.0330992576 | 0.0067234646 | 0.0082645199 |
| 8-14 | 0.0154467626 | 0.0483479255 | 0.0396934104 | -0.023205874 | 0.0154467626 | 0.0017866965 | -0.008704997 |
| 9-21 | -0.002275581 | 0.0154467626 | 0.0389512646 | -0.010898816 | -0.017826561 | 0.0171004289 | -0.020608376 |
| 22-28 | 0.0058810417 | 0.0262204208 | 0.0171004289 | 0.0375980471 | -0.026330286 | 0.0015407984 | 0.0171004289 |
| 29-35 | 0.0553631217 | -0.020863847 | 0.0768305847 | 0.0171004289 | -0.021436726 | -0.033784377 | 0.0210382751 |
| 36-42 | 0.0084403652 | -0.039743071 | 0.0133756010 | 0.0414063436 | 0.0084403652 | 0.0319200798 | -0.028544785 |
| 43-49 | 0.0388449364 | 0.0084403652 | 0.0267388795 | -0.018159714 | 0.0577089325 | 0.0084403652 | -0.036661391 |
| 50-56 | 0.0040323257 | 0.0686688337 | 0.0058566400 | -0.036609343 | 0.0492647197 | 0.0053914149 | 0.0058566400 |
| 57-63 | -0.009729107 | -0.017632775 | 0.0501713873 | 0.0058566400 | -0.021400239 | -0.009471499 | -0.018702756 |
| 64-70 | 0.0058566400 | -0.008449764 | 0.0378112295 | -0.001561545 | -0.012006254 | -0.036016204 | 0.0297633729 |
| 71-77 | -0.024655219 | -0.012006254 | 0.0135192303 | 0.0476933262 | 0.0353934006 | -0.012006254 | -0.021077491 |
| 78-84 | 0.0583582609 | -0.007797441 | -0.012006254 | -0.037982926 | 0.0283905555 | -0.029304333 | -0.013802978 |
| 85-91 | 0.0374284093 | 0.0347750145 | -0.012329788 | -0.013802978 | 0.0087724392 | 0.0773361475 | -0.012983676 |
| 92-98 | -0.013802978 | -0.034516669 | 0.0497839712 | -0.016158867 | -0.013802978 | 0.0313129865 | -0.008251640 |
| 99-105 | -0.036452238 | -0.006982959 | -0.014716460 | -0.002079548 | -0.072013229 | -0.006982959 | -0.030503731 |
| 106-112 | 0.0526731640 | -0.002226971 | -0.006982959 | 0.0179773712 | -0.028551962 | -0.005879638 | -0.006982959 |
| 113-119 | 0.0174538736 | 0.0153199748 | -0.045930142 | -0.009845430 | 0.0352743666 | -0.050882469 | -0.037699834 |
| 120-126 | -0.009845430 | -0.042458824 | 0.0036354026 | 0.0129671937 | -0.009845430 | 0.0270812901 | -0.009883090 |
| 127-133 | -0.001927721 | -0.009845430 | 0.0102136024 | -0.067744807 | -0.022362911 | 0.0005519999 | 0.0254335857 |
| 134-140 | -0.061047474 | 0.0547027790 | 0.0005519999 | 0.0538833173 | 0.0377444402 | 0.0062378061 | 0.0005519999 |
| 141-147 | 0.0193421493 | -0.012607730 | -0.000670591 | 0.0005519999 | 0.0153395699 | -0.025760699 | 0.0504221467 |
| 148-154 | -0.000854123 | -0.022380100 | -0.023434074 | 0.0348483077 | -0.000854123 | -0.007378336 | -0.060997091 |
| 155-161 | 0.0231209941 | -0.000854123 | 0.0192842792 | 0.0153076462 | 0.0213681625 | -0.000854123 | -0.013120027 |
| 162-168 | -0.012466538 | 0.0592246859 | -0.001259902 | -0.002806627 | 0.0357477439 | -0.039875660 | -0.001259902 |
| 169-175 | 0.0261424564 | 0.0148464852 | 0.0442251143 | -0.001259902 | 0.0497103514 | 0.0332818835 | -0.002333098 |
| 176-182 | -0.001259902 | 0.0002801119 | 0.0322582856 | -0.058650348 | -0.009963437 | 0.0125251041 | 0.0898990469 |
| 183-189 | 0.0106436725 | -0.009963437 | 0.0261226236 | -0.000607911 | 0.0213886900 | -0.009963437 | 0.0052435547 |
| 190-196 | 0.0628425374 | 0.0132516623 | -0.009963437 | -0.042633855 | 0.0808657893 | -0.037742276 | -0.010188129 |
| 197-203 | -0.033722258 | 0.0522227944 | -0.004651398 | -0.010188129 | -0.018968478 | 0.0584237048 | -0.014290521 |
| 204-210 | -0.010188129 | 0.0047783160 | 0.0407794074 | -0.025067845 | -0.010188129 | -0.050843702 | 0.0275319744 |
| 211-217 | -0.039961475 | -0.019303624 | -0.000830376 | 0.0106097333 | -0.030672472 | -0.019303624 | -0.027291508 |
| 218-224 | 0.0191455610 | -0.040508550 | -0.019303624 | -0.028861054 | -0.011276469 | -0.010739562 | -0.019303624 |
| 225-231 | -0.088439802 | -0.038448170 | -0.039023807 | -0.023123743 | 0.0493986523 | 0.0379112385 | -0.030021615 |
| 232-238 | -0.023123743 | -0.008366466 | 0.0479754326 | -0.023139452 | -0.023123743 | -0.009823071 | 0.0167136501 |
| 239-245 | -0.017107388 | -0.023123743 | 0.0378426940 | 0.0304346273 | -0.013359622 | 0.0088342872 | 0.0040934492 |
| 246-252 | -0.026926945 | 0.0163575852 | 0.0088342872 | -0.008530758 | 0.0273093227 | -0.016639088 | 0.0088342872 |
| 253-259 | -0.050689184 | -0.030972536 | -0.007185707 | 0.0088342872 | 0.0006347828 | -0.059990760 | 0.0171151408 |
| 260-266 | -0.021581684 | -0.015951585 | -0.011104029 | 0.0228307509 | -0.021581684 | -0.079268463 | -0.029805208 |
| 267-273 | -0.045834328 | -0.021581684 | -0.005457329 | -0.016168189 | -0.001365450 | -0.021581684 | -0.025046724 |
| 274-280 | -0.010332122 | 0.0178379490 | -0.008301042 | -0.012923470 | -0.008856590 | -0.018478270 | -0.008301042 |
| 281-287 | 0.0522419872 | -0.031381790 | -0.005027433 | -0.008301042 | 0.0076452893 | 0.0389743018 | -0.029433941 |
| 288-294 | -0.008301042 | 0.0203232775 | 0.0283491873 | -0.027338973 | -0.010048084 | -0.014962945 | 0.0496983555 |
| 295-301 | -0.028143261 | -0.010048084 | -0.001853222 | 0.0499277446 | -0.020491090 | -0.010048084 | -0.043805133 |
| 302-308 | 0.0010500123 | -0.054883834 | -0.010048084 | 0.0248780787 | 0.0423955625 | -0.007688040 | 0.0174696775 |
| 309-315 | 0.0440012744 | 0.0565885585 | 0.0136520422 | 0.0174696775 | -0.027470918 | -0.020076092 | 0.0110338461 |
| 316-322 | 0.0174696775 | 0.0463940420 | -0.016695082 | -0.000103095 | 0.0174696775 | 0.0040616195 | 0.0234065117 |
| 323-329 | -0.003732955 | -0.012364287 | -0.020257502 | 0.0315162793 | -0.007669444 | -0.012364287 | -0.000242006 |
| 330-336 | -0.013707813 | -0.011063349 | -0.012364287 | -0.000344679 | -0.019186068 | 0.0237933656 | -0.012364287 |
| 337-343 | 0.0031260472 | 0.0562723479 | 0.0300169858 | 0.0036303347 | -0.014335407 | -0.004820895 | -0.033106588 |
| 344-350 | 0.0036303347 | 0.0282505530 | 0.0458572098 | -0.003885661 | 0.0036303347 | -0.018063172 | 0.0027448196 |
| 351-357 | -0.039724896 | 0.0036303347 | -0.001118377 | 0.0070010765 | 0.0078052971 | 0.0210172444 | -0.045089324 |
| 358-364 | -0.003812825 | -0.014496298 | 0.0210172444 | -0.012384127 | 0.0408107926 | 0.0246458916 | 0.0210172444 |
| 365-371 | 0.0570849309 | 0.0930913224 | 0.0304881867 | 0.0210172444 | -0.058669822 | -0.049882085 | -0.039801439 |
| 372-378 | -0.004141627 | -0.013461947 | -0.042664211 | -0.010667594 | -0.004141627 | 0.0286390542 | 0.0383954647 |
| 379-385 | 0.0024896239 | -0.004141627 | 0.0332441496 | 0.0305094889 | 0.0664079829 | -0.004141627 | -0.011509087 |
| 386-392 | -0.066947167 | -0.046611695 | -0.006610136 | -0.008821339 | -0.017819489 | -0.009637664 | -0.006610136 |

Table A.3: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 1 - continued

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 393-399 | -0.006707294 | -0.004996634 | 0.0231704465 | -0.006610136 | 0.0145257921 | 0.0017587102 | 0.0671809091 |
| 400-406 | -0.006610136 | -0.010678321 | -0.006020743 | -0.041909370 | -0.007500898 | -0.058962313 | -0.039122775 |
| 407-413 | -0.032383631 | -0.007500898 | 0.0376562228 | 0.0041401224 | 0.0269228637 | -0.007500898 | 0.0752411318 |
| 414-420 | 0.0163603920 | -0.007489023 | -0.007500898 | 0.0037428306 | 0.0074961172 | 0.0066372753 | 0.0446460876 |
| 421-427 | 0.0426726026 | -0.013726082 | -0.002441241 | 0.0446460876 | 0.0885814748 | 0.0755380539 | 0.0486398362 |
| 428-434 | 0.0446460876 | 0.0755750985 | -0.038841940 | 0.0494573219 | 0.0446460876 | 0.0112402347 | 0.0182982106 |
| 435-441 | 0.0234678231 | 0.0153222000 | -0.000711592 | 0.0281584145 | 0.0203974962 | 0.0153222000 | 0.0283447560 |
| 442-448 | -0.054848393 | -0.019165559 | 0.0153222000 | -0.011199678 | -0.009594187 | 0.0039337377 | 0.0153222000 |
| 449-455 | -0.015126002 | 0.0099399791 | -0.021679043 | -0.003031229 | -0.018670973 | 0.0059753136 | -0.044683398 |
| 456-462 | -0.003031229 | 0.0060708749 | -0.005434293 | -0.000122692 | -0.003031229 | -0.002561751 | 0.0820973998 |
| 463-469 | -0.010552977 | -0.003031229 | -0.061400563 | 0.0232514912 | -0.022059024 | -0.001318295 | 0.0558015387 |
| 470-476 | 0.0437607936 | -0.005823139 | -0.001318295 | 0.0117420098 | 0.0636621381 | -0.014286786 | -0.001318295 |
| 477-483 | -0.015461034 | 0.0169566203 | -0.044387753 | -0.001318295 | -0.020932625 | 0.0303591841 | -0.046650621 |
| 484-490 | -0.019499612 | 0.0049662401 | 0.0170019883 | 0.0160201521 | -0.019499612 | -0.038114825 | 0.0137730345 |
| 491-497 | -0.048676902 | -0.019499612 | -0.058702481 | -0.058154078 | -0.014426665 | -0.019499612 | -0.004042664 |
| 498-504 | 0.0139114440 | 0.0355973856 | -0.027911962 | -0.018867440 | 0.0057588515 | 0.0213039429 | -0.027911962 |
| 505-511 | -0.046281061 | -0.091796696 | -0.056105938 | -0.027911962 | -0.051750448 | -0.085869400 | -0.027688307 |
| 512-518 | -0.027911962 | 0.0215345151 | 0.0121499734 | 0.0262723789 | -0.000462476 | 0.0055925148 | -0.013137881 |
| 519-525 | -0.005524350 | -0.000462476 | -0.041199782 | -0.051642947 | -0.023063748 | -0.000462476 | -0.021977360 |
| 526-532 | -0.101811474 | -0.040307597 | -0.000462476 | 0.0401859600 | 0.0232170445 | -0.009836771 | 0.0086646909 |
| 533-539 | 0.0000542314 | -0.103533061 | -0.021345750 | 0.0086646909 | 0.0481616378 | -0.049419811 | -0.016507517 |
| 540-546 | 0.0086646909 | 0.0338847679 | -0.019184745 | 0.0364844726 | 0.0086646909 | 0.0269899263 | -0.067937533 |
| 547-553 | -0.010917624 | -0.014281427 | -0.019367241 | -0.030559142 | -0.018434057 | -0.014281427 | 0.0662439499 |
| 554-560 | -0.016580570 | 0.0560263146 | -0.014281427 | 0.0036781065 | -0.040656320 | -0.006499824 | -0.014281427 |
| 561-567 | 0.0272789611 | 0.0170962993 | -0.002960910 | -0.000015383 | -0.017543379 | 0.0340404193 | -0.013945402 |
| 568-574 | -0.000015383 | 0.0148848195 | 0.0150913442 | 0.0089942464 | -0.000015383 | 0.0196295473 | -0.025711151 |
| 575-581 | -0.017517073 | -0.000015383 | 0.0193017110 | 0.0562197259 | -0.011742556 | 0.0080693999 | -0.032548030 |
| 582-588 | 0.0412048026 | -0.037335142 | 0.0080693999 | 0.0240982077 | -0.004807707 | 0.0174480759 | 0.0080693999 |
| 589-595 | 0.0237393958 | 0.0127432156 | 0.0054277546 | 0.0080693999 | -0.048795600 | 0.0264570514 | -0.056752147 |
| 596-602 | -0.001443340 | 0.0340597456 | 0.0218306514 | 0.0443570133 | -0.001443340 | 0.0019843591 | 0.0089413866 |
| 603-609 | 0.0131638948 | -0.001443340 | -0.044040160 | -0.000172416 | 0.0454242322 | -0.001443340 | -0.015038273 |
| 610-616 | -0.002104051 | 0.0185059915 | -0.000974643 | -0.012561610 | -0.073410503 | 0.0347454975 | -0.000974643 |
| 617-623 | -0.043385344 | -0.025123227 | 0.0191070491 | -0.000974643 | -0.039097401 | -0.025716739 | 0.0071372770 |
| 624-630 | -0.000974643 | -0.019662719 | -0.037159150 | 0.0245908602 | -0.006387663 | 0.0156253758 | -0.102275237 |
| 631-637 | -0.020815246 | -0.006387663 | -0.007315434 | -0.051763189 | -0.020159215 | -0.006387663 | 0.0060971167 |
| 638-644 | 0.0033596637 | -0.011556083 | -0.006387663 | 0.0501386805 | -0.050564803 | -0.013255526 | 0.0172999508 |
| 645-651 | 0.0775690263 | -0.011361629 | 0.0636090017 | 0.0172999508 | 0.0046153033 | 0.0115525275 | -0.035610658 |
| 652-658 | 0.0172999508 | 0.0006313418 | 0.0098776887 | -0.007471669 | 0.0172999508 | 0.0401003477 | -0.003614254 |
| 659-665 | 0.0520250621 | -0.014774630 | -0.010552319 | -0.059305479 | -0.028844810 | -0.014774630 | 0.0244917956 |
| 666-672 | 0.0312372737 | 0.0177081996 | -0.014774630 | -0.047586344 | 0.0177998791 | 0.0400244487 | -0.014774630 |
| 673-679 | -0.000939812 | 0.0092823729 | 0.0432863249 | 0.0038893710 | 0.0135503573 | -0.048840236 | 0.0095515244 |
| 680-688 | 0.0038893710 | 0.0303669301 | -0.083575637 | 0.0153031306 | 0.0038893710 | 0.0680245917 | -0.000971377 |
| 687-693 | 0.0251147294 | 0.0038893710 | 0.0114005465 | 0.0018718462 | 0.0529944521 | 0.0251746554 | 0.0059961219 |
| 694-700 | 0.0143184472 | 0.0108508639 | 0.0251746554 | 0.0978717689 | -0.003963580 | 0.0212838551 | 0.0251746554 |
| 701-707 | -0.003406576 | 0.0687454899 | -0.007024524 | 0.0251746554 | 0.0095190946 | 0.0292350456 | 0.0079022767 |
| 708-714 | 0.0219617243 | -0.017110688 | 0.0216180020 | 0.0897870749 | 0.0219617243 | -0.006959912 | 0.0444274199 |
| 715-721 | -0.036268489 | 0.0219617243 | -0.018857243 | 0.0049722868 | -0.026081577 | 0.0219617243 | -0.015196651 |
| 722-728 | -0.023351006 | 0.0476832668 | 0.0329526193 | 0.0340838006 | -0.003751909 | 0.0561648168 | 0.0329526193 |
| 729-735 | -0.017084066 | 0.0399794784 | -0.025037188 | 0.0329526193 | 0.0209934013 | 0.0463068860 | 0.0384607586 |
| 736-742 | 0.0329526193 | -0.005019029 | -0.039317308 | 0.0013938226 | -0.006578248 | -0.007998622 | -0.039450369 |
| 743-749 | 0.0108908797 | -0.006578248 | -0.017393617 | -0.017185631 | -0.018595334 | -0.006578248 | 0.0084604567 |
| 750-756 | -0.017091387 | 0.0224377992 | -0.006578248 | 0.0147458455 | -0.017662536 | -0.039695287 | -0.005342688 |
| 757-763 | 0.0187109362 | 0.0186956199 | 0.0021717047 | -0.005342688 | 0.0093498061 | -0.027057749 | 0.0039855139 |
| 764-770 | -0.005342688 | 0.0356269078 | -0.021010351 | 0.0107346099 | -0.005342688 | -0.031137265 | 0.0248154736 |
| 771-777 | -0.063493001 | -0.004732011 | -0.016758549 | -0.005575441 | 0.0283138902 | -0.004732011 | 0.0238256775 |
| 778-784 | -0.026762166 | -0.019371833 | -0.004732011 | 0.0415687747 | 0.0103647852 | -0.010963351 | -0.004732011 |

Table A.4: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 2

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1-7 | 0.0219365371 | 0.0225896352 | -0.022178865 | -0.002620297 | -0.033620202 | 0.0540808721 | 0.0503322114 |
| 8-14 | -0.002620297 | 0.0130189982 | 0.0348809072 | -0.059868789 | -0.002620297 | -0.020614971 | 0.0213456657 |
| 9-21 | 0.0034572361 | -0.002620297 | -0.000097506 | 0.0222731803 | 0.0740163708 | -0.014121539 | 0.0023378578 |
| 22-28 | -0.017995550 | -0.028041498 | -0.014121539 | -0.025803609 | 0.0434262178 | 0.0244863524 | -0.014121539 |
| 29-35 | 0.0378821925 | -0.039434085 | 0.0149285711 | -0.014121539 | 0.0168927633 | -0.015145518 | -0.006804377 |
| 36-42 | 0.0089584392 | -0.015365387 | 0.0299586147 | -0.018727442 | 0.0089584392 | 0.0560930921 | -0.022943054 |
| 43-49 | 0.0418625593 | 0.0089584392 | 0.0136111243 | -0.027902720 | -0.022939756 | 0.0089584392 | 0.0066450053 |
| 50-56 | 0.0538587306 | 0.0048132476 | 0.0205156391 | 0.0374720231 | 0.0364454384 | 0.0070692169 | 0.0205156391 |
| 57-63 | 0.0188072254 | 0.0110113565 | 0.0082911526 | 0.0205156391 | -0.017882621 | 0.0037611610 | 0.0221928199 |
| 64-70 | 0.0205156391 | 0.0113074154 | 0.0160931473 | 0.0072121806 | -0.003048497 | 0.0373336830 | -0.020276553 |
| 71-77 | 0.0019871781 | -0.003048497 | -0.043623992 | -0.004892978 | -0.021870639 | -0.003048497 | -0.004415018 |
| 78-84 | 0.0149982014 | -0.017895935 | -0.003048497 | 0.0252698735 | -0.030651765 | -0.021123392 | -0.019346661 |
| 85-91 | 0.0173191823 | -0.009335128 | -0.039544012 | -0.019346661 | -0.005826166 | 0.0217594962 | -0.008544913 |
| 92-98 | -0.019346661 | -0.001845780 | 0.0141259041 | 0.0090111289 | -0.019346661 | 0.0212228273 | 0.0147461213 |
| 99-105 | -0.037816180 | 0.0050143412 | 0.0924983045 | 0.0763478338 | -0.003344354 | 0.0050143412 | 0.0090957310 |
| 106-112 | 0.0244579573 | 0.0367404284 | 0.0050143412 | -0.003315537 | 0.0455737969 | -0.029061324 | 0.0050143412 |
| 113-119 | 0.0087536821 | 0.0376211067 | -0.056148384 | -0.023180138 | -0.041065010 | 0.0212235389 | 0.0304129959 |
| 120-126 | -0.023180138 | 0.0624864109 | 0.0157605795 | -0.020406412 | -0.023180138 | -0.011866678 | -0.000737250 |
| 127-133 | 0.0226498295 | -0.023180138 | -0.024128815 | 0.0069637548 | 0.0109886576 | -0.018591056 | -0.004490536 |
| 134-140 | 0.0117206487 | 0.0211198938 | -0.018591056 | -0.001656251 | 0.0354764085 | 0.0189335053 | -0.018591056 |
| 141-147 | -0.010073960 | -0.053376276 | -0.073872648 | -0.018591056 | -0.017672520 | -0.015459112 | -0.011928922 |
| 148-154 | -0.035944948 | 0.0054598620 | -0.010094995 | 0.0323950638 | -0.035944948 | -0.007758362 | -0.044015028 |
| 155-161 | -0.083372063 | -0.035944948 | -0.018599583 | -0.105560186 | -0.024083476 | -0.035944948 | 0.0363118531 |
| 162-168 | -0.003348600 | -0.017944953 | -0.000647837 | -0.021160179 | -0.002309845 | 0.0152677644 | -0.000647837 |
| 169-175 | 0.0180872516 | -0.084810615 | -0.003067724 | -0.000647837 | -0.005002847 | 0.0031985087 | -0.033304367 |
| 176-182 | -0.000647837 | -0.009938966 | 0.0225629245 | 0.0014778236 | 0.0115348564 | 0.0059634828 | -0.001684355 |
| 183-189 | 0.0421082609 | 0.0115348564 | -0.049080130 | -0.003490641 | -0.023262089 | 0.0115348564 | -0.029734300 |
| 190-196 | -0.035808222 | 0.0341662751 | 0.0115348564 | -0.028573689 | -0.013373986 | 0.0000038874 | -0.032041370 |
| 197-203 | -0.025383466 | 0.0063247449 | -0.018720676 | -0.032041370 | -0.022812655 | 0.0079990436 | 0.0122767804 |
| 204-210 | -0.032041370 | -0.019631470 | -0.042000290 | -0.010050314 | -0.032041370 | -0.014185030 | 0.0119165545 |
| 211-217 | 0.0177258631 | -0.011580627 | 0.0029651503 | 0.0175787012 | -0.045656294 | -0.011580627 | -0.008436714 |
| 218-224 | 0.0068588459 | 0.0000719381 | -0.011580627 | 0.0701050345 | 0.0842469975 | -0.027221562 | -0.011580627 |
| 225-231 | 0.0497409567 | 0.0407594265 | 0.0173068967 | -0.000386637 | 0.0013109436 | 0.0010687276 | 0.0167642846 |
| 232-238 | -0.000386637 | 0.0524099331 | 0.0671692422 | -0.035161789 | -0.000386637 | -0.017767644 | -0.003308740 |
| 239-245 | 0.0154921016 | -0.000386637 | 0.0211254846 | 0.0662636926 | 0.0488963086 | 0.0098216769 | -0.035377079 |
| 246-252 | -0.041496179 | -0.084124171 | 0.0098216769 | -0.073126145 | -0.022881646 | -0.002026213 | 0.0098216769 |
| 253-259 | -0.051021563 | -0.002006105 | 0.0156485377 | 0.0098216769 | -0.002274328 | -0.002285739 | -0.012482249 |
| 260-266 | 0.0135890498 | 0.0242623295 | -0.037866884 | 0.0523861293 | 0.0135890498 | -0.036166894 | 0.0024608281 |
| 267-273 | 0.0227274272 | 0.0135890498 | -0.024145168 | -0.007024001 | -0.036975499 | 0.0135890498 | -0.000952436 |
| 274-280 | -0.041028991 | 0.0589379430 | 0.0176358342 | -0.023365281 | -0.002959329 | -0.022316400 | 0.0176358342 |
| 281-287 | 0.0110914230 | 0.0115203910 | -0.041938486 | 0.0176358342 | 0.0237916706 | -0.050611450 | 0.0462492579 |
| 288-294 | 0.0176358342 | -0.056596243 | -0.008332555 | -0.016568105 | -0.009605047 | 0.0243214412 | 0.0008450353 |
| 295-301 | 0.0053229947 | -0.009605047 | -0.004476331 | -0.086191852 | 0.0295450876 | -0.009605047 | 0.0135571077 |
| 302-308 | 0.0120196265 | 0.0112177594 | -0.009605047 | 0.0072801773 | -0.014989108 | -0.019131695 | 0.0123333660 |
| 309-315 | -0.020112499 | 0.0101135989 | -0.003177461 | 0.0123333660 | -0.022076567 | -0.027557783 | 0.0037242691 |
| 316-322 | 0.0123333660 | -0.038912257 | 0.0083270661 | 0.0059928971 | 0.0123333660 | -0.035036566 | -0.023291359 |
| 323-329 | -0.005925520 | -0.008149301 | 0.0597934642 | 0.0416549547 | -0.037574859 | -0.008149301 | -0.056434499 |
| 330-336 | 0.0030662393 | -0.012413953 | -0.008149301 | 0.0335139602 | 0.0391893142 | -0.028412530 | -0.008149301 |
| 337-343 | 0.0785868069 | 0.0841567974 | 0.0270057034 | 0.0396992481 | -0.001973848 | -0.000439827 | 0.0180070711 |
| 344-350 | 0.0396992481 | 0.1149270491 | 0.1220947805 | -0.006988631 | 0.0396992481 | -0.016929382 | -0.047791893 |
| 351-357 | 0.0175587349 | 0.0396992481 | 0.0077556695 | -0.017527127 | 0.0391759898 | 0.0026865333 | -0.048032921 |
| 358-364 | -0.030139247 | -0.052537130 | 0.0026865333 | 0.0117501097 | -0.040607145 | -0.018022136 | 0.0026865333 |
| 365-371 | -0.048338209 | -0.031811791 | 0.0328343910 | 0.0026865333 | -0.038806889 | -0.001916038 | -0.028262798 |
| 372-378 | 0.0034214996 | -0.052531018 | 0.0253610517 | -0.006089465 | 0.0034214996 | 0.0130332328 | 0.0118179045 |
| 379-385 | 0.0297098496 | 0.0034214996 | -0.002848899 | 0.0318950596 | -0.026384118 | 0.0034214996 | -0.038874027 |
| 386-392 | -0.016275648 | -0.025287031 | 0.0059988926 | -0.070532498 | -0.031379982 | 0.0569675669 | 0.0059988926 |

Table A.5: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 2 - continued

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 393-399 | 0.0233811277 | 0.0025936102 | 0.0317180647 | 0.0059988926 | -0.039972528 | -0.029001268 | 0.0045913413 |
| 400-406 | 0.0059988926 | -0.034409762 | -0.045257410 | 0.0319467062 | 0.0079559676 | -0.031494172 | -0.023137453 |
| 407-413 | 0.0342305212 | 0.0079559676 | -0.020972121 | -0.075463754 | -0.031136662 | 0.0079559676 | 0.0188317460 |
| 414-420 | -0.011877903 | -0.029472155 | 0.0079559676 | -0.036868568 | -0.006512115 | 0.0180878577 | 0.0165684961 |
| 421-427 | -0.023289476 | 0.0028094368 | 0.0281572046 | 0.0165684961 | -0.018565730 | -0.027113840 | 0.0172528177 |
| 428-434 | 0.0165684961 | -0.022143006 | -0.018383998 | 0.0109776921 | 0.0165684961 | -0.031965904 | -0.010322596 |
| 435-441 | 0.0083260170 | -0.005268038 | 0.0262198886 | 0.0376347699 | -0.033050869 | -0.005268038 | -0.027766038 |
| 442-448 | -0.029179994 | -0.020736303 | -0.005268038 | 0.0301248283 | 0.0656116837 | -0.006550118 | -0.005268038 |
| 449-455 | 0.1019245038 | 0.1021781618 | 0.0328884420 | 0.0195699078 | -0.008416125 | -0.053383189 | -0.024892237 |
| 456-462 | 0.0195699078 | -0.000303747 | 0.0315527556 | -0.052146481 | 0.0195699078 | 0.0143716006 | 0.0304369534 |
| 463-469 | 0.0223913372 | 0.0195699078 | -0.007764144 | -0.037457252 | -0.043720024 | -0.019031155 | -0.001708484 |
| 470-476 | -0.026912767 | 0.0159605729 | -0.019031155 | -0.006767216 | 0.0284495034 | -0.070845455 | -0.019031155 |
| 477-483 | -0.000293311 | -0.043065140 | -0.039800986 | -0.019031155 | -0.026540023 | -0.016463056 | 0.0215281334 |
| 484-490 | 0.0061959841 | 0.0080400975 | -0.005930761 | -0.064766075 | 0.0061959841 | 0.0276145325 | -0.008230977 |
| 491-497 | 0.0231023367 | 0.0061959841 | -0.012686567 | 0.0117000829 | -0.003079008 | 0.0061959841 | 0.0481032121 |
| 498-504 | -0.018110338 | -0.047835643 | 0.0015584672 | -0.042116881 | -0.022539950 | -0.016094814 | 0.0015584672 |
| 505-511 | -0.016494456 | -0.007556217 | -0.021858978 | 0.0015584672 | -0.051132877 | -0.001153207 | 0.0039376076 |
| 512-518 | 0.0015584672 | 0.0016220618 | -0.093111913 | -0.070871147 | 0.0077859841 | -0.022543869 | -0.068337793 |
| 519-525 | 0.0176242136 | 0.0077859841 | -0.028347602 | -0.034408069 | -0.022772997 | 0.0077859841 | -0.043455073 |
| 526-532 | -0.036320715 | 0.0017441353 | 0.0077859841 | -0.020657022 | -0.056781584 | 0.0099848633 | 0.0224398527 |
| 533-539 | -0.079956058 | -0.057667639 | 0.0133865841 | 0.0224398527 | 0.0231718145 | -0.005572011 | -0.008304523 |
| 540-546 | 0.0224398527 | -0.038082413 | -0.024081638 | -0.085362554 | 0.0224398527 | -0.043442224 | -0.045886478 |
| 547-553 | 0.0027104679 | 0.0160032769 | 0.0890904903 | 0.0666900493 | 0.0147211959 | 0.0160032769 | 0.0606763589 |
| 554-560 | -0.037834245 | -0.038216955 | 0.0160032769 | 0.0403090010 | 0.0238400397 | -0.008958695 | 0.0160032769 |
| 561-567 | 0.0291470809 | 0.0553941159 | -0.071945680 | 0.0452155301 | -0.030898551 | 0.0022457897 | 0.0251978982 |
| 568-574 | 0.0452155301 | 0.0551180739 | 0.0759456685 | 0.0080815577 | 0.0452155301 | 0.0040104977 | 0.0468362538 |
| 575-581 | 0.0335765797 | 0.0452155301 | -0.041502976 | 0.0233092295 | -0.004350458 | -0.001204437 | 0.0025777953 |
| 582-588 | -0.027419181 | -0.027028127 | -0.001204437 | -0.025708625 | 0.0206795091 | -0.000531684 | -0.001204437 |
| 589-595 | -0.005198011 | 0.0144761792 | 0.0405890137 | -0.001204437 | 0.0205608567 | -0.027676988 | -0.024423731 |
| 596-602 | 0.0046608345 | -0.036485134 | -0.020129675 | -0.020109020 | 0.0046608345 | -0.011427011 | 0.0240149129 |
| 603-609 | 0.0370595036 | 0.0046608345 | -0.014213642 | 0.0733696319 | 0.0950108219 | 0.0046608345 | -0.050216236 |
| 610-616 | 0.0051554987 | -0.036923949 | 0.0012309392 | -0.004047051 | -0.065028083 | -0.021628209 | 0.0012309392 |
| 617-623 | -0.049748657 | 0.0524007791 | 0.0873612672 | 0.0012309392 | -0.027641107 | 0.0247807482 | 0.0452629521 |
| 624-630 | 0.0012309392 | 0.0032586267 | -0.072204642 | -0.018049440 | 0.0211051476 | -0.035569547 | -0.060713516 |
| 631-637 | 0.0004574500 | 0.0211051476 | 0.0017960713 | 0.0241144332 | 0.0543422956 | 0.0211051476 | -0.032098845 |
| 638-644 | 0.0335866452 | -0.024670246 | 0.0211051476 | 0.0205184323 | -0.045101011 | 0.0278786612 | 0.0348073915 |
| 645-651 | 0.0352412100 | -0.037338311 | -0.088571874 | 0.0348073915 | -0.032725857 | 0.0921670636 | 0.0202812789 |
| 652-658 | 0.0348073915 | 0.0011741421 | -0.068843514 | 0.0330011569 | 0.0348073915 | 0.0325574985 | -0.016669981 |
| 659-665 | -0.057240895 | 0.0132698111 | 0.0434046010 | 0.0334827430 | -0.014121441 | 0.0132698111 | 0.0143287728 |
| 666-672 | -0.081476342 | -0.002540130 | 0.0132698111 | 0.0869146721 | 0.0447610773 | -0.043041222 | 0.0132698111 |
| 673-679 | 0.0477639559 | 0.0375485542 | -0.045141000 | 0.0113311382 | -0.022601962 | 0.0072811428 | 0.0001971268 |
| 680-688 | 0.0113311382 | 0.0021781741 | 0.0066079152 | 0.0029796173 | 0.0113311382 | 0.0153966799 | 0.0067047223 |
| 687-693 | 0.0543464267 | 0.0113311382 | -0.053827845 | -0.002144571 | -0.002224453 | -0.011604241 | 0.0061650682 |
| 694-700 | -0.038436329 | 0.0227265270 | -0.011604241 | -0.007538699 | 0.0139076214 | 0.0139146052 | -0.011604241 |
| 701-707 | -0.013334364 | 0.0192302407 | 0.0432102096 | -0.011604241 | 0.0012908209 | -0.044458567 | 0.0087728040 |
| 708-714 | -0.015348052 | -0.028465725 | 0.0084694258 | 0.0246026429 | -0.015348052 | -0.004669647 | -0.000061171 |
| 715-721 | 0.0005373776 | -0.015348052 | -0.019617256 | -0.004762341 | -0.005852616 | -0.015348052 | -0.029749514 |
| 722-728 | 0.0314752403 | 0.0102406787 | -0.015348052 | -0.037956147 | 0.0127874831 | 0.0732526643 | -0.015348052 |
| 729-735 | -0.019617256 | -0.006340958 | -0.011789170 | -0.015348052 | -0.020810572 | -0.033555570 | -0.009856816 |
| 736-742 | -0.015348052 | -0.007488226 | 0.0454094624 | 0.0615353012 | -0.008872742 | -0.052037324 | 0.0317210810 |
| 743-749 | -0.010713417 | -0.008872742 | -0.014335263 | -0.028466898 | 0.0220205783 | -0.008872742 | -0.003359103 |
| 750-756 | 0.0381217500 | -0.008015657 | -0.008872742 | -0.037291309 | 0.0747847076 | -0.017862293 | -0.015246824 |
| 757-763 | -0.010912694 | -0.084508605 | -0.009590876 | -0.015246824 | -0.009733184 | 0.0482642219 | 0.0247010568 |
| 764-770 | -0.015246824 | -0.033957535 | -0.054762342 | -0.001145417 | -0.015246824 | -0.025027458 | -0.065061069 |
| 771-777 | -0.003444794 | 0.0115339000 | 0.0724465326 | 0.0388785347 | -0.017216631 | 0.0115339000 | 0.0140982187 |
| 778-784 | -0.021403428 | 0.0205589440 | 0.0115339000 | 0.0055417702 | 0.0498175993 | -0.014606492 | 0.0115339000 |

Table A.6: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 3

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1-7 | 0.0071876291 | -0.030949667 | -0.072798466 | -0.003910581 | 0.1221494285 | 0.0769998710 | 0.1412559551 |
| 8-14 | -0.003910581 | -0.057636839 | -0.007389887 | -0.110744773 | -0.003910581 | 0.0258101325 | -0.103828387 |
| 9-21 | 0.0105968341 | -0.003910581 | 0.1399370293 | 0.0261973927 | 0.0314594604 | -0.024834051 | 0.0100218055 |
| 22-28 | 0.0116915871 | -0.026693634 | -0.024834051 | -0.044002392 | 0.0102076534 | 0.0070126706 | -0.024834051 |
| 29-35 | 0.0506702043 | -0.124099060 | 0.0163688218 | -0.024834051 | 0.0638993163 | -0.035808012 | 0.0468521437 |
| 36-42 | 0.0240530822 | -0.132405722 | -0.018744733 | -0.036153408 | 0.0240530822 | 0.0029345730 | -0.100167330 |
| 43-49 | 0.1276147700 | 0.0240530822 | -0.023443394 | -0.015490650 | -0.083547234 | 0.0240530822 | -0.190576143 |
| 50-56 | 0.0774895381 | 0.0130273850 | 0.0296108920 | -0.162456540 | 0.0878201511 | -0.008112824 | 0.0296108920 |
| 57-63 | -0.038101285 | 0.0029277481 | 0.0142065658 | 0.0296108920 | -0.101486422 | 0.0860573806 | 0.0354300065 |
| 64-70 | 0.0296108920 | -0.189509685 | 0.0013773649 | -0.067717728 | -0.018139494 | 0.0956475320 | 0.1634060469 |
| 71-77 | 0.0146698394 | -0.018139494 | 0.0330621393 | 0.0813130538 | 0.0210078038 | -0.018139494 | 0.0026386951 |
| 78-84 | 0.0141245440 | 0.0073366727 | -0.018139494 | -0.082314682 | -0.012049100 | -0.049067275 | -0.121922285 |
| 85-91 | 0.0646269761 | 0.0420192880 | 0.0184245739 | -0.121922285 | -0.098440484 | -0.018107591 | -0.159117432 |
| 92-98 | -0.121922285 | -0.111964224 | -0.043132900 | -0.062071678 | -0.121922285 | 0.1061809345 | 0.0179170451 |
| 99-105 | 0.0633303265 | -0.025807482 | 0.0833333948 | 0.1026119785 | 0.0001278654 | -0.025807482 | 0.0552859282 |
| 106-112 | 0.0734354133 | 0.0717875278 | -0.025807482 | -0.126204253 | 0.1029089372 | 0.0389322640 | -0.025807482 |
| 113-119 | -0.031716672 | -0.077656269 | -0.122214229 | -0.036679353 | 0.0487970035 | 0.0343548478 | 0.0808668343 |
| 120-126 | -0.036679353 | -0.085836523 | -0.042745818 | -0.083757507 | -0.036679353 | 0.0281887895 | -0.048405450 |
| 127-133 | -0.040940863 | -0.036679353 | 0.0092376610 | -0.001813759 | 0.0447984825 | -0.034251551 | -0.050690016 |
| 134-140 | -0.057047311 | 0.0531981947 | -0.034251551 | -0.033781522 | 0.1091711972 | 0.0235813503 | -0.034251551 |
| 141-147 | 0.0696691977 | -0.045081774 | 0.0462982249 | -0.034251551 | -0.001442408 | -0.024865171 | -0.003470466 |
| 148-154 | -0.028926027 | -0.126859266 | -0.042198405 | 0.0794252289 | -0.028926027 | 0.0087072411 | -0.121249772 |
| 155-161 | 0.0645014399 | -0.028926027 | 0.0114788065 | -0.140513497 | -0.016312173 | -0.028926027 | -0.032344130 |
| 162-168 | 0.0258403802 | 0.0492820091 | -0.027082616 | -0.080883246 | 0.0770533248 | -0.042828508 | -0.027082616 |
| 169-175 | 0.1040771356 | -0.055618751 | 0.0782791664 | -0.027082616 | 0.0354503224 | 0.0469864741 | 0.0527912494 |
| 176-182 | -0.027082616 | -0.014185535 | 0.0571095775 | 0.0047833831 | -0.042095657 | -0.059100953 | 0.0462505588 |
| 183-189 | 0.0343160412 | -0.042095657 | -0.012611593 | 0.0426372467 | 0.0093871100 | -0.042095657 | -0.034086331 |
| 190-196 | -0.104046692 | -0.021034067 | -0.042095657 | -0.068932321 | 0.0927676654 | -0.003058226 | 0.0081081940 |
| 197-203 | -0.113066554 | 0.1651415004 | 0.0212764251 | 0.0081081940 | 0.0587364101 | 0.0691094440 | -0.049577472 |
| 204-210 | 0.0081081940 | 0.1002005150 | 0.0158263044 | 0.0203689527 | 0.0081081940 | -0.026245920 | 0.0521551437 |
| 211-217 | 0.0738148548 | 0.0499566069 | -0.075315791 | 0.0932591448 | 0.1325520566 | 0.0499566069 | -0.037298101 |
| 218-224 | -0.025282186 | -0.009152760 | 0.0499566069 | -0.041029812 | 0.0403680078 | 0.0713344620 | 0.0499566069 |
| 225-231 | -0.074587906 | -0.035283650 | -0.042287966 | -0.008403760 | 0.0260517829 | 0.1279369854 | -0.049072623 |
| 232-238 | -0.008403760 | 0.0146380996 | -0.013401800 | -0.140603905 | -0.008403760 | -0.026440669 | 0.1537680295 |
| 239-245 | 0.0435552708 | -0.008403760 | 0.0236959680 | 0.1225614686 | -0.072668072 | -0.044113583 | 0.0143330417 |
| 246-252 | -0.060262806 | 0.0503419260 | -0.044113583 | 0.0369774229 | 0.0179205922 | -0.097020350 | -0.044113583 |
| 253-259 | 0.0109680095 | 0.1453393210 | 0.0066184454 | -0.044113583 | 0.1154738078 | -0.120933578 | 0.0427248319 |
| 260-266 | -0.012525786 | 0.0141174884 | -0.154609578 | -0.034650677 | -0.012525786 | -0.100544046 | -0.117305985 |
| 267-273 | -0.000288779 | -0.012525786 | -0.054140812 | 0.0005564446 | 0.0366280958 | -0.012525786 | -0.032630326 |
| 274-280 | -0.067424139 | 0.0910346687 | -0.008534450 | -0.126456138 | -0.024457838 | 0.0015614950 | -0.008534450 |
| 281-287 | 0.0878964447 | 0.0182174651 | 0.0255666333 | -0.008534450 | -0.063956717 | -0.074790690 | -0.061825991 |
| 288-294 | -0.008534450 | -0.044521973 | 0.0612839991 | -0.035898948 | -0.010893944 | 0.0588074734 | 0.0982999549 |
| 295-301 | -0.046603091 | -0.010893944 | -0.000553739 | -0.066039320 | 0.0124841858 | -0.010893944 | -0.021924999 |
| 302-308 | 0.1943857262 | -0.021427476 | -0.010893944 | 0.0235719350 | 0.0044271447 | 0.0022263041 | 0.0066415734 |
| 309-315 | -0.005739349 | -0.045595368 | 0.0372196296 | 0.0066415734 | -0.029056152 | 0.1159528359 | -0.057119939 |
| 316-322 | 0.0066415734 | 0.0304737862 | -0.020991967 | -0.081843748 | 0.0066415734 | -0.013266401 | -0.028775061 |
| 323-329 | -0.099739896 | -0.046029034 | -0.108195310 | -0.016599249 | -0.001548447 | -0.046029034 | 0.0411277623 |
| 330-336 | 0.0069890271 | -0.175563272 | -0.046029034 | 0.0039495764 | -0.001484132 | 0.1315202514 | -0.046029034 |
| 337-343 | 0.0251629635 | 0.0684101129 | -0.015451710 | 0.0194257687 | 0.0715631256 | 0.0575486084 | 0.0130083335 |
| 344-350 | 0.0194257687 | 0.1768240245 | 0.0756923035 | -0.063867845 | 0.0194257687 | -0.064360140 | -0.000838785 |
| 351-357 | -0.010966395 | 0.0194257687 | 0.0096924369 | -0.101628771 | -0.043334229 | -0.024020533 | 0.0291736354 |
| 358-364 | -0.037065024 | -0.112523113 | -0.024020533 | 0.1213946756 | 0.0520134159 | -0.055596157 | -0.024020533 |
| 365-371 | -0.016781838 | 0.0611248534 | 0.0969530461 | -0.024020533 | -0.030657549 | 0.0309363878 | -0.070256733 |
| 372-378 | 0.0284078807 | -0.088725310 | 0.0492652005 | 0.0019771845 | 0.0284078807 | 0.0704483598 | 0.0846158191 |
| 379-385 | 0.0972376423 | 0.0284078807 | 0.0515665252 | -0.032419745 | 0.0550232756 | 0.0284078807 | 0.0142058844 |
| 386-392 | -0.046906426 | 0.0412719492 | -0.040463196 | -0.048849402 | 0.0026111437 | -0.105940800 | -0.040463196 |

Table A.7: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 3 - continued

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 393-399 | 0.0035471327 | -0.003841004 | -0.088467516 | -0.040463196 | 0.0075900099 | -0.076762142 | -0.026184091 |
| 400-406 | -0.040463196 | -0.058098574 | -0.042664215 | -0.005161650 | -0.002948542 | -0.097503493 | -0.003668579 |
| 407-413 | -0.103182635 | -0.002948542 | 0.0476997296 | -0.073856016 | -0.010936283 | -0.002948542 | -0.106832308 |
| 414-420 | -0.123885657 | 0.0748798128 | -0.002948542 | -0.067836876 | 0.0489949763 | -0.057487513 | -0.032178972 |
| 421-427 | -0.004190189 | -0.018885219 | -0.043146683 | -0.032178972 | -0.121054899 | -0.085751388 | 0.1147258027 |
| 428-434 | -0.032178972 | 0.0562157410 | -0.076372180 | 0.0226907818 | -0.032178972 | 0.0229376211 | 0.0037867791 |
| 435-441 | -0.007916960 | 0.0052775077 | -0.036660881 | -0.009793426 | 0.0944450669 | 0.0052775077 | -0.016135686 |
| 442-448 | -0.074278088 | 0.0201297188 | 0.0052775077 | 0.0064660336 | -0.008131235 | 0.0007323434 | 0.0052775077 |
| 449-455 | 0.0998901266 | 0.1490599926 | 0.0391058550 | -0.047383075 | -0.047260766 | 0.0905474702 | -0.070385187 |
| 456-462 | -0.047383075 | -0.143337873 | -0.044182061 | -0.047901710 | -0.047383075 | 0.0425591721 | 0.0364790537 |
| 463-469 | -0.062745232 | -0.047383075 | 0.0400408418 | -0.026839527 | -0.050312597 | 0.0304078557 | 0.0234360156 |
| 470-476 | -0.132240660 | 0.0368602106 | 0.0304078557 | 0.0996238804 | 0.1506429924 | -0.017077182 | 0.0304078557 |
| 477-483 | 0.0706638886 | 0.0573874986 | 0.0097350485 | 0.0304078557 | -0.092639691 | -0.025305268 | -0.033314063 |
| 484-490 | 0.0221291216 | 0.0282089875 | -0.037335461 | -0.086472611 | 0.0221291216 | 0.0028930065 | 0.0021390082 |
| 491-497 | 0.0973377430 | 0.0221291216 | -0.017856391 | 0.0432201759 | 0.0351482024 | 0.0221291216 | 0.1416710700 |
| 498-504 | -0.097102115 | -0.100175723 | -0.008034313 | 0.0167893079 | -0.067036097 | 0.0268557892 | -0.008034313 |
| 505-511 | -0.015999814 | -0.032889518 | -0.033671709 | -0.008034313 | -0.120693393 | 0.0272614681 | 0.0040654611 |
| 512-518 | -0.008034313 | 0.0380303223 | -0.116051917 | -0.050422491 | 0.0107360321 | 0.0316495996 | -0.133884852 |
| 519-525 | -0.023106658 | 0.0107360321 | -0.139371588 | 0.0765563604 | 0.0511353847 | 0.0107360321 | -0.021512336 |
| 526-532 | -0.071073321 | 0.0132706410 | 0.0107360321 | 0.1087236828 | -0.042508557 | 0.0638789379 | 0.0420238599 |
| 533-539 | 0.1406494685 | -0.186894317 | 0.0435516465 | 0.0420238599 | 0.0668471741 | -0.038516904 | 0.0465098691 |
| 540-546 | 0.0420238599 | 0.0669247796 | -0.006124615 | 0.0184968312 | 0.0420238599 | 0.0347267327 | -0.115799963 |
| 547-553 | 0.0221313499 | -0.017513797 | 0.0552079790 | -0.028503865 | -0.050039142 | -0.017513797 | 0.0973367995 |
| 554-560 | -0.096180177 | 0.0144134831 | -0.017513797 | 0.0672928572 | 0.0049578857 | 0.0131190392 | -0.017513797 |
| 561-567 | -0.119409922 | 0.0731586356 | -0.181408722 | -0.018091240 | -0.113471377 | -0.038375273 | -0.135012262 |
| 568-574 | -0.018091240 | 0.0070966826 | 0.1237266208 | 0.0864652115 | -0.018091240 | 0.0160652132 | 0.0368552155 |
| 575-581 | -0.008095634 | -0.018091240 | 0.0841902212 | 0.0263463147 | -0.043679433 | 0.0153399350 | -0.003132597 |
| 582-588 | -0.123083207 | 0.0508175526 | 0.0153399350 | 0.0277140768 | 0.0934955586 | 0.0601086873 | 0.0153399350 |
| 589-595 | -0.008763420 | 0.0957518541 | 0.0735814540 | 0.0153399350 | -0.024972244 | -0.006056837 | 0.0012795730 |
| 596-602 | 0.0100432428 | -0.031494266 | -0.091110494 | 0.0089881456 | 0.0100432428 | 0.0288160193 | 0.1046247827 |
| 603-609 | 0.1494372823 | 0.0100432428 | -0.130658150 | 0.0480778989 | 0.1219356282 | 0.0100432428 | -0.023201688 |
| 610-616 | 0.0413590258 | -0.068253626 | 0.0279191163 | 0.0473464507 | -0.012890974 | 0.0123974990 | 0.0279191163 |
| 617-623 | -0.069718126 | -0.001779517 | 0.0304502481 | 0.0279191163 | 0.0291528489 | 0.0038170115 | 0.0626797646 |
| 624-630 | 0.0279191163 | -0.090925239 | 0.0112946980 | -0.007960867 | -0.037265097 | -0.017057731 | -0.091210876 |
| 631-637 | -0.116060305 | -0.037265097 | -0.022443576 | -0.084197081 | -0.068390870 | -0.037265097 | -0.006618304 |
| 638-644 | -0.052707913 | -0.016808834 | -0.037265097 | 0.0488795631 | -0.035857314 | -0.098752454 | -0.004233971 |
| 645-651 | -0.005670020 | -0.188096750 | 0.0376711725 | -0.004233971 | 0.0633018366 | -0.016167854 | -0.020392233 |
| 652-658 | -0.004233971 | 0.0560665532 | -0.031198237 | 0.0607256094 | -0.004233971 | 0.0583823638 | -0.052059511 |
| 659-665 | 0.1172877517 | 0.0026147711 | 0.0541753187 | -0.007493613 | 0.0157176859 | 0.0026147711 | 0.0698085823 |
| 666-672 | -0.008625695 | 0.1161382822 | 0.0026147711 | 0.1056857189 | 0.1045290246 | 0.0375742090 | 0.0026147711 |
| 673-679 | -0.135068357 | 0.0503384961 | -0.026971276 | -0.085028572 | 0.0763248325 | -0.051111692 | -0.101987956 |
| 680-688 | -0.085028572 | 0.0325306192 | 0.0527071138 | -0.099225987 | -0.085028572 | -0.052998834 | 0.0411750317 |
| 687-693 | -0.046783506 | -0.085028572 | 0.0661434680 | -0.023459226 | -0.050119796 | -0.036430318 | 0.0077573119 |
| 694-700 | 0.0975063866 | 0.1383737979 | -0.036430318 | 0.0131287222 | 0.0833319594 | -0.019410364 | -0.036430318 |
| 701-707 | -0.049171433 | 0.0409471052 | -0.014893414 | -0.036430318 | -0.052100243 | 0.0281839128 | -0.027324749 |
| 708-714 | -0.017063703 | 0.0222941895 | 0.0940447114 | 0.0861408023 | -0.017063703 | -0.010672247 | 0.0228051171 |
| 715-721 | -0.025764366 | -0.017063703 | 0.0307137555 | 0.0522344027 | 0.0263152714 | -0.017063703 | -0.027925449 |
| 722-728 | -0.025216065 | -0.043342062 | -0.003955869 | 0.0321862809 | -0.083123442 | 0.0874923545 | -0.003955869 |
| 729-735 | 0.0099352099 | 0.0363534298 | 0.0548715364 | -0.003955869 | 0.0815310985 | 0.0058996941 | 0.1129153673 |
| 736-742 | -0.003955869 | 0.0362803697 | -0.093246201 | 0.0163453470 | 0.0204034229 | -0.015789499 | -0.081544232 |
| 743-749 | 0.0359680056 | 0.0204034229 | 0.0929792462 | -0.020125579 | 0.1925175371 | 0.0204034229 | 0.0160093688 |
| 750-756 | -0.063225508 | 0.0209536429 | 0.0204034229 | 0.0390521723 | 0.0644245773 | -0.058818937 | -0.003996761 |
| 757-763 | 0.1359721849 | -0.051602226 | 0.0916440997 | -0.003996761 | 0.0235296771 | -0.102117078 | 0.0396386138 |
| 764-770 | -0.003996761 | 0.0080748912 | -0.049634588 | 0.0428231627 | -0.003996761 | 0.0255062658 | -0.034137160 |
| 771-777 | 0.0683138871 | -0.050070172 | -0.037546010 | 0.0712606397 | 0.0032199335 | -0.050070172 | -0.019647974 |
| 778-784 | -0.039251033 | -0.017634989 | -0.050070172 | -0.082399212 | -0.157991877 | -0.122172364 | -0.050070172 |

Table A.8: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 4

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1-7 | 0.0241015713 | 0.0230763973 | 0.0188718686 | -0.002733327 | 0.0198400344 | -0.025000473 | 0.0401934512 |
| 8-14 | -0.002733327 | 0.0014185787 | 0.0647376251 | -0.099446200 | -0.002733327 | -0.057076090 | -0.005284780 |
| 9-21 | -0.039832568 | -0.002733327 | 0.0525206214 | -0.021123919 | 0.0463897538 | -0.013957063 | 0.0417518581 |
| 22-28 | -0.018767437 | 0.0090556044 | -0.013957063 | -0.064549278 | 0.0318651056 | -0.035089041 | -0.013957063 |
| 29-35 | -0.009113984 | 0.0407183508 | 0.0511981624 | -0.013957063 | 0.0155092323 | -0.053498524 | -0.029728209 |
| 36-42 | -0.014459564 | -0.040704210 | -0.019165786 | -0.097737084 | -0.014459564 | 0.0027289661 | 0.0559742695 |
| 43-49 | 0.0901494671 | -0.014459564 | 0.0102098703 | -0.024891887 | 0.0198204303 | -0.014459564 | -0.015817832 |
| 50-56 | -0.010453005 | -0.052174214 | 0.0118314806 | 0.0189628336 | 0.0280849181 | -0.040899322 | 0.0118314806 |
| 57-63 | 0.0121298911 | -0.004222905 | 0.0181321960 | 0.0118314806 | -0.053856602 | 0.0088899776 | 0.0230751819 |
| 64-70 | 0.0118314806 | 0.0064675103 | 0.0178917646 | -0.039792781 | -0.025338793 | 0.0260748184 | -0.007566987 |
| 71-77 | 0.0159955728 | -0.025338793 | -0.035972072 | -0.039036209 | -0.024422961 | -0.025338793 | -0.009655695 |
| 78-84 | -0.010126168 | -0.020836231 | -0.025338793 | 0.0295694525 | 0.0370671256 | 0.0290020501 | -0.017511589 |
| 85-91 | 0.0587436487 | 0.0031669967 | 0.0331009593 | -0.017511589 | 0.0050253404 | 0.0025163062 | -0.023255779 |
| 92-98 | -0.017511589 | -0.029434727 | -0.013981554 | 0.0161022437 | -0.017511589 | 0.0386100303 | -0.022006050 |
| 99-105 | 0.0244788464 | 0.0258642876 | 0.1100628541 | 0.0480999039 | -0.038150953 | 0.0258642876 | -0.017064734 |
| 106-112 | 0.0265500125 | 0.0357088570 | 0.0258642876 | 0.0176740789 | 0.0749787995 | 0.0109752299 | 0.0258642876 |
| 113-119 | 0.0508905692 | 0.0933313575 | -0.005397511 | 0.0115056941 | -0.014945879 | 0.0392378694 | 0.0243712563 |
| 120-126 | 0.0115056941 | 0.0361984363 | 0.0514048295 | -0.035207902 | 0.0115056941 | 0.0125555680 | 0.0789061593 |
| 127-133 | -0.054228018 | 0.0115056941 | -0.046897611 | 0.0340126974 | -0.001469784 | -0.002459853 | -0.020584303 |
| 134-140 | 0.0720285456 | 0.0767816408 | -0.002459853 | -0.027789791 | 0.0718122226 | -0.043166747 | -0.002459853 |
| 141-147 | -0.026020240 | 0.0487616499 | 0.0289407403 | -0.002459853 | -0.053346152 | 0.0227797065 | 0.0603619991 |
| 148-154 | -0.042649221 | -0.005842367 | -0.026980681 | -0.036552681 | -0.042649221 | -0.060622134 | -0.029060927 |
| 155-161 | 0.0219748286 | -0.042649221 | 0.0164159167 | -0.021278299 | 0.0080181821 | -0.042649221 | -0.004301721 |
| 162-168 | -0.009994305 | -0.051593088 | -0.027712288 | -0.026533883 | 0.0148132007 | -0.024959047 | -0.027712288 |
| 169-175 | 0.0056780521 | -0.000586202 | 0.0337526722 | -0.027712288 | -0.010465280 | -0.016292417 | 0.0055909959 |
| 176-182 | -0.027712288 | -0.064357460 | 0.0307460881 | 0.0100619311 | -0.025995595 | 0.0290892969 | -0.004776782 |
| 183-189 | -0.001453530 | -0.025995595 | -0.045855773 | 0.0152767761 | 0.0142668017 | -0.025995595 | -0.047317077 |
| 190-196 | -0.066052227 | -0.011162216 | -0.025995595 | 0.0161649380 | 0.0391247907 | -0.003618619 | -0.028726194 |
| 197-203 | -0.056220989 | 0.0341070004 | 0.0101829576 | -0.028726194 | -0.002128004 | -0.037713892 | 0.0354757695 |
| 204-210 | -0.028726194 | -0.010592113 | -0.004486177 | 0.0202240273 | -0.028726194 | -0.026189807 | 0.0186146419 |
| 211-217 | 0.0194467203 | 0.0083611531 | 0.0123100596 | 0.0637736681 | -0.011613738 | 0.0083611531 | 0.0178158067 |
| 218-224 | -0.033366113 | -0.011843922 | 0.0083611531 | 0.0708539077 | 0.0300289149 | -0.019858923 | 0.0083611531 |
| 225-231 | 0.0176744913 | 0.0506353474 | -0.008827597 | 0.0033758432 | -0.002364760 | 0.0163554577 | -0.100430111 |
| 232-238 | 0.0033758432 | 0.0251028203 | 0.0641024906 | 0.0180481616 | 0.0033758432 | -0.011726309 | 0.0529248547 |
| 239-245 | 0.0372258354 | 0.0033758432 | -0.036204851 | 0.0538816645 | -0.074560021 | -0.026756727 | 0.0018963430 |
| 246-252 | 0.0166368464 | -0.006628567 | -0.026756727 | -0.013395512 | 0.0004512262 | 0.0050819328 | -0.026756727 |
| 253-259 | -0.018805724 | 0.0272642048 | 0.0030079787 | -0.026756727 | 0.0193208941 | -0.070132778 | 0.0202249373 |
| 260-266 | -0.032627636 | 0.0056161631 | 0.0208164120 | 0.0537677511 | -0.032627636 | -0.003822056 | 0.0266949434 |
| 267-273 | 0.0453223405 | -0.032627636 | -0.086255820 | -0.014460634 | -0.003831733 | -0.032627636 | -0.020400492 |
| 274-280 | 0.0025939328 | 0.0871761546 | -0.028186095 | -0.015111868 | -0.008763394 | 0.0304163620 | -0.028186095 |
| 281-287 | -0.044400485 | -0.037643510 | 0.0131000454 | -0.028186095 | 0.0150265393 | -0.035823961 | 0.0152988103 |
| 288-294 | -0.028186095 | -0.033826863 | 0.0560400521 | 0.0353747850 | 0.0039905391 | 0.0490917629 | -0.001508215 |
| 295-301 | 0.0459536353 | 0.0039905391 | 0.0014714338 | -0.010894847 | 0.0342502099 | 0.0039905391 | -0.033217148 |
| 302-308 | 0.0454358286 | 0.0263359281 | 0.0039905391 | 0.0533780068 | 0.0150776312 | 0.0062882346 | -0.011372268 |
| 309-315 | -0.023308371 | 0.0200743186 | -0.034138362 | -0.011372268 | -0.022875983 | 0.0413508892 | 0.0280241255 |
| 316-322 | -0.011372268 | -0.025473908 | -0.100122482 | -0.055262186 | -0.011372268 | -0.014164601 | -0.012330602 |
| 323-329 | -0.043423398 | 0.0179422434 | 0.0770577624 | 0.0099878362 | -0.020690038 | 0.0179422434 | -0.043380412 |
| 330-336 | -0.092402663 | -0.043519694 | 0.0179422434 | 0.1096462469 | 0.1030766492 | 0.0133645933 | 0.0179422434 |
| 337-343 | 0.0411489754 | 0.0622509717 | 0.0052557368 | 0.0068333686 | -0.051013761 | -0.014209880 | 0.0280738678 |
| 344-350 | 0.0068333686 | 0.0585446222 | 0.0836105859 | 0.0233168358 | 0.0068333686 | -0.057195669 | -0.016205625 |
| 351-357 | 0.0076451461 | 0.0068333686 | -0.077719971 | 0.0089505533 | -0.006949699 | -0.025448509 | -0.033471454 |
| 358-364 | -0.036541633 | -0.050807423 | -0.025448509 | -0.053143310 | -0.018863263 | -0.040353307 | -0.025448509 |
| 365-371 | -0.039480319 | 0.0170185663 | 0.0984179663 | -0.025448509 | -0.009135648 | 0.0517553603 | 0.0584498937 |
| 372-378 | 0.0015327499 | -0.090027551 | -0.043758263 | 0.0103548314 | 0.0015327499 | -0.057381733 | 0.0321410407 |
| 379-385 | 0.0692807552 | 0.0015327499 | -0.012096343 | 0.0198372885 | 0.0559197127 | 0.0015327499 | -0.094099636 |
| 386-392 | -0.038946622 | -0.003935028 | -0.022241868 | -0.046990321 | -0.015972960 | 0.0718086076 | -0.022241868 |

Table A.9: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 4 - continued

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 393-399 | -0.028327157 | -0.002840880 | 0.0763188261 | -0.022241868 | -0.043160712 | 0.0208066628 | 0.0742810621 |
| 400-406 | -0.022241868 | -0.021754189 | -0.029068117 | 0.0262359553 | -0.011428050 | -0.065188670 | -0.046212719 |
| 407-413 | 0.0297613925 | -0.011428050 | -0.025583993 | 0.0104370213 | 0.0344131735 | -0.011428050 | -0.048149641 |
| 414-420 | 0.0111750512 | 0.0839587619 | -0.011428050 | -0.049143360 | -0.000943539 | 0.0422327722 | -0.005775237 |
| 421-427 | -0.010395181 | -0.020387593 | 0.0336735908 | -0.005775237 | -0.079639009 | -0.014178765 | 0.1137117471 |
| 428-434 | -0.005775237 | -0.003378831 | -0.060147866 | -0.021729614 | -0.005775237 | -0.037634502 | -0.027698324 |
| 435-441 | 0.0101082331 | -0.000166270 | 0.0495647429 | 0.1039875526 | -0.008269478 | -0.000166270 | -0.026352183 |
| 442-448 | -0.068381763 | -0.047444872 | -0.000166270 | 0.0553607951 | 0.0588285752 | -0.009269057 | -0.000166270 |
| 449-455 | 0.0475811663 | 0.0435163204 | 0.0380590965 | 0.0016144528 | -0.063578850 | -0.022823879 | -0.026717599 |
| 456-462 | 0.0016144528 | 0.0392125789 | 0.0785635883 | -0.035348023 | 0.0016144528 | -0.003904372 | 0.0525052813 |
| 463-469 | 0.0519878757 | 0.0016144528 | -0.073976507 | -0.005465322 | -0.061313824 | -0.006879278 | -0.049128497 |
| 470-476 | -0.030872703 | 0.1208765626 | -0.006879278 | -0.063841277 | 0.0544068674 | -0.012618866 | -0.006879278 |
| 477-483 | -0.022610512 | 0.0141675332 | 0.0186261264 | -0.006879278 | -0.081897260 | 0.0073874487 | 0.0542640627 |
| 484-490 | 0.0164145124 | -0.005615528 | -0.004898121 | 0.0448226552 | 0.0164145124 | 0.0105660256 | 0.0194463009 |
| 491-497 | 0.0786072860 | 0.0164145124 | -0.001628667 | 0.0272454436 | 0.0073140176 | 0.0164145124 | 0.0268043861 |
| 498-504 | -0.019296655 | 0.0021805538 | 0.0130198279 | -0.029241589 | 0.0265966395 | 0.0526712158 | 0.0130198279 |
| 505-511 | -0.031213366 | 0.0278697318 | 0.0189655969 | 0.0130198279 | -0.084240704 | -0.023062585 | 0.0227589776 |
| 512-518 | 0.0130198279 | -0.010196094 | -0.011412611 | -0.031691086 | -0.017076848 | -0.061541367 | -0.066643448 |
| 519-525 | 0.0765289759 | -0.017076848 | -0.052879642 | -0.022707213 | 0.0482379147 | -0.017076848 | -0.096305804 |
| 526-532 | -0.059461735 | 0.0733349383 | -0.017076848 | -0.034881011 | -0.037677901 | 0.0649452859 | 0.0063284031 |
| 533-539 | -0.098710275 | -0.092528942 | -0.040639705 | 0.0063284031 | -0.045778576 | -0.006967901 | 0.1110472411 |
| 540-546 | 0.0063284031 | 0.0213056840 | -0.018683796 | 0.0131787117 | 0.0063284031 | -0.050913415 | -0.061554792 |
| 547-553 | -0.040526830 | 0.0003800171 | 0.0291017423 | 0.0866049984 | -0.027778486 | 0.0003800171 | 0.0200168490 |
| 554-560 | 0.0178397260 | -0.006942162 | 0.0003800171 | 0.0895822901 | 0.0883710588 | -0.042928189 | 0.0003800171 |
| 561-567 | 0.0324050165 | 0.0777571570 | -0.033767451 | 0.0184366214 | -0.040854269 | 0.0126153300 | 0.0463396920 |
| 568-574 | 0.0184366214 | 0.0388640995 | 0.0377049666 | 0.0134897566 | 0.0184366214 | 0.0515862335 | 0.0501332427 |
| 575-581 | 0.0699910844 | 0.0184366214 | -0.054412218 | 0.0182684445 | 0.0176502303 | 0.0033243998 | -0.019212018 |
| 582-588 | 0.0264187127 | -0.020806948 | 0.0033243998 | 0.0799940262 | 0.0628425578 | 0.0824777105 | 0.0033243998 |
| 589-595 | -0.006748890 | 0.0500364497 | 0.0440164256 | 0.0033243998 | -0.024176323 | -0.014702428 | 0.0276528725 |
| 596-602 | -0.006134741 | -0.023814170 | 0.0110255990 | 0.0095782976 | -0.006134741 | -0.019740867 | 0.0097569418 |
| 603-609 | 0.0480454560 | -0.006134741 | 0.0211700363 | -0.009871489 | 0.0602634525 | -0.006134741 | -0.024435163 |
| 610-616 | 0.0273121617 | 0.0103586370 | 0.0204229942 | -0.033025577 | -0.054405128 | 0.0050635150 | 0.0204229942 |
| 617-623 | -0.038874779 | 0.0229253656 | 0.0660411695 | 0.0204229942 | -0.046695745 | 0.0251769711 | -0.029487021 |
| 624-630 | 0.0204229942 | -0.048590937 | -0.036422463 | 0.0354401913 | 0.0098555343 | -0.011970031 | -0.054318178 |
| 631-637 | 0.1148687601 | 0.0098555343 | -0.017778859 | 0.0352249626 | -0.000276256 | 0.0098555343 | -0.045957472 |
| 638-644 | 0.0065356916 | 0.0572970019 | 0.0098555343 | -0.058857841 | -0.040582906 | 0.0730201866 | -0.000879126 |
| 645-651 | -0.008237599 | -0.016652897 | -0.019930530 | -0.000879126 | -0.058620214 | 0.0158876080 | 0.0101018360 |
| 652-658 | -0.000879126 | -0.015515836 | 0.0220055932 | -0.008027321 | -0.000879126 | 0.0258554935 | -0.023973538 |
| 659-665 | 0.0032412626 | -0.006887678 | 0.0719931665 | 0.0521168286 | -0.050195884 | -0.006887678 | 0.0313295710 |
| 666-672 | 0.0236663032 | -0.009695318 | -0.006887678 | 0.0389514271 | 0.0370242283 | -0.064512027 | -0.006887678 |
| 673-679 | 0.0491063797 | 0.0016585257 | -0.018719099 | 0.0111536212 | 0.0600728889 | 0.0094309370 | 0.0428662563 |
| 680-688 | 0.0111536212 | 0.0208123762 | -0.038702524 | -0.033424601 | 0.0111536212 | 0.0473396797 | 0.0251936929 |
| 687-693 | 0.0445064948 | 0.0111536212 | 0.0312424353 | 0.0070310956 | 0.0374983539 | -0.027275771 | -0.014959983 |
| 694-700 | 0.0364251142 | 0.0613235418 | -0.027275771 | 0.0224207912 | -0.005836897 | 0.0228411349 | -0.027275771 |
| 701-707 | 0.0060328310 | 0.0131723273 | -0.036437807 | -0.027275771 | -0.021573924 | 0.0137710759 | 0.0059221200 |
| 708-714 | -0.039602299 | -0.088900561 | 0.0147676944 | 0.0328099759 | -0.039602299 | 0.0074793565 | 0.0153216580 |
| 715-721 | -0.068423578 | -0.039602299 | -0.025100455 | -0.025002198 | 0.0205680249 | -0.039602299 | -0.076306362 |
| 722-728 | 0.0433420852 | 0.0169731069 | -0.030259535 | -0.037739852 | -0.006615766 | -0.022147685 | -0.030259535 |
| 729-735 | -0.020657948 | -0.008463701 | 0.0082352528 | -0.030259535 | -0.015459718 | -0.047916209 | -0.057075313 |
| 736-742 | -0.030259535 | -0.030582621 | -0.009769762 | -0.016418901 | -0.016787565 | -0.056056530 | -0.001047498 |
| 743-749 | 0.0336602877 | -0.016787565 | -0.018310741 | -0.029380719 | -0.025847781 | -0.016787565 | -0.001863632 |
| 750-756 | -0.006360624 | 0.0045137710 | -0.016787565 | -0.041128431 | 0.0093728358 | 0.0547517157 | 0.0024729333 |
| 757-763 | 0.0539279691 | 0.0153044298 | 0.0021051806 | 0.0024729333 | 0.0078126696 | 0.0225139446 | 0.0376068028 |
| 764-770 | 0.0024729333 | -0.000777678 | -0.025324393 | -0.071724887 | 0.0024729333 | -0.007266144 | 0.0466437041 |
| 771-777 | 0.0354463211 | -0.004514360 | 0.0122976754 | 0.0460128646 | -0.022480808 | -0.004514360 | 0.0222993769 |
| 778-784 | -0.004203542 | -0.084836378 | -0.004514360 | -0.037671532 | 0.0245755871 | -0.044981738 | -0.004514360 |

Table A.10: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 5

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1-7 | -0.008347195 | 0.0020927944 | 0.0029989866 | 0.0000920102 | -0.025784686 | -0.000497003 | 0.0029410876 |
| 8-14 | 0.0000920102 | -0.007375376 | 0.0144341386 | -0.010707819 | 0.0000920102 | 0.0063331015 | 0.0098908566 |
| 9-21 | 0.0192551751 | 0.0000920102 | -0.038584933 | 0.0070623442 | 0.0171468947 | -0.011644864 | -0.008389180 |
| 22-28 | -0.000591226 | 0.0000145954 | -0.011644864 | -0.012505169 | 0.0218674173 | 0.0196209863 | -0.011644864 |
| 29-35 | -0.013340871 | 0.0105932941 | -0.028819732 | -0.011644864 | 0.0024858682 | 0.0167157401 | -0.005487669 |
| 36-42 | 0.0008601846 | 0.0056024611 | 0.0112304004 | -0.008017854 | 0.0008601846 | -0.007175993 | 0.0073998665 |
| 43-49 | -0.028262080 | 0.0008601846 | -0.006680815 | -0.007311451 | -0.017353030 | 0.0008601846 | 0.0104133006 |
| 50-56 | 0.0222672681 | -0.023425069 | -0.006337139 | 0.0200986144 | -0.011234678 | 0.0130861528 | -0.006337139 |
| 57-63 | 0.0142217849 | -0.003249940 | -0.014850413 | -0.006337139 | 0.0144466985 | 0.0069737138 | 0.0151601346 |
| 64-70 | -0.006337139 | -0.002106243 | -0.004652544 | 0.0014246692 | -0.010171104 | 0.0092678069 | -0.004042613 |
| 71-77 | 0.0345613017 | -0.010171104 | 0.0056043432 | -0.014080666 | -0.015118080 | -0.010171104 | -0.016321551 |
| 78-84 | -0.027928309 | -0.005112925 | -0.010171104 | 0.0066510438 | -0.006220394 | 0.0254827744 | -0.006701270 |
| 85-91 | -0.028357697 | -0.006711737 | -0.006060983 | -0.006701270 | -0.009644199 | -0.020565623 | 0.0115046243 |
| 92-98 | -0.006701270 | -0.006855544 | -0.016513379 | 0.0024740343 | -0.006701270 | -0.018983583 | 0.0236183293 |
| 99-105 | 0.0077345778 | 0.0079377488 | 0.0191019103 | 0.0220771477 | 0.0080684440 | 0.0079377488 | 0.0072910298 |
| 106-112 | -0.004619364 | 0.0262196015 | 0.0079377488 | 0.0188835464 | 0.0218311381 | -0.001818780 | 0.0079377488 |
| 113-119 | -0.011662369 | -0.006315656 | -0.029660888 | -0.010274132 | -0.021803753 | 0.0228153246 | 0.0198489796 |
| 120-126 | -0.010274132 | 0.0365616785 | 0.0273523698 | -0.024903468 | -0.010274132 | -0.020037319 | -0.015462763 |
| 127-133 | 0.0029830581 | -0.010274132 | -0.007478464 | 0.0218602287 | 0.0165959306 | -0.009119159 | -0.017366953 |
| 134-140 | 0.0310230862 | -0.039774742 | -0.009119159 | -0.026049201 | -0.008690199 | -0.014251153 | -0.009119159 |
| 141-147 | -0.018818480 | -0.005252327 | -0.030108148 | -0.009119159 | -0.005768514 | 0.0425626301 | -0.017387085 |
| 148-154 | -0.013872335 | -0.002078029 | 0.0180023581 | -0.005129479 | -0.013872335 | -0.016694966 | -0.012366914 |
| 155-161 | -0.013164502 | -0.013872335 | -0.033921973 | -0.047023802 | -0.014359898 | -0.013872335 | 0.0001675179 |
| 162-168 | 0.0036432798 | 0.0006480019 | -0.008780387 | 0.0093610788 | -0.004133136 | 0.0392159888 | -0.008780387 |
| 169-175 | -0.029107873 | -0.042098960 | -0.029655513 | -0.008780387 | -0.044960221 | -0.023746861 | -0.020176249 |
| 176-182 | -0.008780387 | 0.0145739212 | -0.000452488 | 0.0214082948 | -0.003722188 | -0.011316932 | -0.028933493 |
| 183-189 | 0.0164773727 | -0.003722188 | -0.039635857 | -0.017804602 | -0.019911917 | -0.003722188 | -0.024736346 |
| 190-196 | -0.057555121 | -0.012725452 | -0.003722188 | -0.002297754 | -0.013218268 | 0.0341649567 | -0.000438933 |
| 197-203 | 0.0111712902 | -0.011965993 | 0.0037317158 | -0.000438933 | -0.015493875 | -0.035231423 | -0.001733187 |
| 204-210 | -0.000438933 | 0.0034286645 | -0.037609260 | 0.0040128626 | -0.000438933 | 0.0242255886 | -0.003570924 |
| 211-217 | 0.0361249852 | 0.0075819382 | 0.0273643036 | 0.0150780821 | -0.004468073 | 0.0075819382 | 0.0024029620 |
| 218-224 | -0.025484866 | 0.0117221598 | 0.0075819382 | 0.0283833302 | 0.0195985217 | -0.001239289 | 0.0075819382 |
| 225-231 | 0.0301094654 | 0.0333492130 | 0.0155955701 | -0.002077676 | -0.036606559 | -0.025285209 | 0.0136809629 |
| 232-238 | -0.002077676 | 0.0015524970 | -0.009520216 | -0.021926695 | -0.002077676 | -0.008325665 | -0.015799138 |
| 239-245 | 0.0013369550 | -0.002077676 | -0.002698707 | -0.012654962 | 0.0146911331 | 0 | -0.011708922 |
| 246-252 | 0.0034838175 | -0.020511002 | 0 | -0.021250338 | -0.008708461 | 0.0168659112 | 0 |
| 253-259 | 0.0059267277 | 0.0071179593 | -0.003711443 | 0 | 0.0027393124 | 0.0185435843 | -0.007967529 |
| 260-266 | 0.0065423934 | -0.012232776 | -0.032674414 | -0.002890353 | 0.0065423934 | 0.0181346690 | 0.0129454715 |
| 267-273 | 0.0174812330 | 0.0065423934 | -0.008558142 | -0.016763215 | -0.032653483 | 0.0065423934 | -0.016266609 |
| 274-280 | -0.014818587 | 0.0110455288 | 0.0066864515 | -0.028516572 | 0.0114248619 | 0.0081173540 | 0.0066864515 |
| 281-287 | -0.016243022 | 0.0120536590 | -0.022082650 | 0.0066864515 | 0.0079414764 | -0.038231537 | 0.0216856640 |
| 288-294 | 0.0066864515 | -0.027563401 | -0.013875174 | -0.008196642 | -0.004948481 | -0.013904095 | -0.030501289 |
| 295-301 | 0.0069111041 | -0.004948481 | -0.006275088 | -0.054972807 | 0.0138350693 | -0.004948481 | 0.0271054709 |
| 302-308 | -0.006713703 | 0.0161815067 | -0.004948481 | -0.017023949 | -0.044855142 | 0.0057362785 | 0 |
| 309-315 | -0.008903707 | -0.031940164 | 0.0041402216 | 0 | 0.0107983133 | -0.002561311 | -0.003272866 |
| 316-322 | 0 | -0.012957033 | -0.003599061 | 0.0129617072 | 0 | -0.023083532 | -0.040295968 |
| 323-329 | -0.004816334 | 0.0033964784 | 0.0208054633 | 0.0008593732 | -0.006008889 | 0.0033964784 | 0.0109631951 |
| 330-336 | -0.009645536 | 0.0082002451 | 0.0033964784 | 0.0333448986 | 0.0260137744 | -0.013548923 | 0.0033964784 |
| 337-343 | 0.0317634501 | 0.0075069606 | -0.019202236 | 0.0109814640 | -0.001774556 | 0.0097182905 | 0.0297318110 |
| 344-350 | 0.0109814640 | 0.0147947254 | 0.0019834502 | -0.019529325 | 0.0109814640 | -0.017114393 | -0.023936310 |
| 351-357 | 0.0084410885 | 0.0109814640 | -0.000333873 | -0.003113147 | 0.0212690059 | 0 | -0.001732781 |
| 358-364 | -0.007985394 | -0.018803623 | 0 | -0.013328272 | -0.053404512 | -0.013501140 | 0 |
| 365-371 | -0.056206876 | -0.050934718 | -0.006164012 | 0 | 0.0236891488 | 0.0065042078 | 0.0041244676 |
| 372-378 | 0.0075447316 | -0.008074148 | 0.0074411066 | -0.007695325 | 0.0075447316 | -0.010598633 | -0.006558435 |
| 379-385 | 0.0168464715 | 0.0075447316 | -0.003009363 | -0.019670662 | -0.032479288 | 0.0075447316 | 0.0067713380 |
| 386-392 | 0.0130570571 | -0.002142726 | 0.0105436316 | 0.0037780358 | -0.014834232 | 0.0132707440 | 0.0105436316 |

Table A.11: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 5 - continued

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 393-399 | 0.0078580025 | -0.014943500 | -0.013648776 | 0.0105436316 | -0.027918799 | -0.012725592 | -0.011349593 |
| 400-406 | 0.0105436316 | 0.0066806502 | -0.031362445 | 0.0267458414 | 0.0067925032 | 0.0265542059 | 0.0203353588 |
| 407-413 | 0.0248205383 | 0.0067925032 | -0.029049207 | -0.038815193 | -0.028854357 | 0.0067925032 | -0.038424470 |
| 414-420 | -0.017441891 | 0.0046288480 | 0.0067925032 | 0.0075610102 | 0.0061643642 | 0.0098728598 | 0 |
| 421-427 | -0.004222449 | 0.0062320523 | 0.0183743718 | 0 | -0.038126456 | -0.031813794 | 0.0030947841 |
| 428-434 | 0 | -0.051117205 | -0.002706075 | -0.004722106 | 0 | 0.0163558973 | -0.022766570 |
| 435-441 | 0.0013246549 | 0.0005090020 | 0.0323691398 | 0.0102152340 | -0.013602342 | 0.0005090020 | -0.038142087 |
| 442-448 | -0.001187789 | 0.0047596580 | 0.0005090020 | 0.0045025542 | 0.0137927486 | -0.006448064 | 0.0005090020 |
| 449-455 | 0.0455663195 | 0.0202984313 | 0.0050923704 | 0.0237515050 | 0.0079553452 | -0.015224131 | 0.0090976207 |
| 456-462 | 0.0237515050 | 0.0001352382 | 0.0091220025 | -0.023098548 | 0.0237515050 | -0.000041425 | -0.001482002 |
| 463-469 | 0.0227333049 | 0.0237515050 | 0.0028031531 | -0.031526451 | -0.003448346 | -0.003304559 | -0.032830697 |
| 470-476 | -0.026761771 | 0.0034608436 | -0.003304559 | -0.015126381 | -0.019118360 | 0.0008231157 | -0.003304559 |
| 477-483 | -0.014334072 | -0.047195144 | 0.0032688903 | -0.003304559 | -0.001024082 | -0.011654063 | 0.0225610246 |
| 484-490 | 0.0103307289 | 0.0031503919 | -0.010085178 | -0.022921299 | 0.0103307289 | 0.0045945509 | -0.014357481 |
| 491-497 | 0.0165012005 | 0.0103307289 | -0.017393413 | 0.0277659532 | 0.0156203822 | 0.0103307289 | 0.0121276793 |
| 498-504 | 0.0011106435 | -0.030479918 | 0.0249969137 | -0.008597192 | -0.004155712 | -0.027206045 | 0.0249969137 |
| 505-511 | -0.003028165 | 0.0228852854 | 0.0362264775 | 0.0249969137 | -0.018070313 | 0.0235144366 | 0.0196458731 |
| 512-518 | 0.0249969137 | -0.017488326 | -0.026371863 | -0.032750510 | 0.0147701420 | -0.036640713 | -0.022708703 |
| 519-525 | 0.0089820850 | 0.0147701420 | -0.033410877 | -0.005533602 | 0.0136643760 | 0.0147701420 | 0.0174481315 |
| 526-532 | 0.0277348977 | 0.0261072489 | 0.0147701420 | -0.034492170 | -0.021162961 | -0.003653763 | 0.0106683224 |
| 533-539 | -0.022462429 | 0.0106157882 | 0.0167198862 | 0.0106683224 | 0.0063440619 | 0.0225403761 | 0.0248133662 |
| 540-546 | 0.0106683224 | -0.005797528 | -0.012081636 | -0.034643128 | 0.0106683224 | -0.038187101 | -0.004951399 |
| 547-553 | 0.0103146409 | 0.0136246089 | 0.0204436218 | 0.0280562122 | 0.0099186028 | 0.0136246089 | -0.004216826 |
| 554-560 | -0.007574644 | -0.035231476 | 0.0136246089 | 0.0162634157 | 0.0329728215 | 0.0006618895 | 0.0136246089 |
| 561-567 | -0.004891672 | 0.0085662903 | -0.048592852 | 0.0150825984 | -0.013648675 | -0.013771713 | 0.0019051176 |
| 568-574 | 0.0150825984 | 0.0153078416 | 0.0241531074 | -0.012787445 | 0.0150825984 | -0.005599314 | 0.0352389796 |
| 575-581 | 0.0224858561 | 0.0150825984 | -0.043809392 | -0.025139585 | -0.003100422 | 0.0012933042 | -0.003864852 |
| 582-588 | -0.052735096 | 0.0073291782 | 0.0012933042 | -0.010000862 | 0.0257881217 | -0.005135975 | 0.0012933042 |
| 589-595 | -0.005062657 | 0.0127387680 | 0.0321774388 | 0.0012933042 | -0.002320127 | -0.041457489 | 0.0102238467 |
| 596-602 | 0.0047849964 | -0.032088604 | -0.017226421 | -0.015284641 | 0.0047849964 | 0.0074983395 | 0.0250582774 |
| 603-609 | 0.0384898730 | 0.0047849964 | 0.0098738779 | 0.0263967421 | 0.0191105979 | 0.0047849964 | -0.013850302 |
| 610-616 | -0.006160678 | 0.0009265974 | 0.0179576778 | -0.014494964 | 0.0050391021 | 0.0036000565 | 0.0179576778 |
| 617-623 | 0.0097503922 | 0.0226088689 | 0.0295245984 | 0.0179576778 | 0.0046660017 | 0.0476100438 | 0.0389631097 |
| 624-630 | 0.0179576778 | -0.024022609 | -0.017229524 | 0.0009455478 | 0.0156117250 | -0.003042550 | 0.0195019189 |
| 631-637 | 0.0210662564 | 0.0156117250 | -0.009336050 | 0.0409242273 | 0.0394353797 | 0.0156117250 | -0.022380411 |
| 638-644 | 0.0250480581 | 0.0254763051 | 0.0156117250 | -0.017048769 | 0.0114492465 | 0.0308917405 | 0.0065271736 |
| 645-651 | -0.021642501 | -0.015151363 | -0.043863827 | 0.0065271736 | -0.019583992 | 0.0359310236 | 0.0305145117 |
| 652-658 | 0.0065271736 | 0.0007287545 | -0.038174143 | -0.003556733 | 0.0065271736 | 0.0064642815 | -0.001689314 |
| 659-665 | -0.031498952 | 0.0113311261 | 0.0192450378 | 0.0309111286 | -0.001631593 | 0.0113311261 | -0.014432609 |
| 666-672 | -0.035527888 | -0.002109237 | 0.0113311261 | 0.0492347655 | 0.0077479229 | -0.036837931 | 0.0113311261 |
| 673-679 | 0.0171484044 | 0.0244221193 | -0.034096043 | 0.0078669890 | -0.015730244 | 0.0133929172 | 0.0152992226 |
| 680-688 | 0.0078669890 | -0.012449375 | 0.0428465089 | -0.008260583 | 0.0078669890 | -0.035060747 | 0.0340880270 |
| 687-693 | 0.0302807204 | 0.0078669890 | -0.039839097 | 0.0092945666 | -0.004496219 | -0.011905846 | -0.002418235 |
| 694-700 | 0.0074940779 | 0.0248136178 | -0.011905846 | -0.054833583 | 0.0215125601 | 0.0134078876 | -0.011905846 |
| 701-707 | -0.003142988 | 0.0012960954 | 0.0450016461 | -0.011905846 | 0.0052000465 | 0.0018553333 | 0.0153982097 |
| 708-714 | -0.012147111 | -0.016764129 | -0.021585148 | -0.021628350 | -0.012147111 | -0.005326504 | -0.010442688 |
| 715-721 | 0.0314903194 | -0.012147111 | -0.001041885 | -0.005845722 | 0.0342420456 | -0.012147111 | -0.016805236 |
| 722-728 | -0.006394175 | -0.024479953 | -0.012164048 | -0.019810752 | 0.0051894444 | 0.0032397794 | -0.012164048 |
| 729-735 | -0.000576051 | -0.006382954 | 0.0347176802 | -0.012164048 | -0.012407080 | -0.005198570 | 0.0028770967 |
| 736-742 | -0.012164048 | -0.009355512 | 0.0141772962 | 0.0225226997 | 0.0015228055 | -0.015723390 | 0.0168647702 |
| 743-749 | 0.0172903493 | 0.0015228055 | -0.001058031 | 0.0100568787 | 0.0207453720 | 0.0015228055 | -0.009168288 |
| 750-756 | 0.0082075548 | 0.0153161102 | 0.0015228055 | -0.011429383 | 0.0362349598 | 0.0273252126 | -0.002425388 |
| 757-763 | -0.012167439 | -0.041038779 | -0.017494712 | -0.002425388 | -0.008701938 | 0.0186944458 | 0.0180905955 |
| 764-770 | -0.002425388 | -0.039289373 | -0.005327867 | 0.0171755321 | -0.002425388 | 0.0101115234 | -0.043026276 |
| 771-777 | 0.0089282266 | 0.0121405891 | 0.0418144900 | 0.0296821352 | -0.022661841 | 0.0121405891 | -0.005932428 |
| 778-784 | 0.0064500715 | 0.0418955932 | 0.0121405891 | -0.019788077 | 0.0147786087 | -0.026806261 | 0.0121405891 |

Table A.12: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 6

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1-7 | 0.0163142225 | -0.029001151 | 0.0495411241 | -0.015404335 | -0.003701840 | 0.0086573417 | -0.012150626 |
| 8-14 | -0.015404335 | -0.059413390 | -0.078609411 | 0.0053689569 | -0.015404335 | 0.0001278558 | -0.021766333 |
| 9-21 | 0.0114254720 | -0.015404335 | -0.006164931 | 0.0332544721 | 0.0294628210 | -0.008060799 | 0.0244851806 |
| 22-28 | 0.0045328979 | -0.041865368 | -0.008060799 | -0.036506583 | 0.0185072657 | 0.0419253504 | -0.008060799 |
| 29-35 | -0.024168266 | 0.0131387432 | -0.069820535 | -0.008060799 | 0.0427984421 | 0.0124309604 | -0.033400605 |
| 36-42 | -0.007646129 | -0.007998562 | -0.040595617 | -0.080517057 | -0.007646129 | 0.0007648307 | 0.0044412213 |
| 43-49 | -0.052722321 | -0.007646129 | -0.039889895 | -0.010120907 | -0.047653897 | -0.007646129 | 0.0012221628 |
| 50-56 | 0.0455715402 | -0.077335853 | 0.0037730805 | -0.021389109 | -0.011238885 | -0.039105778 | 0.0037730805 |
| 57-63 | 0.0194257079 | 0.0332291833 | -0.003833687 | 0.0037730805 | 0.0117047429 | -0.002183824 | 0.0247200476 |
| 64-70 | 0.0037730805 | -0.076915531 | -0.039237858 | -0.042372264 | -0.014845490 | 0.0266751445 | 0.0025223983 |
| 71-77 | 0.0175194085 | -0.014845490 | -0.006254469 | -0.053764000 | -0.023879830 | -0.014845490 | 0.0145614624 |
| 78-84 | -0.053084359 | -0.029001898 | -0.014845490 | -0.005825506 | 0.0020836815 | 0.0095340608 | -0.000148295 |
| 85-91 | 0.0124689977 | -0.007549094 | 0.0657937251 | -0.000148295 | -0.034401570 | -0.072611017 | -0.041183733 |
| 92-98 | -0.000148295 | -0.044695637 | -0.049643672 | 0.0033524211 | -0.000148295 | -0.009802796 | -0.003875018 |
| 99-105 | 0.0549918479 | -0.000164795 | 0.0283590743 | 0.0361763282 | -0.007537587 | -0.000164795 | 0.0410963523 |
| 106-112 | -0.014066655 | 0.0260230976 | -0.000164795 | -0.017909121 | 0.0378661979 | 0.0097059838 | -0.000164795 |
| 113-119 | -0.030915017 | -0.096753158 | 0.0180985300 | -0.008422970 | -0.030155389 | 0.0241780306 | 0.0411811107 |
| 120-126 | -0.008422970 | -0.026138254 | -0.065740213 | -0.040013150 | -0.008422970 | -0.049933437 | 0.0212007110 |
| 127-133 | 0.0351751769 | -0.008422970 | -0.036779518 | 0.0487924628 | 0.0334903400 | -0.002230380 | -0.035415316 |
| 134-140 | 0.0148751354 | -0.060475053 | -0.002230380 | -0.084296799 | -0.017424969 | 0.0107344707 | -0.002230380 |
| 141-147 | -0.026284839 | 0.0025991649 | -0.031351353 | -0.002230380 | 0.0019621891 | 0.0642534078 | -0.044411459 |
| 148-154 | 0.0213625689 | 0.0019566716 | 0.0568301538 | -0.033169042 | 0.0213625689 | 0.0145335660 | 0.0530297655 |
| 155-161 | 0.0141122803 | 0.0213625689 | 0.0195625860 | 0.0126401085 | -0.004614834 | 0.0213625689 | -0.048813368 |
| 162-168 | 0.0086725387 | -0.083194179 | -0.038372945 | -0.028808776 | -0.030104498 | 0.0333419759 | -0.038372945 |
| 169-175 | -0.014235220 | -0.041469720 | -0.038930696 | -0.038372945 | -0.061451731 | -0.017859609 | -0.022802408 |
| 176-182 | -0.038372945 | -0.015403414 | -0.063111572 | 0.0505628837 | -0.015075963 | -0.005671150 | -0.057986379 |
| 183-189 | -0.047720876 | -0.015075963 | -0.044280558 | -0.031662078 | -0.031650179 | -0.015075963 | -0.042705178 |
| 190-196 | -0.065423808 | -0.039849142 | -0.015075963 | 0.0314189018 | -0.045544047 | 0.0164206906 | 0.0108582100 |
| 197-203 | 0.0369711760 | -0.031741716 | 0.0325381810 | 0.0108582100 | 0.0194455062 | -0.033403648 | 0.0213992757 |
| 204-210 | 0.0108582100 | -0.030228880 | -0.059442501 | 0.0296341466 | 0.0108582100 | 0.0633074872 | -0.021452263 |
| 211-217 | 0.0496127494 | 0.0346403993 | 0.0218141902 | 0.0270789478 | 0.0363820412 | 0.0346403993 | -0.028633172 |
| 218-224 | -0.046054018 | 0.0648525346 | 0.0346403993 | -0.026310308 | -0.013419789 | 0.0734913083 | 0.0346403993 |
| 225-231 | 0.0290601002 | 0.0103193900 | -0.004969226 | 0.0152986012 | -0.041013963 | 0.0209400223 | 0.0185591562 |
| 232-238 | 0.0152986012 | 0.0141708882 | -0.015109039 | -0.041248533 | 0.0152986012 | 0.0468435936 | -0.007433164 |
| 239-245 | -0.004220574 | 0.0152986012 | -0.055445193 | -0.007915527 | 0.0038581690 | -0.016675541 | 0.0009691968 |
| 246-252 | -0.011724064 | -0.049983474 | -0.016675541 | -0.044554475 | -0.049344034 | -0.007764729 | -0.016675541 |
| 253-259 | 0.0221321857 | 0.0324326680 | -0.012005665 | -0.016675541 | 0.0006880571 | 0.0026744272 | -0.012242102 |
| 260-266 | 0.0173237565 | 0.0062051030 | -0.045579458 | -0.026150222 | 0.0173237565 | 0.0609311234 | 0.0470987977 |
| 267-273 | 0.0380455845 | 0.0173237565 | 0.0034513164 | 0.0097778955 | 0.0126022817 | 0.0173237565 | -0.004547904 |
| 274-280 | -0.016084977 | -0.027090937 | 0.0315545293 | -0.062171083 | 0.0104115214 | -0.002244791 | 0.0315545293 |
| 281-287 | -0.056374542 | 0.0060786434 | 0.0276518582 | 0.0315545293 | 0.0367163946 | -0.035541124 | 0.0524851972 |
| 288-294 | 0.0315545293 | -0.014863328 | -0.055481873 | -0.027286605 | 0.0065212190 | -0.021760573 | -0.011573148 |
| 295-301 | 0.0332865041 | 0.0065212190 | -0.036535871 | -0.049767507 | 0.0169788777 | 0.0065212190 | 0.0387233963 |
| 302-308 | 0.0329804562 | 0.0253287511 | 0.0065212190 | -0.018236887 | -0.070445986 | 0.0019832453 | -0.001706437 |
| 309-315 | -0.044991392 | -0.060317622 | -0.011950308 | -0.001706437 | 0.0547403610 | 0.0409274925 | -0.020488844 |
| 316-322 | -0.001706437 | -0.048249573 | -0.025106970 | -0.003810870 | -0.001706437 | -0.056565985 | -0.047403001 |
| 323-329 | 0.0108994055 | 0.0171241054 | 0.0184607136 | -0.019694761 | 0.0317098080 | 0.0171241054 | -0.006454555 |
| 330-336 | 0.0532834981 | -0.004171699 | 0.0171241054 | 0.0409773240 | 0.0351580001 | 0.0383482137 | 0.0171241054 |
| 337-343 | 0.0369174148 | -0.007974383 | -0.029279109 | 0.0092998674 | 0.0240534550 | 0.0724400436 | 0.0582888761 |
| 344-350 | 0.0092998674 | 0.0386639669 | -0.051865538 | -0.013092918 | 0.0092998674 | -0.022324682 | -0.013068565 |
| 351-357 | 0.0224245020 | 0.0092998674 | -0.010632386 | -0.040556128 | 0.0011204475 | -0.029950991 | 0.0284244595 |
| 358-364 | 0.0154034376 | -0.020593157 | -0.029950991 | -0.028308108 | -0.065818178 | -0.023102105 | -0.029950991 |
| 365-371 | -0.104823324 | -0.044656575 | -0.028915086 | -0.029950991 | 0.0283059433 | 0.0389784989 | 0.0144199720 |
| 372-378 | 0.0244336506 | 0.0062499796 | 0.0506070045 | 0.0528675466 | 0.0244336506 | -0.022267969 | -0.002075613 |
| 379-385 | 0.0135641609 | 0.0244336506 | -0.005963788 | -0.041883586 | -0.042897522 | 0.0244336506 | -0.026566225 |
| 386-392 | -0.001732894 | 0.0606220876 | 0.0232791354 | 0.0269296043 | -0.003486460 | 0.0006928777 | 0.0232791354 |

Table A.13: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 6 - continued

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 393-399 | 0.0152776911 | -0.030778828 | -0.004008067 | 0.0232791354 | -0.015638181 | -0.031629089 | -0.054973323 |
| 400-406 | 0.0232791354 | 0.0412951201 | -0.019164722 | 0.0399502220 | 0.0311431623 | 0.0429917983 | 0.0681273169 |
| 407-413 | 0.0188698458 | 0.0311431623 | -0.013266568 | -0.060681676 | -0.022138883 | 0.0311431623 | -0.104738704 |
| 414-420 | -0.091881324 | 0.0170100073 | 0.0311431623 | 0.0032173537 | 0.0347184925 | -0.047875108 | -0.041309648 |
| 421-427 | -0.071501410 | 0.0087306968 | -0.008670732 | -0.041309648 | -0.087580015 | -0.072513661 | -0.042816130 |
| 428-434 | -0.041309648 | -0.066750450 | -0.007814393 | -0.070677381 | -0.041309648 | -0.024111754 | 0.0120086727 |
| 435-441 | 0.0125519741 | 0.0204160793 | 0.0259107776 | -0.013497995 | 0.0389207476 | 0.0204160793 | -0.016696333 |
| 442-448 | 0.0352046056 | 0.0605828579 | 0.0204160793 | 0.0146357257 | -0.033169518 | 0.0172097180 | 0.0204160793 |
| 449-455 | 0.0359055666 | 0.0173517290 | 0.0173145034 | 0.0226821866 | 0.0240684307 | 0.0088730582 | 0.0382130718 |
| 456-462 | 0.0226821866 | -0.013059448 | -0.011274375 | -0.021817846 | 0.0226821866 | -0.028232319 | -0.011935568 |
| 463-469 | 0.0224813963 | 0.0226821866 | 0.0662158846 | -0.009515751 | 0.0161878366 | -0.006396974 | -0.038533617 |
| 470-476 | -0.020520978 | -0.000553838 | -0.006396974 | -0.066872908 | -0.021350125 | -0.004779566 | -0.006396974 |
| 477-483 | -0.024399175 | -0.001688009 | 0.0378461792 | -0.006396974 | -0.018351387 | -0.028140976 | 0.0222209926 |
| 484-490 | 0.0012795736 | -0.001186960 | -0.035469831 | -0.023276719 | 0.0012795736 | -0.009486467 | -0.089267557 |
| 491-497 | 0.0321994757 | 0.0012795736 | 0.0306665473 | 0.0293317886 | 0.0142762321 | 0.0012795736 | 0.0360543099 |
| 498-504 | -0.008508671 | -0.001118152 | 0.0315832812 | 0.0118781970 | -0.006111945 | -0.008516984 | 0.0315832812 |
| 505-511 | 0.0355308357 | 0.0516858946 | 0.0609294379 | 0.0315832812 | 0.0037000936 | 0.0281021023 | 0.0148330133 |
| 512-518 | 0.0315832812 | -0.031054056 | -0.047217917 | -0.025789048 | 0.0275203598 | -0.052969329 | -0.056779053 |
| 519-525 | -0.015880055 | 0.0275203598 | 0.0017420055 | 0.0222267429 | 0.0410608280 | 0.0275203598 | 0.0576925465 |
| 526-532 | 0.0361981641 | 0.0465148356 | 0.0275203598 | -0.063673636 | -0.017669828 | 0.0084625166 | 0.0221616647 |
| 533-539 | 0.0034524064 | 0.0357297801 | -0.007780352 | 0.0221616647 | 0.0182269731 | -0.000820133 | 0.0241959622 |
| 540-546 | 0.0221616647 | 0.0123750639 | -0.010292946 | -0.018074751 | 0.0221616647 | -0.006688941 | 0.0436655459 |
| 547-553 | 0.0305638825 | 0.0283883460 | 0.0319915810 | 0.0010213237 | 0.0337845295 | 0.0283883460 | 0.0079735127 |
| 554-560 | -0.007880914 | -0.010379643 | 0.0283883460 | 0.0069733877 | 0.0067896943 | 0.0332795130 | 0.0283883460 |
| 561-567 | -0.097464713 | -0.033828670 | -0.049666966 | -0.005335610 | -0.037494802 | -0.042247054 | -0.005603956 |
| 568-574 | -0.005335610 | 0.0361802399 | 0.0586864234 | 0.0063937464 | -0.005335610 | -0.004471942 | 0.0700435858 |
| 575-581 | 0.0075128405 | -0.005335610 | -0.075785837 | -0.079998025 | 0.0039071776 | -0.000390021 | 0.0088215655 |
| 582-588 | -0.036559462 | 0.0441470101 | -0.000390021 | 0.0064051400 | 0.0664173152 | -0.000016917 | -0.000390021 |
| 589-595 | -0.008671292 | 0.0509167108 | 0.0386530506 | -0.000390021 | 0.0241898034 | -0.082367150 | 0.0591189436 |
| 596-602 | 0.0058063934 | -0.013849766 | -0.034592509 | -0.007605572 | 0.0058063934 | 0.0111249860 | 0.0290340725 |
| 603-609 | 0.0240015619 | 0.0058063934 | -0.002117388 | -0.003818706 | -0.011952756 | 0.0058063934 | 0.0418486803 |
| 610-616 | 0.0005131994 | -0.015653865 | 0.0263175171 | 0.0355989654 | 0.0393909355 | -0.018876805 | 0.0263175171 |
| 617-623 | 0.0294304491 | 0.0373685874 | -0.003744559 | 0.0263175171 | 0.0189184777 | 0.0391341516 | 0.0173625254 |
| 624-630 | 0.0263175171 | -0.011733358 | -0.016380027 | -0.035973169 | 0.0088942574 | 0.0083632069 | 0.0469675072 |
| 631-637 | 0.0219346539 | 0.0088942574 | -0.008707908 | 0.0177870078 | 0.0071459054 | 0.0088942574 | 0.0105528474 |
| 638-644 | 0.0161919784 | 0.0692730674 | 0.0088942574 | -0.001450388 | 0.0275278518 | 0.0214968672 | -0.013589707 |
| 645-651 | -0.070642156 | -0.033320890 | -0.064175869 | -0.013589707 | 0.0239822936 | -0.003234763 | 0.0537313178 |
| 652-658 | -0.013589707 | 0.0215755428 | -0.033817587 | 0.0432129770 | -0.013589707 | -0.000381255 | 0.0008874670 |
| 659-665 | -0.005051151 | 0.0233421453 | 0.0313091378 | 0.0046985978 | 0.0333051914 | 0.0233421453 | -0.002420014 |
| 666-672 | -0.036887779 | 0.0649413191 | 0.0233421453 | 0.0587355692 | 0.0078429113 | -0.008032254 | 0.0233421453 |
| 673-679 | 0.0081854836 | 0.0771639341 | -0.013465624 | -0.016377139 | -0.033112638 | 0.0480531969 | -0.012352861 |
| 680-688 | -0.016377139 | -0.069159732 | 0.0586150308 | -0.006823714 | -0.016377139 | -0.079008254 | 0.0914567010 |
| 687-693 | 0.0176069287 | -0.016377139 | -0.020033487 | 0.0118156267 | -0.047187855 | -0.026057278 | 0.0118363393 |
| 694-700 | 0.0454212608 | 0.0263366257 | -0.026057278 | -0.088688393 | 0.0430619974 | 0.0096148176 | -0.026057278 |
| 701-707 | 0.0144234536 | -0.006624999 | 0.0005508562 | -0.026057278 | -0.009085061 | 0.0407687345 | 0.0238340259 |
| 708-714 | -0.022073121 | -0.024764255 | 0.0345892457 | -0.041954008 | -0.022073121 | 0.0026509276 | -0.026435454 |
| 715-721 | 0.0248540802 | -0.022073121 | 0.0428163256 | -0.018717383 | 0.0435364052 | -0.022073121 | -0.062637187 |
| 722-728 | 0.0295197456 | -0.033475269 | -0.022805323 | -0.036622051 | -0.029468968 | -0.090387451 | -0.022805323 |
| 729-735 | 0.0555152058 | -0.012138117 | 0.0333218421 | -0.022805323 | 0.0252142976 | -0.051470805 | 0.0254643986 |
| 736-742 | -0.022805323 | -0.037695490 | 0.0219481520 | -0.031993688 | 0.0010821413 | 0.0257860665 | -0.038935169 |
| 743-749 | 0.0214971996 | 0.0010821413 | 0.0491017628 | -0.012184630 | 0.0419222348 | 0.0010821413 | -0.014723326 |
| 750-756 | -0.010336853 | 0.0446850456 | 0.0010821413 | -0.006284455 | 0.0318161711 | 0.0230808926 | 0.0029339543 |
| 757-763 | 0.0228011603 | -0.034660901 | 0.0278866154 | 0.0029339543 | -0.004680728 | -0.022758136 | 0.0327415612 |
| 764-770 | 0.0029339543 | -0.048925942 | 0.0229733344 | 0.0137639263 | 0.0029339543 | 0.0401300081 | -0.062240518 |
| 771-777 | 0.0419135784 | -0.014904313 | 0.0203112356 | 0.0063441182 | -0.039627439 | -0.014904313 | -0.059538974 |
| 778-784 | 0.0239783215 | 0.0090888634 | -0.014904313 | -0.070867296 | -0.077308065 | -0.087985399 | -0.014904313 |

Table A.14: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 7

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1-7 | 0.0046810052 | 0.0157900253 | 0.0178901982 | -0.004195552 | -0.016279938 | -0.002721639 | 0.0115263463 |
| 8-14 | -0.004195552 | -0.019545930 | -0.003468248 | -0.019925986 | -0.004195552 | -0.020774307 | 0.0192589773 |
| 9-21 | 0.0024113149 | -0.004195552 | -0.031563954 | -0.007995400 | 0.0125034575 | -0.014738010 | 0.0198426680 |
| 22-28 | -0.004463111 | -0.001390299 | -0.014738010 | -0.041162280 | 0.0192191482 | -0.006615136 | -0.014738010 |
| 29-35 | -0.016586487 | 0.0472987228 | -0.033118743 | -0.014738010 | 0.0096092846 | -0.017374652 | -0.024070916 |
| 36-42 | -0.016278068 | 0.0236540307 | -0.001656150 | -0.032548688 | -0.016278068 | -0.020235607 | 0.0421875756 |
| 43-49 | -0.027422763 | -0.016278068 | -0.008586425 | -0.006468458 | -0.041695907 | -0.016278068 | 0.0339370331 |
| 50-56 | 0.0154598729 | -0.041571818 | -0.012503075 | 0.0350777939 | -0.019855592 | -0.005114604 | -0.012503075 |
| 57-63 | -0.003102601 | -0.009424657 | -0.031702357 | -0.012503075 | 0.0033065293 | 0.0033183359 | 0.0247697863 |
| 64-70 | -0.012503075 | 0.0259026836 | -0.018531467 | -0.010180239 | -0.016970622 | 0.0097501133 | -0.003720170 |
| 71-77 | 0.0102627753 | -0.016970622 | -0.011749813 | -0.022334617 | -0.017282476 | -0.016970622 | -0.006677670 |
| 78-84 | -0.044976937 | 0.0130065989 | -0.016970622 | 0.0161989607 | 0.0091297370 | 0.0174908048 | -0.005529061 |
| 85-91 | 0.0101792938 | -0.006329859 | 0.0089502272 | -0.005529061 | -0.017528095 | -0.040219958 | 0.0198103751 |
| 92-98 | -0.005529061 | -0.012144482 | -0.039915662 | 0.0049903396 | -0.005529061 | 0.0064015261 | 0.0211166730 |
| 99-105 | 0.0276151273 | 0.0150302762 | 0.0173798372 | 0.0141380033 | -0.010782184 | 0.0150302762 | 0.0010707531 |
| 106-112 | -0.012065509 | 0.0262678536 | 0.0150302762 | 0.0108045542 | 0.0324105651 | 0.0105485286 | 0.0150302762 |
| 113-119 | -0.016600791 | -0.007528564 | -0.010337606 | -0.001168311 | -0.037043923 | 0.0249910266 | 0.0151817121 |
| 120-126 | -0.001168311 | 0.0069700845 | 0.0166850177 | -0.002554235 | -0.001168311 | -0.025130497 | 0.0232185010 |
| 127-133 | -0.001692428 | -0.001168311 | -0.035877944 | 0.0221894174 | 0.0120278945 | -0.014012169 | -0.005989133 |
| 134-140 | 0.0498002555 | -0.030564942 | -0.014012169 | -0.050042301 | 0.0222493804 | -0.003559828 | -0.014012169 |
| 141-147 | -0.031180589 | 0.0291733781 | 0.0009784972 | -0.014012169 | -0.018531322 | 0.0511621825 | -0.014876708 |
| 148-154 | -0.016205419 | 0.0045689556 | 0.0200368824 | -0.026200564 | -0.016205419 | -0.031505537 | 0.0216418287 |
| 155-161 | 0.0103272677 | -0.016205419 | -0.010700455 | -0.013701529 | -0.010219737 | -0.016205419 | 0.0039598073 |
| 162-168 | 0.0050958266 | -0.019872870 | -0.011041828 | -0.000503999 | 0.0048459504 | 0.0344476519 | -0.011041828 |
| 169-175 | -0.008233417 | -0.006217174 | -0.021775530 | -0.011041828 | -0.052755607 | -0.024660060 | -0.010754552 |
| 176-182 | -0.011041828 | -0.004799709 | 0.0017068220 | 0.0329438362 | -0.009699523 | -0.000454987 | -0.030104946 |
| 183-189 | 0.0184897574 | -0.009699523 | -0.062129338 | -0.015241760 | -0.017583543 | -0.009699523 | 0.0041209109 |
| 190-196 | -0.038427680 | -0.006251817 | -0.009699523 | 0.0238510033 | -0.009882181 | 0.0336785179 | 0.0086091576 |
| 197-203 | 0.0070955951 | 0.0041323448 | 0.0178478583 | 0.0086091576 | 0.0144406651 | -0.045398184 | -0.002276017 |
| 204-210 | 0.0086091576 | -0.003721958 | -0.030168021 | 0.0297634515 | 0.0086091576 | 0.0083922696 | 0.0000298031 |
| 211-217 | 0.0367498538 | 0.0139749562 | 0.0158339299 | 0.0236056916 | 0.0046188996 | 0.0139749562 | 0.0190140259 |
| 218-224 | -0.034231050 | 0.0319275031 | 0.0139749562 | 0.0281840445 | 0.0183576028 | -0.007196029 | 0.0139749562 |
| 225-231 | 0.0036995815 | 0.0216194307 | 0.0136141961 | 0.0006361539 | -0.037674698 | -0.017805215 | 0.0040969883 |
| 232-238 | 0.0006361539 | -0.008520277 | 0.0025230454 | -0.007521848 | 0.0006361539 | -0.001759387 | 0.0164790023 |
| 239-245 | 0.0088147705 | 0.0006361539 | -0.039961592 | 0.0130029829 | 0.0095765683 | -0.003227813 | -0.003427598 |
| 246-252 | 0.0353027042 | 0.0100851921 | -0.003227813 | -0.019918010 | -0.006217260 | 0.0239625291 | -0.003227813 |
| 253-259 | 0.0127787052 | 0.0351762591 | -0.000398702 | -0.003227813 | -0.028699297 | 0.0187029958 | 0.0098440499 |
| 260-266 | 0.0006275393 | 0.0049461963 | -0.000556687 | 0.0051137834 | 0.0006275393 | 0.0243338438 | 0.0231166690 |
| 267-273 | 0.0298023883 | 0.0006275393 | -0.018834535 | 0.0219998858 | -0.024660925 | 0.0006275393 | 0.0001077472 |
| 274-280 | 0.0003713050 | 0.0166321886 | 0.0063145894 | -0.013116278 | 0.0071309614 | 0.0087874480 | 0.0063145894 |
| 281-287 | -0.015024872 | 0.0328702682 | 0.0127886832 | 0.0063145894 | 0.0028245374 | 0.0023375022 | 0.0122236966 |
| 288-294 | 0.0063145894 | -0.029867644 | -0.000344715 | 0.0052173475 | 0.0079738862 | 0.0183947599 | -0.023711741 |
| 295-301 | 0.0182141567 | 0.0079738862 | -0.004473277 | -0.013946937 | 0.0124827978 | 0.0079738862 | 0.0159490625 |
| 302-308 | -0.001500064 | 0.0306779326 | 0.0079738862 | 0.0082164150 | -0.030523567 | -0.000702943 | -0.000619956 |
| 309-315 | -0.008413775 | -0.030101369 | 0.0198323479 | -0.000619956 | 0.0173818327 | 0.0002011518 | 0.0063256763 |
| 316-322 | -0.000619956 | -0.031159199 | 0.0019947073 | -0.010984907 | -0.000619956 | -0.027354639 | -0.027431271 |
| 323-329 | 0.0117170095 | 0.0105207200 | 0.0151684513 | 0.0083390542 | -0.010714900 | 0.0105207200 | -0.022025577 |
| 330-336 | 0.0198169169 | 0.0005258161 | 0.0105207200 | 0.0446296063 | 0.0359901097 | -0.018461289 | 0.0105207200 |
| 337-343 | 0.0076401651 | 0.0062987756 | -0.006260562 | 0.0148037005 | 0.0034427871 | 0.0221836571 | 0.0362041762 |
| 344-350 | 0.0148037005 | -0.010525547 | -0.003561390 | 0.0072058604 | 0.0148037005 | -0.005363144 | 0.0026688005 |
| 351-357 | 0.0111654845 | 0.0148037005 | -0.015595290 | 0.0030184653 | 0.0135267527 | -0.007949185 | -0.005566962 |
| 358-364 | 0.0140497123 | 0.0026305106 | -0.007949185 | -0.042173411 | -0.025288156 | -0.004124699 | -0.007949185 |
| 365-371 | -0.054288077 | -0.023363928 | -0.020620838 | -0.007949185 | 0.0204015346 | 0.0347054818 | 0.0294124311 |
| 372-378 | 0.0038558249 | -0.018331686 | 0.0089633692 | -0.004176216 | 0.0038558249 | -0.031841407 | 0.0114940856 |
| 379-385 | -0.000922744 | 0.0038558249 | -0.036144063 | -0.007510529 | 0.0043995865 | 0.0038558249 | 0.0019974741 |
| 386-392 | 0.0438435486 | 0.0058823087 | 0.0092720775 | -0.003510676 | 0.0030225572 | 0.0111721660 | 0.0092720775 |

Table A.15: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 7 - continued

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 393-399 | -0.022746838 | 0.0029709279 | 0.0206561773 | 0.0092720775 | -0.016619614 | -0.001679975 | -0.011710535 |
| 400-406 | 0.0092720775 | -0.008623487 | -0.010657548 | 0.0277218517 | 0.0092657467 | 0.0214384458 | -0.003898835 |
| 407-413 | 0.0314775247 | 0.0092657467 | -0.030213094 | -0.007711368 | -0.000593011 | 0.0092657467 | -0.079504547 |
| 414-420 | -0.006549720 | 0.0114109180 | 0.0092657467 | 0.0050881442 | -0.010504349 | 0.0149525076 | -0.005383930 |
| 421-427 | -0.028123256 | 0.0096836089 | 0.0119047859 | -0.005383930 | -0.073149920 | -0.039716443 | 0.0241375952 |
| 428-434 | -0.005383930 | -0.027509287 | -0.034939649 | -0.040295103 | -0.005383930 | -0.019597485 | 0.0066336201 |
| 435-441 | 0.0092320758 | 0.0076152469 | 0.0446295166 | 0.0185525024 | -0.023273621 | 0.0076152469 | -0.027786394 |
| 442-448 | -0.014271387 | -0.006191827 | 0.0076152469 | 0.0206426727 | 0.0081218490 | -0.014971120 | 0.0076152469 |
| 449-455 | -0.001023855 | 0.0084543434 | 0.0099615516 | 0.0159013434 | -0.010537392 | -0.013060587 | 0.0228276146 |
| 456-462 | 0.0159013434 | 0.0011026265 | 0.0180118833 | -0.013624120 | 0.0159013434 | -0.020875380 | -0.001033584 |
| 463-469 | 0.0320734468 | 0.0159013434 | -0.026278162 | -0.011938855 | 0.0059686719 | -0.000864505 | -0.037858357 |
| 470-476 | -0.023581209 | 0.0102878250 | -0.000864505 | -0.041205797 | -0.008341692 | 0.0245702430 | -0.000864505 |
| 477-483 | -0.021060992 | 0.0052913527 | 0.0178784887 | -0.000864505 | -0.019683619 | -0.017397609 | 0.0051039789 |
| 484-490 | 0.0039818037 | -0.027686225 | 0.0020027580 | 0.0040474965 | 0.0039818037 | 0.0074194803 | 0.0156371690 |
| 491-497 | 0.0378583666 | 0.0039818037 | 0.0174030845 | 0.0400422823 | 0.0448334068 | 0.0039818037 | 0.0001831922 |
| 498-504 | 0.0274132056 | 0.0000549274 | 0.0281140227 | -0.023624350 | -0.005183845 | -0.009207336 | 0.0281140227 |
| 505-511 | 0.0279269313 | 0.0536671302 | 0.0573744122 | 0.0281140227 | 0.0063264395 | 0.0455823246 | 0.0388898066 |
| 512-518 | 0.0281140227 | -0.019688201 | -0.023224483 | -0.019076879 | 0.0206684078 | -0.045249760 | -0.015356279 |
| 519-525 | 0.0197021517 | 0.0206684078 | 0.0147102800 | 0.0456670479 | 0.0295866177 | 0.0206684078 | -0.009037460 |
| 526-532 | 0.0380486610 | 0.0597964416 | 0.0206684078 | -0.049726310 | -0.026769226 | 0.0060813775 | 0.0072194310 |
| 533-539 | -0.029291969 | -0.003006508 | -0.017634875 | 0.0072194310 | -0.020865910 | 0.0116009639 | 0.0470416652 |
| 540-546 | 0.0072194310 | -0.005821255 | -0.001315465 | -0.002846155 | 0.0072194310 | -0.027963439 | -0.009302781 |
| 547-553 | 0.0005636199 | 0.0240345073 | 0.0285013426 | 0.0231340430 | -0.003733747 | 0.0240345073 | -0.004811493 |
| 554-560 | 0.0060236429 | -0.000644863 | 0.0240345073 | 0.0395518148 | 0.0486545389 | -0.000985264 | 0.0240345073 |
| 561-567 | -0.011798310 | 0.0045693267 | -0.027925901 | 0.0065069208 | -0.019024191 | -0.009642035 | 0.0106815016 |
| 568-574 | 0.0065069208 | 0.0084226851 | 0.0268049165 | -0.007027011 | 0.0065069208 | 0.0040296133 | 0.0282444269 |
| 575-581 | 0.0278609564 | 0.0065069208 | -0.046427828 | -0.024783967 | 0.0200726126 | -0.015201298 | -0.020796629 |
| 582-588 | -0.017013991 | 0.0011237702 | -0.015201298 | -0.010539674 | 0.0247872440 | 0.0033291679 | -0.015201298 |
| 589-595 | -0.009098512 | 0.0117650679 | 0.0390799567 | -0.015201298 | -0.009279718 | -0.018721856 | 0.0226798521 |
| 596-602 | -0.010529802 | -0.021906883 | 0.0024360018 | 0.0082166628 | -0.010529802 | -0.011301729 | 0.0181741755 |
| 603-609 | 0.0366925204 | -0.010529802 | 0.0104391397 | 0.0057226718 | 0.0091024954 | -0.010529802 | -0.005908515 |
| 610-616 | 0.0116128467 | 0.0217220520 | 0.0102558122 | -0.014614559 | 0.0250590714 | 0.0073371055 | 0.0102558122 |
| 617-623 | 0.0083795743 | 0.0229497355 | 0.0187582821 | 0.0102558122 | -0.005648488 | 0.0271210467 | 0.0148732944 |
| 624-630 | 0.0102558122 | -0.009265849 | 0.0089831129 | 0.0050989363 | 0.0025890570 | -0.022621495 | 0.0305721425 |
| 631-637 | 0.0425378949 | 0.0025890570 | -0.012876162 | 0.0272139874 | 0.0187173393 | 0.0025890570 | -0.020356134 |
| 638-644 | 0.0101705299 | 0.0520489931 | 0.0025890570 | -0.038116336 | 0.0155873598 | 0.0340549099 | -0.000037435 |
| 645-651 | -0.029521976 | -0.007464696 | -0.015908416 | -0.000037435 | -0.009711722 | 0.0307273605 | 0.0535375250 |
| 652-658 | -0.000037435 | 0.0126386118 | -0.004404162 | 0.0042791515 | -0.000037435 | -0.011867764 | -0.001154335 |
| 659-665 | -0.017763237 | 0.0022084647 | 0.0224375523 | 0.0262929569 | -0.022085179 | 0.0022084647 | -0.000401871 |
| 666-672 | -0.003583776 | 0.0135940641 | 0.0022084647 | 0.0303274506 | 0.0008068089 | -0.028150150 | 0.0022084647 |
| 673-679 | 0.0108845415 | 0.0235559937 | -0.024496025 | -0.005404409 | -0.010005690 | 0.0265728543 | 0.0116062114 |
| 680-688 | -0.005404409 | -0.025025882 | 0.0044496371 | -0.026322387 | -0.005404409 | -0.023594663 | 0.0290500501 |
| 687-693 | 0.0017141365 | -0.005404409 | -0.020018863 | 0.0109029032 | -0.004673029 | -0.020102803 | -0.010005873 |
| 694-700 | 0.0063772595 | 0.0238194917 | -0.020102803 | -0.043008178 | 0.0060285280 | 0.0108947611 | -0.020102803 |
| 701-707 | -0.001695087 | -0.018441655 | 0.0173933358 | -0.020102803 | -0.033440999 | 0.0021251599 | 0.0246739783 |
| 708-714 | -0.028195816 | -0.026603908 | -0.008756193 | -0.004717168 | -0.028195816 | -0.006572034 | -0.028767684 |
| 715-721 | 0.0055819779 | -0.028195816 | -0.004657127 | -0.008156057 | 0.0317937071 | -0.028195816 | -0.030422960 |
| 722-728 | 0.0059841011 | -0.019623805 | -0.024842771 | -0.026527209 | -0.003024670 | -0.024869147 | -0.024842771 |
| 729-735 | 0.0023291107 | -0.016513701 | 0.0276418532 | -0.024842771 | -0.014319388 | -0.014539237 | -0.008972581 |
| 736-742 | -0.024842771 | -0.019182887 | 0.0122277010 | 0.0074005680 | -0.006514014 | -0.023452438 | 0.0109910506 |
| 743-749 | 0.0319980940 | -0.006514014 | -0.000195842 | 0.0037042755 | 0.0095512977 | -0.006514014 | -0.017258751 |
| 750-756 | -0.008433197 | 0.0165983171 | -0.006514014 | -0.029782393 | 0.0298062278 | 0.0465849201 | -0.009344759 |
| 757-763 | -0.003961143 | -0.010627650 | -0.008327324 | -0.009344759 | -0.014363811 | 0.0069824410 | 0.0188967928 |
| 764-770 | -0.009344759 | -0.036301573 | -0.008944752 | -0.017496773 | -0.009344759 | 0.0193071865 | -0.004610733 |
| 771-777 | 0.0317259618 | 0.0101317768 | 0.0309845123 | 0.0244620688 | -0.016872891 | 0.0101317768 | -0.004506752 |
| 778-784 | 0.0163189612 | 0.0013047239 | 0.0101317768 | -0.025439995 | 0.0026322508 | -0.043140692 | 0.0101317768 |

Table A.16: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 8

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1-7 | -0.006609747 | -0.007210959 | 0.0427073896 | 0.0026467585 | 0.0288564148 | 0.0106079080 | 0.0440652985 |
| 8-14 | 0.0026467585 | 0.0076997837 | -0.041951255 | 0.0282422793 | 0.0026467585 | -0.016175178 | -0.048119072 |
| 9-21 | -0.019807276 | 0.0026467585 | 0.0099015364 | -0.025602544 | -0.016522951 | -0.001797739 | 0.0143926735 |
| 22-28 | 0.0034784123 | -0.000526395 | -0.001797739 | -0.016836435 | -0.060425565 | -0.022034951 | -0.001797739 |
| 29-35 | -0.019161531 | -0.025410488 | -0.032764840 | -0.001797739 | 0.0282011760 | -0.014313400 | 0.0135730565 |
| 36-42 | 0.0105622355 | -0.044961905 | -0.070308128 | 0.0109298154 | 0.0105622355 | -0.034826710 | 0.0224384186 |
| 43-49 | 0.0197743771 | 0.0105622355 | -0.035920452 | 0.0075131353 | -0.009887409 | 0.0105622355 | -0.069763612 |
| 50-56 | -0.019337998 | 0.0049083223 | -0.013377040 | -0.053332718 | 0.0000728046 | -0.021338582 | -0.013377040 |
| 57-63 | -0.071618826 | -0.039954069 | 0.0414789973 | -0.013377040 | -0.044802031 | -0.005784545 | -0.036839074 |
| 64-70 | -0.013377040 | -0.041641908 | -0.011329018 | -0.028897769 | 0.0123296696 | 0.0302004421 | 0.0504744024 |
| 71-77 | 0.0155881833 | 0.0123296696 | -0.010500851 | 0.0017427372 | -0.000040954 | 0.0123296696 | 0.0079713091 |
| 78-84 | 0.0002874157 | 0.0122826621 | 0.0123296696 | 0.0030960811 | -0.010107704 | 0.0033670355 | -0.012149518 |
| 85-91 | 0.0410137229 | 0.0118500390 | 0.0581384510 | -0.012149518 | -0.025437709 | -0.078644417 | -0.018954432 |
| 92-98 | -0.012149518 | -0.003720435 | -0.016561574 | -0.001569729 | -0.012149518 | 0.0504397013 | -0.000215322 |
| 99-105 | 0.0510026364 | -0.011143690 | -0.015800051 | -0.018919156 | -0.011290560 | -0.011143690 | 0.0243965996 |
| 106-112 | -0.022759951 | -0.014771489 | -0.011143690 | -0.037734334 | -0.007130423 | 0.0273321881 | -0.011143690 |
| 113-119 | 0.0253464353 | -0.022765973 | 0.0831031874 | 0.0304308062 | 0.0331977039 | -0.025975546 | 0.0026330383 |
| 120-126 | 0.0304308062 | -0.094739994 | -0.084379808 | -0.003764824 | 0.0304308062 | -0.011553630 | -0.062529982 |
| 127-133 | -0.013034302 | 0.0304308062 | 0.0141255470 | -0.006969151 | 0.0297312639 | 0.0124743218 | -0.019569859 |
| 134-140 | -0.011385973 | 0.0019791947 | 0.0124743218 | -0.030352593 | 0.0132370384 | -0.007632992 | 0.0124743218 |
| 141-147 | 0.0074118120 | 0.0287293427 | 0.0392966518 | 0.0124743218 | -0.008357424 | 0.0350532137 | 0.0560321292 |
| 148-154 | 0.0351935712 | -0.034214541 | -0.015396824 | -0.018763102 | 0.0351935712 | 0.0022290778 | 0.0246674491 |
| 155-161 | 0.0393379282 | 0.0351935712 | 0.0519219117 | 0.0090475904 | 0.0109460346 | 0.0351935712 | -0.048515260 |
| 162-168 | -0.013241049 | 0.0006611421 | -0.004266506 | -0.049875490 | 0.0146579599 | -0.026315553 | -0.004266506 |
| 169-175 | 0.0117254872 | -0.015567976 | -0.012413237 | -0.004266506 | 0.0340336997 | 0.0055761162 | 0.0005548097 |
| 176-182 | -0.004266506 | -0.030544344 | -0.013870724 | -0.034549640 | -0.004346793 | -0.018186045 | 0.0015839199 |
| 183-189 | 0.0314550459 | -0.004346793 | -0.006945535 | -0.016402223 | 0.0392346116 | -0.004346793 | 0.0338400952 |
| 190-196 | 0.0352481162 | -0.024575858 | -0.004346793 | -0.017592172 | 0.0218908434 | 0.0075308970 | 0.0221639901 |
| 197-203 | -0.006034098 | 0.0618898030 | 0.0492012187 | 0.0221639901 | 0.0178777877 | -0.009341709 | -0.016787626 |
| 204-210 | 0.0221639901 | -0.002654768 | -0.004202594 | 0.0228681951 | 0.0221639901 | 0.0198273974 | 0.0061904455 |
| 211-217 | 0.0207717322 | 0.0027288117 | -0.026272140 | -0.015810994 | 0.0357643151 | 0.0027288117 | -0.027631440 |
| 218-224 | -0.008815328 | 0.0162725375 | 0.0027288117 | -0.071031027 | -0.055905869 | 0.0447153923 | 0.0027288117 |
| 225-231 | -0.058250059 | -0.059323683 | -0.005581500 | 0.0308492655 | 0.0035703461 | 0.0211574651 | 0.0266798459 |
| 232-238 | 0.0308492655 | -0.037529915 | -0.050000465 | -0.011811424 | 0.0308492655 | 0.0296429139 | 0.0263220960 |
| 239-245 | 0.0083078306 | 0.0308492655 | -0.008529823 | -0.052781098 | -0.012696791 | -0.016947696 | 0.0222202164 |
| 246-252 | -0.014485280 | 0.0858415974 | -0.016947696 | 0.0904356511 | 0.0488711527 | -0.026853213 | -0.016947696 |
| 253-259 | 0.0628932905 | -0.019096608 | -0.003004940 | -0.016947696 | 0.0461000505 | -0.042639350 | 0.0260444160 |
| 260-266 | -0.013195423 | 0.0226945989 | -0.019633994 | -0.033709909 | -0.013195423 | 0.0223910486 | -0.030100286 |
| 267-273 | 0.0335067210 | -0.013195423 | -0.019898150 | 0.0564974173 | 0.0144124137 | -0.013195423 | 0.0081796271 |
| 274-280 | -0.014323971 | -0.050271731 | -0.011748578 | -0.026000822 | -0.020249500 | -0.029152338 | -0.011748578 |
| 281-287 | -0.001717963 | -0.052840969 | -0.005193176 | -0.011748578 | -0.007922497 | -0.002679690 | -0.067243789 |
| 288-294 | -0.011748578 | -0.003559972 | -0.008529081 | -0.005514965 | 0.0136235723 | -0.007841063 | 0.0132431327 |
| 295-301 | -0.011829559 | 0.0136235723 | 0.0333527761 | 0.0183486788 | -0.028636859 | 0.0136235723 | -0.006898188 |
| 302-308 | 0.0241918487 | -0.039620424 | 0.0136235723 | 0.0238513448 | 0.0066383848 | -0.000842337 | -0.032082019 |
| 309-315 | 0.0116463364 | 0.0064407602 | 0.0284219378 | -0.032082019 | 0.0063406029 | 0.0173374759 | -0.049337236 |
| 316-322 | -0.032082019 | 0.0357556093 | -0.058549187 | -0.021706111 | -0.032082019 | 0.0231203992 | -0.004681609 |
| 323-329 | -0.019495190 | -0.028879299 | -0.058357287 | -0.066634103 | 0.0180588777 | -0.028879299 | 0.0017794999 |
| 330-336 | -0.037588545 | -0.014952724 | -0.028879299 | -0.057708655 | -0.081437981 | 0.0479136550 | -0.028879299 |
| 337-343 | -0.071697989 | -0.079307859 | -0.043416760 | -0.018176635 | 0.0451212873 | 0.0175116308 | -0.014971321 |
| 344-350 | -0.018176635 | -0.013344877 | -0.073114598 | 0.0056458910 | -0.018176635 | 0.0066881320 | 0.0321587329 |
| 351-357 | -0.001167397 | -0.018176635 | 0.0121206521 | 0.0062321540 | -0.090709705 | -0.004305760 | 0.1011821731 |
| 358-364 | 0.0243999131 | 0.0446754056 | -0.004305760 | -0.001463461 | 0.0666791104 | 0.0171338554 | -0.004305760 |
| 365-371 | 0.0616624066 | 0.0316680163 | 0.0123782921 | -0.004305760 | 0.0141000480 | -0.016812908 | 0.0372747383 |
| 372-378 | 0.0004067974 | -0.064007456 | -0.030448078 | -0.033824773 | 0.0004067974 | 0.0063825983 | 0.0007076618 |
| 379-385 | 0.0180931236 | 0.0004067974 | 0.0154180690 | -0.063647504 | 0.0742728157 | 0.0004067974 | -0.036192114 |
| 386-392 | -0.020636704 | 0.0274009800 | -0.008398847 | 0.0054527826 | 0.0432034852 | -0.075624464 | -0.008398847 |

Table A.17: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 8 - continued

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 393-399 | -0.006395193 | -0.014756529 | -0.000363015 | -0.008398847 | 0.0504314067 | 0.0280316807 | -0.004560974 |
| 400-406 | -0.008398847 | 0.0312452014 | -0.037170200 | -0.042368064 | -0.003787615 | 0.0412695607 | 0.0352747212 |
| 407-413 | -0.081341950 | -0.003787615 | 0.0173490435 | 0.0365665604 | 0.0262532596 | -0.003787615 | -0.028056202 |
| 414-420 | -0.003085579 | 0.0330553991 | -0.003787615 | -0.003463606 | -0.040403875 | -0.048076907 | -0.029107124 |
| 421-427 | -0.007945668 | -0.027789074 | -0.026010814 | -0.029107124 | -0.020892925 | 0.0080453415 | -0.003823329 |
| 428-434 | -0.029107124 | 0.0494402999 | -0.039823601 | -0.005198972 | -0.029107124 | 0.0362646087 | 0.0162080068 |
| 435-441 | -0.029241624 | -0.007505781 | -0.044027685 | -0.064031823 | 0.0594168611 | -0.007505781 | 0.0566262032 |
| 442-448 | -0.033949454 | 0.0009446051 | -0.007505781 | 0.0005586867 | -0.056091163 | 0.0205122470 | -0.007505781 |
| 449-455 | -0.047715131 | -0.049509351 | -0.025083068 | -0.031102131 | -0.010233773 | 0.0562551916 | -0.018482794 |
| 456-462 | -0.031102131 | -0.009988744 | 0.0500934972 | 0.0598197845 | -0.031102131 | -0.027062299 | 0.0009455901 |
| 463-469 | -0.065529403 | -0.031102131 | -0.016431224 | 0.0613335016 | 0.0188191392 | 0.0105558957 | -0.007095153 |
| 470-476 | -0.032247353 | 0.0173825770 | 0.0105558957 | 0.0585483219 | -0.002384773 | 0.0393915438 | 0.0105558957 |
| 477-483 | 0.0236102492 | 0.0686689763 | -0.006585946 | 0.0105558957 | -0.000698495 | -0.011078834 | 0.0121452411 |
| 484-490 | -0.011752235 | 0.0033639610 | -0.042441913 | 0.0351529038 | -0.011752235 | 0.0034552982 | -0.012833339 |
| 491-497 | -0.044986646 | -0.011752235 | 0.0153029551 | -0.015329211 | -0.016281934 | -0.011752235 | -0.017649061 |
| 498-504 | -0.047214677 | 0.0346442794 | -0.017767475 | 0.0350874464 | -0.034794209 | 0.0354625625 | -0.017767475 |
| 505-511 | 0.0118789314 | -0.018193700 | -0.020687788 | -0.017767475 | 0.0445971052 | -0.027760939 | 0.0058567913 |
| 512-518 | -0.017767475 | -0.008494772 | 0.0160590037 | 0.0374419576 | -0.026362954 | 0.0021810585 | 0.0316249374 |
| 519-525 | -0.034138229 | -0.026362954 | 0.0024086406 | 0.0077871321 | 0.0302577470 | -0.026362954 | 0.0250837333 |
| 526-532 | 0.0302800256 | -0.016219817 | -0.026362954 | -0.000990879 | 0.0555047612 | -0.004622566 | -0.003347788 |
| 533-539 | 0.0766834663 | -0.052841009 | 0.0048765010 | -0.003347788 | 0.0730241583 | 0.0146902590 | 0.0322354617 |
| 540-546 | -0.003347788 | 0.0352850873 | -0.017249525 | 0.0858184826 | -0.003347788 | 0.1331622503 | -0.002854800 |
| 547-553 | -0.001085515 | -0.006340207 | -0.012780999 | -0.055875040 | 0.0260528729 | -0.006340207 | -0.014325711 |
| 554-560 | 0.0117808461 | 0.0506588955 | -0.006340207 | -0.021149445 | -0.077573243 | 0.0549669437 | -0.006340207 |
| 561-567 | -0.047676709 | 0.0152999345 | 0.0305724468 | -0.048211266 | -0.035375730 | -0.014178756 | -0.070288304 |
| 568-574 | -0.048211266 | -0.047730354 | -0.027263893 | 0.0616468470 | -0.048211266 | 0.0162430302 | -0.018956183 |
| 575-581 | -0.029579042 | -0.048211266 | 0.0168573617 | -0.043278984 | 0.0026773599 | -0.013013828 | 0.0286827618 |
| 582-588 | 0.0424181437 | 0.0029277956 | -0.013013828 | 0.0301111795 | 0.0482434490 | 0.0237914680 | -0.013013828 |
| 589-595 | 0.0026035738 | 0.0171578727 | -0.046679283 | -0.013013828 | 0.0344276441 | 0.0472125430 | 0.0027447869 |
| 596-602 | -0.007012336 | 0.0672672555 | 0.0304062586 | 0.0371030186 | -0.007012336 | 0.0196785511 | 0.0195819498 |
| 603-609 | -0.063083371 | -0.007012336 | -0.025598580 | -0.055701421 | -0.045560735 | -0.007012336 | 0.0403532244 |
| 610-616 | 0.0032651882 | 0.0203289826 | 0.0020613966 | 0.0386025852 | 0.0393660547 | 0.0601063528 | 0.0020613966 |
| 617-623 | -0.014914993 | -0.055360890 | -0.053843305 | 0.0020613966 | 0.0364604074 | -0.073518650 | -0.013853791 |
| 624-630 | 0.0020613966 | 0.0047223400 | 0.0527751362 | 0.0096750586 | -0.021392411 | 0.0172663447 | 0.0045688384 |
| 631-637 | -0.011041043 | -0.021392411 | -0.018535525 | -0.071133858 | -0.057233573 | -0.021392411 | 0.0960668785 |
| 638-644 | -0.053966883 | 0.0621067117 | -0.021392411 | -0.004706664 | -0.003042344 | -0.046445922 | -0.047047685 |
| 645-651 | 0.0207583504 | -0.020373942 | 0.0802325947 | -0.047047685 | 0.0622087371 | -0.090018085 | 0.0332078708 |
| 652-658 | -0.047047685 | 0.0268790806 | 0.0563456057 | 0.0014844438 | -0.047047685 | -0.022975678 | -0.024856232 |
| 659-665 | 0.0710547058 | -0.022019626 | -0.028714954 | -0.093163389 | 0.0392875240 | -0.022019626 | 0.0230756131 |
| 666-672 | 0.0780169018 | 0.0627933940 | -0.022019626 | -0.034037846 | -0.013538038 | 0.0443901260 | -0.022019626 |
| 673-679 | -0.070397827 | -0.040123433 | 0.0716563665 | -0.036705830 | 0.0450238497 | 0.0361167755 | -0.021534401 |
| 680-688 | -0.036705830 | 0.0410458221 | 0.0179997822 | 0.0323303290 | -0.036705830 | -0.038039713 | -0.048849295 |
| 687-693 | -0.078909428 | -0.036705830 | 0.0998381105 | 0.0281056016 | -0.010281989 | 0.0086486833 | -0.005620891 |
| 694-700 | 0.0318959695 | -0.020961423 | 0.0086486833 | 0.0073148008 | -0.003494781 | -0.039007879 | 0.0086486833 |
| 701-707 | 0.0075348494 | -0.035880188 | -0.079359984 | 0.0086486833 | 0.0041578585 | 0.0415175727 | -0.054604461 |
| 708-714 | 0.0208828948 | 0.0130256345 | -0.020979598 | 0.0082934157 | 0.0208828948 | 0.0197690609 | -0.004171224 |
| 715-721 | -0.057917395 | 0.0208828948 | 0.0315286021 | 0.0282654204 | -0.016617354 | 0.0208828948 | 0.0082858007 |
| 722-728 | -0.005909946 | -0.026530082 | 0.0208828948 | 0.0269263394 | -0.025182245 | -0.020237856 | 0.0208828948 |
| 729-735 | 0.0315286021 | 0.0187519610 | -0.022505255 | 0.0208828948 | 0.0308659687 | 0.0007098548 | -0.024721629 |
| 736-742 | 0.0208828948 | 0.0191029024 | -0.013928037 | -0.048168162 | 0.0270527261 | 0.0934555982 | 0.0046247496 |
| 743-749 | 0.0484935438 | 0.0270527261 | 0.0374620102 | -0.005956860 | 0.0008305672 | 0.0270527261 | 0.0225519413 |
| 750-756 | -0.034521715 | 0.0360828566 | 0.0270527261 | 0.0648489236 | -0.031523817 | 0.0013608996 | 0.0021979969 |
| 757-763 | 0.0286582657 | 0.0613988105 | 0.0494053497 | 0.0021979969 | 0.0033391442 | -0.035245491 | -0.001701171 |
| 764-770 | 0.0021979969 | 0.0164059444 | 0.0367227883 | -0.003762473 | 0.0021979969 | -0.008501176 | 0.0243518057 |
| 771-777 | 0.0026257811 | -0.041259089 | -0.072276148 | -0.064837299 | 0.0233145784 | -0.041259089 | -0.050439191 |
| 778-784 | -0.006734297 | -0.060981795 | -0.041259089 | -0.025294537 | -0.049085737 | 0.0121496060 | -0.041259089 |

Table A.18: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 9

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1-7 | 0.0324303854 | 0.0172552374 | 0.0224786435 | 0.0158723547 | 0.0575442802 | -0.074250510 | -0.009091056 |
| 8-14 | 0.0158723547 | 0.0095530521 | -0.022745471 | 0.0633220479 | 0.0158723547 | -0.034981456 | -0.003428934 |
| 9-21 | -0.001378973 | 0.0158723547 | 0.0497569194 | -0.078583991 | -0.031984863 | -0.001103885 | -0.019737224 |
| 22-28 | -0.005394864 | 0.0127638200 | -0.001103885 | -0.006361577 | 0.0026954103 | -0.038519960 | -0.001103885 |
| 29-35 | 0.0034381409 | 0.0151069754 | 0.0314386947 | -0.001103885 | -0.030852668 | -0.033567490 | -0.003904762 |
| 36-42 | 0.0000409095 | 0.0162751166 | -0.037321637 | -0.034652869 | 0.0000509095 | -0.005252107 | 0.0300741598 |
| 43-49 | 0.0242256503 | 0.0000409095 | 0.0131051471 | 0.0492731196 | 0.0028173786 | 0.0000409095 | 0.0158911444 |
| 50-56 | -0.061952951 | -0.033665962 | -0.014118177 | 0.0099408673 | -0.025556716 | -0.068683251 | -0.014118177 |
| 57-63 | -0.065333053 | -0.025976033 | -0.012916588 | -0.014118177 | -0.014909230 | 0.0035448110 | -0.018172437 |
| 64-70 | -0.014118177 | 0.0118239711 | -0.007595119 | -0.050771237 | 0.0015011825 | -0.008020876 | 0.0070895440 |
| 71-77 | -0.045187788 | 0.0015011825 | -0.010469043 | -0.009900068 | 0.0149093800 | 0.0015011825 | 0.0382235150 |
| 78-84 | -0.044977691 | 0.0300811064 | 0.0015011825 | 0.0254604922 | 0.0395306275 | 0.0242638327 | 0.0281918058 |
| 85-91 | 0.0110514388 | -0.020791185 | 0.0505405330 | 0.0281918058 | 0.0255158544 | -0.061011606 | 0.0273650170 |
| 92-98 | 0.0281918058 | -0.055566541 | -0.026124410 | 0.0393522688 | 0.0281918058 | -0.006437450 | -0.030269101 |
| 99-105 | 0.0674872008 | 0.0116664267 | -0.027436947 | -0.009336907 | -0.008812688 | 0.0116664267 | 0.0255144905 |
| 106-112 | -0.029072942 | -0.006893357 | 0.0116664267 | 0.0608001614 | 0.0104698631 | 0.0027137611 | 0.0116664267 |
| 113-119 | 0.0212825844 | 0.0059883873 | 0.0851272242 | 0.0293280943 | -0.000066835 | 0.0216176235 | 0.0314564114 |
| 120-126 | 0.0293280943 | -0.032918528 | -0.035418678 | -0.036101783 | 0.0293280943 | -0.008404081 | -0.021901953 |
| 127-133 | -0.048321407 | 0.0293280943 | -0.060928535 | -0.001072259 | 0.0102659082 | -0.013871489 | -0.021795336 |
| 134-140 | 0.0281072915 | 0.0381868122 | -0.013871489 | -0.024814005 | -0.002568735 | -0.051876838 | -0.013871489 |
| 141-147 | -0.014693275 | 0.0481962853 | 0.0761303820 | -0.013871489 | -0.007044564 | 0.0075359532 | 0.0889406477 |
| 148-154 | -0.001432991 | -0.055723338 | -0.004764073 | -0.074761119 | -0.001432991 | -0.026133444 | 0.0627664615 |
| 155-161 | 0.0884399144 | -0.001432991 | 0.0288286603 | 0.0636831229 | 0.0280094633 | -0.001432991 | -0.018156399 |
| 162-168 | 0.0039177914 | -0.013267971 | -0.006570424 | -0.080446235 | 0.0015195037 | 0.0257580193 | -0.006570424 |
| 169-175 | -0.040466122 | -0.030414633 | 0.0378897286 | -0.006570424 | 0.0241776457 | -0.033796436 | 0.0650247551 |
| 176-182 | -0.006570424 | -0.095055650 | -0.047061096 | -0.026127078 | -0.016863094 | 0.0042955110 | -0.056978167 |
| 183-189 | -0.049367711 | -0.016863094 | 0.0174246956 | 0.0046351198 | 0.0688960602 | -0.016863094 | 0.0844855410 |
| 190-196 | 0.0548847748 | 0.0022820122 | -0.016863094 | 0.0587853994 | 0.0231263838 | 0.0080504036 | 0.0139794996 |
| 197-203 | -0.027266954 | 0.0290042522 | 0.0376479343 | 0.0139794996 | 0.0662278655 | 0.0130077519 | 0.0547558340 |
| 204-210 | 0.0139794996 | 0.0589737780 | -0.015745045 | 0.0187845095 | 0.0139794996 | -0.019070561 | -0.008663167 |
| 211-217 | -0.014407868 | 0.0116664267 | 0.0702947054 | -0.013063610 | 0.0083742362 | 0.0116664267 | 0.0207588107 |
| 218-224 | -0.034610750 | -0.029857774 | 0.0116664267 | -0.015472983 | 0.0074304709 | 0.0077143929 | 0.0116664267 |
| 225-231 | -0.037672055 | -0.042375411 | -0.018994660 | -0.003816497 | -0.018855589 | 0.0059539852 | -0.043976675 |
| 232-238 | -0.003816497 | 0.0158647600 | 0.0428444289 | 0.0423434193 | -0.003816497 | 0.0285639310 | 0.0495379724 |
| 239-245 | -0.028504669 | -0.003816497 | -0.039060825 | -0.047217099 | -0.047404452 | -0.020426991 | 0.0449147222 |
| 246-252 | 0.0345934060 | 0.0773810550 | -0.020426991 | 0.0696002754 | 0.0371289351 | -0.029562985 | -0.020426991 |
| 253-259 | -0.018653450 | 0.0513196047 | 0.0595627379 | -0.020426991 | -0.019290990 | -0.007617689 | 0.0558899488 |
| 260-266 | -0.045013314 | -0.034381385 | 0.0080821699 | -0.019523623 | -0.045013314 | 0.0018817300 | 0.0536616919 |
| 267-273 | 0.0411715732 | -0.045013314 | -0.061453106 | 0.0053516500 | -0.038385050 | -0.045013314 | 0.0042049419 |
| 274-280 | 0.0553920582 | 0.0360516343 | -0.029679457 | 0.0093269815 | -0.069202081 | 0.0458465698 | -0.029679457 |
| 281-287 | -0.053746426 | -0.017213590 | -0.030670212 | -0.029679457 | -0.000144249 | -0.030833622 | -0.087481218 |
| 288-294 | -0.029679457 | 0.0470703299 | -0.056360699 | 0.0504730128 | 0.0098768463 | 0.0400207729 | -0.002093984 |
| 295-301 | 0.0485136962 | 0.0098768463 | 0.0134503602 | -0.013072297 | -0.042813653 | 0.0098768463 | -0.051918608 |
| 302-308 | 0.0123390589 | -0.011494694 | 0.0098768463 | 0.0842544743 | -0.067105295 | 0.0547975479 | -0.032038182 |
| 309-315 | -0.056535470 | -0.041843279 | -0.032920168 | -0.032038182 | 0.0369431559 | -0.011543680 | 0.0156149428 |
| 316-322 | -0.032038182 | -0.002101499 | -0.064931699 | -0.054205243 | -0.032038182 | 0.0377862106 | -0.000249569 |
| 323-329 | -0.023256348 | 0.0068734871 | 0.0125286589 | 0.0012963004 | 0.0029214534 | 0.0068734871 | -0.001989704 |
| 330-336 | -0.081188287 | -0.036909431 | 0.0068734871 | 0.0269448172 | 0.0120913818 | 0.0020539179 | 0.0068734871 |
| 337-343 | 0.0110844075 | 0.0017882500 | 0.0573883581 | -0.011382028 | 0.0187661801 | -0.044533471 | -0.019773509 |
| 344-350 | -0.011382028 | -0.119496681 | -0.017244536 | 0.0748692454 | -0.011382028 | -0.035828720 | 0.0089846806 |
| 351-357 | -0.023664231 | -0.011382028 | -0.097233640 | 0.0466385384 | -0.081996598 | -0.008770603 | 0.1011373349 |
| 358-364 | 0.0351622573 | 0.0562032886 | -0.008770603 | -0.077983199 | 0.1053352472 | 0.0572269854 | -0.008770603 |
| 365-371 | 0.0254839197 | 0.0857447628 | 0.0393752941 | -0.008770603 | 0.0648164864 | -0.012610249 | 0.0960892444 |
| 372-378 | -0.008577142 | -0.039133783 | -0.038720860 | 0.0310437586 | -0.008577142 | -0.039672380 | -0.014304718 |
| 379-385 | 0.0135072855 | -0.008577142 | -0.077776648 | 0.0142423053 | 0.1269930308 | -0.008577142 | -0.006771364 |
| 386-392 | -0.058612117 | -0.055412765 | -0.022058765 | 0.0325973962 | 0.0261247538 | -0.033363391 | -0.022058765 |

Table A.19: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 9 - continued

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 393-399 | -0.018361694 | -0.002769767 | 0.1463024985 | -0.022058765 | -0.052088164 | 0.0862149871 | 0.0019401132 |
| 400-406 | -0.022058765 | 0.0365656486 | 0.0288040591 | 0.0037346372 | -0.005889241 | 0.0235344872 | 0.0464560331 |
| 407-413 | 0.0806952927 | -0.005889241 | -0.014881305 | 0.0336286980 | 0.0074208872 | -0.005889241 | -0.119810405 |
| 414-420 | -0.011304590 | 0.0541415084 | -0.005889241 | 0.0418252770 | 0.0350750831 | 0.0675993137 | -0.000855970 |
| 421-427 | 0.0109994059 | -0.057780890 | -0.053323558 | -0.000855970 | -0.064831013 | 0.0133380829 | 0.0607485442 |
| 428-434 | -0.000855970 | 0.0512962853 | -0.056976145 | -0.025907673 | -0.000855970 | -0.027999361 | -0.052050330 |
| 435-441 | -0.018816026 | 0.0068734871 | -0.022483339 | -0.000162610 | -0.002739021 | 0.0068734871 | 0.0399945125 |
| 442-448 | -0.059599857 | -0.003998026 | 0.0068734871 | 0.0365396380 | -0.018334271 | 0.0047474718 | 0.0068734871 |
| 449-455 | -0.063562387 | -0.027723580 | -0.006155086 | -0.006750612 | -0.043524865 | 0.0282283857 | 0.0431346790 |
| 456-462 | -0.006750612 | -0.027077834 | 0.0344577549 | 0.1263957474 | -0.006750612 | -0.008855428 | 0.0202079384 |
| 463-469 | 0.0044768358 | -0.006750612 | -0.036546046 | 0.0185789130 | -0.006493623 | 0.0123677136 | -0.048181075 |
| 470-476 | 0.0384838861 | 0.0655188611 | 0.0123677136 | 0.0007414878 | 0.0573624367 | 0.0976972500 | 0.0123677136 |
| 477-483 | 0.0155349426 | 0.0522841865 | 0.0301227534 | 0.0123677136 | -0.108151072 | 0.0337823377 | 0.0100158951 |
| 484-490 | -0.012720430 | -0.053459961 | 0.0349292884 | 0.0892244141 | -0.012720430 | -0.055660736 | -0.027701259 |
| 491-497 | 0.0595083520 | -0.012720430 | -0.012149215 | 0.1037638110 | -0.002914794 | -0.012720430 | -0.047527043 |
| 498-504 | 0.0694309875 | 0.0491678610 | -0.011022660 | -0.066920184 | 0.0968659433 | 0.0506959719 | -0.011022660 |
| 505-511 | -0.018109023 | 0.0587016568 | 0.0164975471 | -0.011022660 | 0.0200607170 | 0.0421056173 | -0.032754538 |
| 512-518 | -0.011022660 | -0.032159352 | 0.1004764450 | 0.0252251785 | -0.003475020 | -0.048812219 | 0.0165711709 |
| 519-525 | 0.0448252606 | -0.003475020 | 0.0190564032 | 0.1133205968 | 0.0712670911 | -0.003475020 | -0.089510655 |
| 526-532 | 0.0560286014 | 0.0649533416 | -0.003475020 | -0.027401227 | 0.0275205339 | -0.003881951 | 0.0078226248 |
| 533-539 | 0.0362382783 | -0.043030407 | -0.019710954 | 0.0078226248 | -0.048979801 | 0.0109522585 | 0.0934334985 |
| 540-546 | 0.0078226248 | 0.0310767439 | -0.036554912 | 0.1676665255 | 0.0078226248 | 0.0394175655 | -0.035260122 |
| 547-553 | -0.009017287 | 0 | 0.0119966996 | -0.024710030 | -0.002126015 | 0 | -0.068366913 |
| 554-560 | 0.0296520906 | 0.1289808736 | 0 | -0.018441680 | 0.0411685360 | 0.0233537317 | 0 |
| 561-567 | -0.087254748 | 0.0117535455 | 0.1039876329 | -0.029563248 | -0.008131570 | 0.0334289655 | 0.0168611799 |
| 568-574 | -0.029563248 | -0.066308987 | -0.017458841 | 0.0626325961 | -0.029563248 | 0.0298012450 | 0.0309922184 |
| 575-581 | 0.0689170702 | -0.029563248 | -0.011439162 | -0.010336480 | 0.0686515892 | -0.048778265 | -0.037913075 |
| 582-588 | 0.0825791122 | 0.0030872003 | -0.048778265 | -0.021343110 | 0.0012499811 | 0.0865379054 | -0.048778265 |
| 589-595 | -0.059759613 | 0.0279784437 | -0.019302869 | -0.048778265 | -0.048203784 | 0.0265024060 | 0.0040128565 |
| 596-602 | -0.025571883 | 0.0002146784 | 0.0058053330 | 0.0240329146 | -0.025571883 | -0.072935793 | -0.024952620 |
| 603-609 | -0.009808368 | -0.025571883 | -0.017768646 | -0.037423604 | 0.0069128250 | -0.025571883 | 0.0119586593 |
| 610-616 | -0.013106967 | 0.0504771631 | -0.022189439 | -0.015972801 | 0.0602998028 | 0.0496119295 | -0.022189439 |
| 617-623 | -0.013404578 | -0.036405816 | -0.007962077 | -0.022189439 | -0.053967697 | -0.019508674 | -0.034303615 |
| 624-630 | -0.022189439 | -0.050427192 | 0.0617270225 | 0.0213923733 | -0.034609067 | -0.048969466 | 0.0318476331 |
| 631-637 | 0.0168114740 | -0.034609067 | -0.045818824 | -0.040662853 | -0.072699451 | -0.034609067 | 0.0508563389 |
| 638-644 | 0.0293402156 | 0.0859631964 | -0.034609067 | -0.060033159 | 0.0611900372 | 0.0102968043 | -0.049803761 |
| 645-651 | -0.016921707 | 0.0385726105 | 0.0767222254 | -0.049803761 | 0.0048102113 | -0.033232192 | 0.0218209077 |
| 652-658 | -0.049803761 | -0.017907585 | 0.1043600320 | -0.042272133 | -0.049803761 | -0.045686639 | 0.0185432721 |
| 659-665 | 0.1032713290 | -0.005660475 | 0.0008013251 | 0.0008105987 | 0.0176932566 | -0.005660475 | 0.0337480632 |
| 666-672 | 0.0771396296 | 0.0105379630 | -0.005660475 | -0.085303770 | -0.030194602 | -0.010726646 | -0.005660475 |
| 673-679 | -0.002640432 | -0.016778348 | 0.0875046943 | -0.009207442 | 0.0724540958 | 0.0790636425 | 0.1191978203 |
| 680-688 | -0.009207442 | -0.023985464 | -0.073871533 | 0.0333467996 | -0.009207442 | 0.0353184259 | -0.006247974 |
| 687-693 | -0.135109807 | -0.009207442 | 0.0549227922 | 0.0552590754 | 0.0802683749 | 0.0028657396 | -0.056381828 |
| 694-700 | -0.023642553 | 0.0413376569 | 0.0028657396 | 0.0581667637 | 0.0030435236 | -0.010622634 | 0.0028657396 |
| 701-707 | 0.0127849981 | -0.015871194 | -0.079977372 | 0.0028657396 | -0.074331821 | -0.017021860 | -0.026302943 |
| 708-714 | -0.002934700 | 0.0017183382 | 0.0179653832 | 0.0037021394 | -0.002934700 | 0.0134647521 | 0.0094114657 |
| 715-721 | -0.051344013 | -0.002934700 | -0.002825833 | -0.045458553 | -0.031894313 | -0.002934700 | -0.017629514 |
| 722-728 | 0.0423783719 | -0.020364997 | -0.014159762 | -0.034239528 | -0.010855892 | -0.032805176 | -0.014159762 |
| 729-735 | 0.0141668769 | -0.028465843 | -0.050680318 | -0.014159762 | -0.004463291 | -0.038520962 | -0.016963780 |
| 736-742 | -0.014159762 | -0.062396009 | -0.016476282 | -0.033711581 | 0.0047161138 | 0.0994453359 | 0.0010164914 |
| 743-749 | 0.1004608197 | 0.0047161138 | 0.0000923519 | -0.017308845 | -0.039226147 | 0.0047161138 | 0.0297042532 |
| 750-756 | -0.074285170 | -0.043313572 | 0.0047161138 | 0.0709481426 | 0.0090650743 | 0.0463789303 | 0.0085818221 |
| 757-763 | 0.0255947902 | 0.1008156675 | -0.008484093 | 0.0085818221 | 0.0125304604 | -0.084315531 | -0.057877242 |
| 764-770 | 0.0085818221 | 0.0490965689 | -0.035631469 | -0.072784168 | 0.0085818221 | 0.0283088642 | 0.0917982807 |
| 771-777 | 0.0403374374 | 0 | -0.000536173 | -0.028266696 | -0.005066171 | 0 | 0.0196628050 |
| 778-784 | -0.016216999 | -0.080456650 | 0 | -0.023755627 | 0.0181479983 | 0.0077143929 | 0 |

Table A.20: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 10

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1-7 | 0.0215230892 | 0.0691822156 | -0.023296600 | -0.007372382 | 0.0024177617 | 0.0037602245 | -0.019413409 |
| 8-14 | -0.007372382 | 0.0019399191 | 0.0667314704 | -0.059862574 | -0.007372382 | -0.064237395 | 0.0225723348 |
| 9-21 | -0.040150093 | -0.007372382 | -0.013289393 | 0.0133240899 | 0.0058655593 | 0.0156058990 | -0.000885657 |
| 22-28 | 0.0108096206 | 0.0134578601 | 0.0156058990 | -0.032279413 | 0.0480142234 | -0.029955890 | 0.0156058990 |
| 29-35 | 0.0003357855 | 0.0823098770 | -0.010591053 | 0.0156058990 | -0.061292810 | -0.014481161 | -0.040541508 |
| 36-42 | -0.039094804 | -0.014289078 | 0.0070789542 | -0.050339393 | -0.039094804 | 0.0083324285 | 0.0324851472 |
| 43-49 | 0.0006458698 | -0.039094804 | 0.0103310062 | -0.013441625 | -0.011303859 | -0.039094804 | 0.0060303151 |
| 50-56 | -0.033526261 | -0.015559904 | -0.018775333 | 0.0579019846 | 0.0151838143 | -0.021274511 | -0.018775333 |
| 57-63 | 0.0269196514 | 0.0055432294 | -0.046502407 | -0.018775333 | 0.0090145484 | 0.0249602584 | 0.0317067415 |
| 64-70 | -0.018775333 | 0.0737518056 | 0.0342841092 | 0.0088165631 | -0.017736051 | -0.020678760 | -0.022153185 |
| 71-77 | -0.003391419 | -0.017736051 | -0.004821807 | -0.006067698 | -0.026857843 | -0.017736051 | -0.036093859 |
| 78-84 | 0.0094015091 | 0.0167945490 | -0.017736051 | 0.0618847471 | 0.0624159156 | 0.0441621758 | 0.0384181385 |
| 85-91 | -0.004605855 | -0.003942708 | -0.012081384 | 0.0384181385 | 0.0041694731 | 0.0301089029 | 0.0720061609 |
| 92-98 | 0.0384181385 | -0.001779483 | -0.010874686 | 0.0295716997 | 0.0384181385 | -0.009823724 | 0.0141455554 |
| 99-105 | 0.0032927961 | 0.0422447876 | 0.0465676598 | 0.0376686002 | -0.016514345 | 0.0422447876 | -0.027885040 |
| 106-112 | 0.0353062852 | -0.007518928 | 0.0422447876 | 0.0816549270 | 0.0578218008 | 0.0058003625 | 0.0422447876 |
| 113-119 | 0.0315664282 | 0.0950001512 | -0.031825369 | 0.0040956179 | -0.026747452 | 0.0696021414 | -0.025438225 |
| 120-126 | 0.0040956179 | 0.0317230475 | 0.1106119706 | -0.016880238 | 0.0040956179 | -0.014931956 | 0.0132228354 |
| 127-133 | -0.065928660 | 0.0040956179 | -0.050519425 | 0.0349834390 | -0.018254398 | 0.0061274824 | 0.0332583044 |
| 134-140 | 0.0488063917 | 0.0606507504 | 0.0061274824 | -0.011269913 | 0.0322735434 | -0.081941604 | 0.0061274824 |
| 141-147 | -0.052338914 | 0.0541459445 | 0.0102007342 | 0.0061274824 | -0.055518665 | -0.044052567 | -0.020986449 |
| 148-154 | -0.073611427 | 0.0139567762 | 0.0171708642 | -0.054113121 | -0.073611427 | -0.084282379 | -0.021574999 |
| 155-161 | -0.038497553 | -0.073611427 | -0.042967704 | 0.0425437322 | -0.003052344 | -0.073611427 | 0.0425886963 |
| 162-168 | 0.0500636675 | 0.0026012260 | -0.006105696 | 0.0178644110 | 0.0278621126 | 0.0061359941 | -0.006105696 |
| 169-175 | -0.006698674 | 0.1204230453 | 0.0250456722 | -0.006105696 | -0.050748567 | -0.073929157 | 0.0005751624 |
| 176-182 | -0.006105696 | 0.0141801214 | 0.0105150163 | -0.025763154 | -0.040321228 | 0.0043939537 | 0.0215473840 |
| 183-189 | -0.026509040 | -0.040321228 | -0.035924035 | -0.019916153 | -0.001156289 | -0.040321228 | 0.0088613871 |
| 190-196 | -0.050821697 | 0.0300601024 | -0.040321228 | 0.0535322529 | 0.0591906110 | 0.0249009460 | 0.0071034696 |
| 197-203 | -0.038900408 | -0.011949809 | -0.024761449 | 0.0071034696 | 0.0133415518 | -0.033481364 | 0.1178014361 |
| 204-210 | 0.0071034696 | 0.0087897351 | -0.035162515 | -0.013598433 | 0.0071034696 | 0.0065120294 | 0.0338627856 |
| 211-217 | 0.0035786569 | 0.0311066382 | 0.0879764368 | 0.0675421726 | -0.013291006 | 0.0311066382 | 0.0008613578 |
| 218-224 | -0.022561443 | -0.022118028 | 0.0311066382 | 0.0863400480 | 0.0625026861 | -0.049436459 | 0.0311066382 |
| 225-231 | -0.023825358 | 0.0529629783 | -0.028863954 | -0.025730548 | -0.037516046 | 0.0036774964 | -0.077776663 |
| 232-238 | -0.025730548 | 0.0061632537 | 0.0465621698 | 0.0211067891 | -0.025730548 | -0.012242121 | 0.0077041255 |
| 239-245 | -0.013367603 | -0.025730548 | 0.0000078527 | 0.0706040764 | -0.040703126 | -0.026476308 | 0.0262313940 |
| 246-252 | 0.0307598146 | -0.038195335 | -0.026476308 | -0.047721523 | 0.0393544741 | 0.0101282453 | -0.026476308 |
| 253-259 | -0.042077179 | 0.0387721544 | 0.0449307868 | -0.026476308 | -0.049606080 | 0.0196836611 | -0.009214559 |
| 260-266 | -0.049428464 | 0.0419958589 | 0.1299166853 | 0.0630661658 | -0.049428464 | -0.002982153 | 0.0608558461 |
| 267-273 | 0.0094291502 | -0.049428464 | -0.012781120 | -0.005341514 | -0.019056641 | -0.049428464 | 0.0904330275 |
| 274-280 | 0.0587285449 | 0.1096367296 | -0.004127440 | 0.0513232299 | -0.043729385 | 0.0435441101 | -0.004127440 |
| 281-287 | -0.000581376 | 0.0198883021 | 0.0152665529 | -0.004127440 | 0.0300231172 | 0.0098856464 | 0.0081065669 |
| 288-294 | -0.004127440 | 0.0904703321 | 0.0044366120 | 0.0365596562 | 0.0288080821 | 0.0526210722 | -0.031227548 |
| 295-301 | 0.0543619568 | 0.0288080821 | 0.0257034578 | 0.0099439464 | 0.0336873945 | 0.0288080821 | -0.023081782 |
| 302-308 | -0.059015815 | 0.0134529731 | 0.0288080821 | 0.0411790479 | -0.007589352 | 0.0465766225 | 0.0388005647 |
| 309-315 | -0.024505144 | 0.0135595776 | -0.030016358 | 0.0388005647 | -0.014569699 | -0.044032531 | 0.0353381938 |
| 316-322 | 0.0388005647 | 0.0045783364 | -0.063022700 | 0.0042562145 | 0.0388005647 | -0.005319366 | -0.000326887 |
| 323-329 | 0.0079483955 | 0.0493485982 | 0.0857754709 | 0.0752272787 | -0.038471091 | 0.0493485982 | -0.014428094 |
| 330-336 | -0.072833641 | -0.006528234 | 0.0493485982 | 0.1065460550 | 0.1342573009 | -0.012948249 | 0.0493485982 |
| 337-343 | 0.0303360607 | 0.0785847700 | -0.000726693 | -0.008171679 | -0.091101815 | 0.0094919907 | -0.014948582 |
| 344-350 | -0.008171679 | -0.015205679 | 0.0457489376 | 0.0033235456 | -0.008171679 | -0.010389394 | 0.0135516423 |
| 351-357 | -0.013360536 | -0.008171679 | -0.113540825 | -0.005549736 | 0.0019809841 | -0.052603292 | -0.072200850 |
| 358-364 | 0.0148041245 | 0.0167754289 | -0.052603292 | -0.052355794 | -0.002925042 | -0.020207958 | -0.052603292 |
| 365-371 | -0.085921427 | 0.0212289273 | -0.003542809 | -0.052603292 | 0.0242843174 | 0.0271004621 | 0.0340097426 |
| 372-378 | 0.0002388442 | 0.0140458654 | -0.006982492 | -0.003470517 | 0.0002388442 | -0.073275817 | -0.000748243 |
| 379-385 | 0.0010485248 | 0.0002388442 | -0.098580969 | 0.0093689572 | 0.0066125432 | 0.0002388442 | 0.0501209412 |
| 386-392 | 0.0394297628 | -0.019967938 | 0.0093268099 | 0.0549209442 | 0.0328161851 | 0.0927223456 | 0.0093268099 |

Table A.21: SVM weights for the binary single-feature system on Caltech-256 - binary classifier 10 - continued

| Weight index | | | | | | | |
|---|---|---|---|---|---|---|---|
| 393-399 | -0.070753651 | 0.0025433272 | 0.0930758495 | 0.0093268099 | -0.049909359 | 0.0283550003 | 0.0412897672 |
| 400-406 | 0.0093268099 | 0.0535573722 | -0.013303572 | 0.0661961788 | 0.0161558651 | 0.0009630146 | -0.021745207 |
| 407-413 | 0.1082797122 | 0.0161558651 | -0.032990026 | 0.0554864765 | 0.0611857466 | 0.0161558651 | -0.037477607 |
| 414-420 | -0.024640181 | 0.0189643729 | 0.0161558651 | -0.038597115 | -0.036490081 | 0.1212611599 | 0.0199769401 |
| 421-427 | 0.0399422365 | -0.004488001 | 0.0110206531 | 0.0199769401 | -0.123061768 | -0.052716977 | 0.0593118809 |
| 428-434 | 0.0199769401 | -0.020790618 | -0.038029567 | -0.028155832 | 0.0199769401 | -0.056170190 | -0.070930470 |
| 435-441 | 0.0308239164 | 0.0008992509 | 0.0747060231 | 0.0910227103 | -0.052923321 | 0.0008992509 | -0.057340471 |
| 442-448 | -0.037220131 | -0.026796388 | 0.0008992509 | 0.0227610330 | 0.0094826600 | -0.066663823 | 0.0008992509 |
| 449-455 | -0.010014221 | -0.017282939 | 0.0463641308 | -0.012109591 | -0.026336844 | -0.008685482 | 0.0098291684 |
| 456-462 | -0.012109591 | 0.0949628515 | 0.0288854878 | -0.022554490 | -0.012109591 | -0.021080430 | -0.042668116 |
| 463-469 | 0.0430466460 | -0.012109591 | -0.010733912 | 0.0017411360 | -0.018340490 | -0.006967679 | -0.083257551 |
| 470-476 | -0.009781694 | 0.0395421840 | -0.006967679 | -0.013672133 | 0.0018640670 | -0.002266948 | -0.006967679 |
| 477-483 | -0.071002137 | -0.029542535 | -0.001766659 | -0.006967679 | 0.0142858922 | 0.0130877681 | -0.017979015 |
| 484-490 | 0.0121430436 | -0.038518870 | 0.0316633195 | 0.0643664227 | 0.0121430436 | -0.007349178 | 0.0351782909 |
| 491-497 | 0.0843430410 | 0.0121430436 | -0.030328146 | 0.0279698790 | 0.0033949231 | 0.0121430436 | -0.078855550 |
| 498-504 | 0.0513948392 | 0.0552527590 | 0.0105953588 | -0.020591425 | 0.0626328385 | -0.011416529 | 0.0105953588 |
| 505-511 | -0.019216485 | 0.0220804680 | 0.0277360763 | 0.0105953588 | -0.013606426 | -0.043260867 | -0.011433279 |
| 512-518 | 0.0105953588 | -0.007045754 | 0.0458859341 | -0.011734259 | 0.0101721828 | -0.058296741 | -0.048477687 |
| 519-525 | 0.0513022273 | 0.0101721828 | 0.0577671226 | -0.037567457 | 0.0124829577 | 0.0101721828 | -0.096366457 |
| 526-532 | 0.0466045862 | -0.008873766 | 0.0101721828 | -0.082160573 | -0.029768516 | 0.0340931398 | 0.0062112318 |
| 533-539 | -0.064013294 | -0.034936533 | -0.014144860 | 0.0062112318 | -0.072240335 | 0.0194637790 | -0.036454130 |
| 540-546 | 0.0062112318 | -0.001611032 | 0.0138048833 | 0.0337922872 | 0.0062112318 | -0.083703388 | 0.0091500164 |
| 547-553 | 0.0080928227 | -0.008960860 | 0.0029869121 | 0.0733808228 | -0.068505493 | -0.008960860 | 0.0330882871 |
| 554-560 | 0.0286365041 | 0.0118595165 | -0.008960860 | 0.0417733335 | 0.1040734024 | -0.041078870 | -0.008960860 |
| 561-567 | 0.1298353530 | 0.0465125604 | 0.0777260887 | 0.0199000385 | 0.0533196211 | 0.0323410669 | 0.0879744294 |
| 568-574 | 0.0199000385 | -0.019095792 | 0.0025809659 | -0.055385837 | 0.0199000385 | -0.004365801 | -0.028362923 |
| 575-581 | 0.0206851550 | 0.0199000385 | -0.058631362 | 0.0016673011 | 0.0126950061 | 0.0109482869 | -0.060374560 |
| 582-588 | 0.0233019494 | -0.039695160 | 0.0109482869 | 0.0281199375 | 0.0314597261 | 0.0359708982 | 0.0109482869 |
| 589-595 | -0.002544218 | -0.000347986 | 0.0699587094 | 0.0109482869 | -0.048390328 | 0.0162625777 | 0.0625964232 |
| 596-602 | -0.007346740 | -0.025669958 | 0.0119269201 | -0.002790319 | -0.007346740 | -0.013041427 | -0.029007320 |
| 603-609 | 0.1189591708 | -0.007346740 | 0.0548148344 | -0.012031235 | 0.0508806498 | -0.007346740 | 0.0013032749 |
| 610-616 | -0.006141256 | 0.0765884661 | 0.0252707236 | 0.0006896104 | 0.0356669973 | 0.0299773067 | 0.0252707236 |
| 617-623 | -0.007188120 | -0.029783397 | 0.0623059738 | 0.0252707236 | -0.005095138 | 0.0277870508 | -0.034028371 |
| 624-630 | 0.0252707236 | -0.049635741 | -0.000494471 | 0.0063892962 | 0.0181484387 | -0.049441108 | 0.0703414181 |
| 631-637 | 0.0325637873 | 0.0181484387 | 0.0040918359 | 0.0286535043 | -0.019156615 | 0.0181484387 | -0.034759303 |
| 638-644 | 0.0307392691 | 0.0592538631 | 0.0181484387 | -0.058481081 | 0.0367221256 | 0.0682414583 | 0.0620505076 |
| 645-651 | 0.0172294164 | 0.0178359535 | 0.0526362513 | 0.0620505076 | -0.037296847 | 0.0348248705 | 0.0882729462 |
| 652-658 | 0.0620505076 | -0.019335434 | 0.0359620704 | 0.0021152526 | 0.0620505076 | -0.008115558 | 0.0126314779 |
| 659-665 | -0.008442524 | -0.013234858 | 0.0513319166 | 0.1328482463 | -0.045352869 | -0.013234858 | -0.049539556 |
| 666-672 | 0.0097530162 | -0.039546718 | -0.013234858 | -0.030292677 | -0.001735090 | -0.041174066 | -0.013234858 |
| 673-679 | 0.0358328116 | 0.0082598463 | -0.059424354 | 0.0273898985 | 0.0053784278 | 0.0151155778 | 0.0343671363 |
| 680-688 | 0.0273898985 | -0.049413863 | -0.114008479 | -0.031422891 | 0.0273898985 | 0.0499423303 | -0.014256621 |
| 687-693 | 0.0470119116 | 0.0273898985 | -0.058142060 | -0.038193132 | 0.0177667224 | -0.032298092 | -0.013919330 |
| 694-700 | -0.040295081 | 0.0486624226 | -0.032298092 | 0.0027259319 | -0.075059845 | 0.0235527187 | -0.032298092 |
| 701-707 | 0.0039590743 | -0.018732089 | -0.004748929 | -0.032298092 | -0.035450649 | -0.005029193 | 0.0656724619 |
| 708-714 | -0.026637742 | -0.028653057 | 0.0167013774 | -0.022124595 | -0.026637742 | 0.0063559200 | -0.010582430 |
| 715-721 | 0.0347368501 | -0.026637742 | -0.031231619 | -0.046072211 | -0.050665260 | -0.026637742 | -0.048509883 |
| 722-728 | 0.0201557337 | -0.025831379 | -0.026443599 | -0.051907662 | 0.0277064365 | -0.071101411 | -0.026443599 |
| 729-735 | -0.022963952 | -0.074053106 | -0.076398680 | -0.026443599 | 0.0021005240 | 0.0025591375 | 0.0514378701 |
| 736-742 | -0.026443599 | -0.033755500 | 0.0254192054 | -0.049084506 | -0.022887903 | -0.104112518 | 0.0078299959 |
| 743-749 | 0.0273337923 | -0.022887903 | -0.007231004 | -0.029827806 | -0.012623385 | -0.022887903 | 0.0106963766 |
| 750-756 | -0.038283432 | -0.023877041 | -0.022887903 | -0.059906237 | -0.001706281 | 0.0642065315 | -0.020245915 |
| 757-763 | -0.090507078 | -0.001118822 | -0.049712175 | -0.020245915 | 0.0251532808 | 0.0175203041 | -0.016337844 |
| 764-770 | -0.020245915 | 0.0269436845 | -0.009465399 | -0.098420600 | -0.020245915 | -0.007292255 | 0.0655366762 |
| 771-777 | 0.0209857269 | 0.0325913547 | 0.0107765610 | 0.0362961210 | 0.0046521474 | 0.0325913547 | 0.0725731063 |
| 778-784 | 0.0504127078 | -0.045429629 | 0.0325913547 | -0.018593457 | 0.0837426428 | -0.011781491 | 0.0325913547 |

# References


[1] C. Kyrkou and T. Theocharides, "A flexible parallel hardware architecture for AdaBoost-based real-time object detection," *IEEE Transactions on Very Large Scale Integration (VLSI) systems*, vol. 19, no. 6, pp. 1034–1047, 2010.

[2] K. Negi, K. Dohi, Y. Shibata, and K. Oguri, "Deep pipelined one-chip FPGA implementation of a real-time image-based human detection algorithm," in *International Conference on Field-Programmable Technology*, IEEE, 2011, pp. 1–8.

[3] O. S. Al-Kadi, "Combined statistical and model based texture features for improved image classification," 2008.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Ieee, vol. 1, 2005, pp. 886–893.

[5] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[6] V. Vapnik, *Statistical Learning Theory. new york: John willey & sons*, 1998.

[7] Y. Wei, X. Bing, and C. Chareonsak, "FPGA implementation of AdaBoost algorithm for detection of face biometrics," in *IEEE International Workshop on Biomedical Circuits and Systems*, IEEE, 2004, pp. 1–6.

[8] J. Cho, S. Mirzaei, J. Oberg, and R. Kastner, "FPGA-based face detection system using Haar classifiers," in *Proceedings of the ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, ACM, 2009, pp. 103–112.

[9] M. Hiromoto and R. Miyamoto, "Hardware architecture for high-accuracy real-time pedestrian detection with CoHOG features," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, IEEE, 2009, pp. 894–899.

[10] Y. Yazawa, T. Yoshimi, T. Tsuzuki, *et al.*, "FPGA hardware with target-reconfigurable object detector," *IEICE TRANSACTIONS on Information and Systems*, vol. 98, no. 9, pp. 1637–1645, 2015.

[11] Q. Zhang, Y. Chen, Y. Zhang, and Y. Xu, "SIFT implementation and optimization for multi-core systems," in *2008 IEEE International Symposium on Parallel and Distributed Processing*, IEEE, 2008, pp. 1–8.

[12] C. He, A. Papakonstantinou, and D. Chen, "A novel SoC architecture on FPGA for ultra fast face detection," in *2009 IEEE International Conference on Computer Design*, IEEE, 2009, pp. 412–418.

[13] K. Tan, J. Zhang, Q. Du, and X. Wang, "GPU parallel implementation of support vector machines for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 10, pp. 4647–4656, 2015.

[14] Z. Wen, J. Shi, B. He, J. Chen, K. Ramamohanarao, and Q. Li, "Exploiting GPUs for efficient gradient boosting decision tree training," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2706–2717, 2019.

[15] M. Qasaimeh, A. Sagahyroon, and T. Shanableh, "FPGA-based parallel hardware architecture for real-time image classification," *IEEE Transactions on Computational Imaging*, vol. 1, no. 1, pp. 56–70, 2015.

[16] S. Jin, D. Kim, T. T. Nguyen, B. Jun, D. Kim, and J. W. Jeon, "An FPGA-based parallel hardware architecture for real-time face detection using a face certainty map," in *2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, IEEE, 2009, pp. 61–66.

[17] D. Berten, "GPU vs FPGA performance comparison," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays-FPGA'17*, 2016.

[18] H. M. Hussain, K. Benkrid, A. Ebrahim, A. T. Erdogan, and H. Seker, "Novel dynamic partial reconfiguration implementation of k-means clustering on FPGAs: Comparative results with GPPs and GPUs," *International Journal of Reconfigurable Computing*, vol. 2012, p. 1, 2012.

[19] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2012, pp. 47–56.

[20] H. Hussain, K. Benkrid, C. Hong, and H. Seker, "An adaptive FPGA implementation of multi-core k-nearest neighbour ensemble classifier using dynamic partial reconfiguration," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, IEEE, 2012, pp. 627–630.

[21] H. Hussain, K. Benkrid, and H. ŞEKER, "Novel dynamic partial reconfiguration implementations of the support vector machine classifier on FPGA," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 24, no. 5, pp. 3371–3387, 2016.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[23] S.-W. Chen and C.-S. Tao, "PolSAR image classification using polarimetric-feature-driven deep convolutional neural network," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 4, pp. 627–631, 2018.

[24] C. Zhang, I. Sargent, X. Pan, A. Gardiner, J. Hare, and P. M. Atkinson, "VPRS-based regional decision fusion of CNN and MRF classifications for very fine resolution remotely sensed images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 8, pp. 4507–4521, 2018.

[25] K. Abdelouahab, M. Pelcat, J. Serot, and F. Berry, "Accelerating CNN inference on FPGAs: A survey," *arXiv preprint arXiv:1806.01683*, 2018.

[26] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of FPGA-based neural network inference accelerators," *ACM Trans. on Reconfigurable Technology and Sys. (TRETS)*, vol. 12, no. 1, pp. 1–26, 2019.

[27] N. Attarmoghaddam and K. F. Li, "Data mining hardware acceleration for object detection," in *2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, IEEE, 2019, pp. 1–7.

[28] T. Kryjak, M. Komorkiewicz, and M. Gorgon, "FPGA implementation of real-time head-shoulder detection using local binary patterns, SVM and foreground object detection," in *Proceedings of the 2012 Conference on Design and Architectures for Signal and Image Processing*, IEEE, 2012, pp. 1–8.

[29] R. Kadota, H. Sugano, M. Hiromoto, H. Ochi, R. Miyamoto, and Y. Nakamura, "Hardware architecture for HOG feature extraction," in *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, IEEE, 2009, pp. 1330–1333.

[30] L. Zhang, W. Qu, Y. Guo, and S. Li, "A FPGA-based hardware architecture for real-time texture classification using local binary patterns," *DEStech Transactions on Engineering and Technology Research*, no. iceta, 2016.

[31] O. Mujahid, Z. Ullah, H. Mahmood, and A. Hafeez, "Fast pattern recognition through an LBP driven CAM on FPGA," *IEEE Access*, vol. 6, pp. 39 525–39 531, 2018.

[32] Y. Wei, Q. Tian, and T. Guo, "An improved pedestrian detection algorithm integrating Haar-like features and HOG descriptors," *Advances in Mechanical Engineering*, vol. 5, p. 546 206, 2013.

[33] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland, vol. 12, 1995, pp. 296–301.

[34] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[35] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution grayscale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[36] H. Madadum and Y. Becerikli, "The implementation of support vector machine (SVM) using FPGA for human detection," in *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*, IEEE, 2017, pp. 1286–1290.

[37] S. Bauer, S. Köhler, K. Doll, and U. Brunsmann, "FPGA-GPU architecture for kernel SVM pedestrian detection," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, IEEE, 2010, pp. 61–68.

[38] M. Tian, X. Zhang, Z. Yang, J. Huang, H. Chen, *et al.*, "The implementation of a KNN classifier on FPGA with a parallel and pipelined architecture based on predetermined range search," in *13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, IEEE, 2016, pp. 1491–1493.

[39] D. Nguyen, D. Halupka, P. Aarabi, and A. Sheikholeslami, "Real-time face detection and lip feature extraction using field-programmable gate arrays," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 4, pp. 902–912, 2006.

[40] T.-W. Chen and S.-Y. Chien, "Bandwidth adaptive hardware architecture of k-means clustering for video analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) systems*, vol. 18, no. 6, pp. 957–966, 2009.

[41] D. Mahmoodi, A. Soleimani, H. Khosravi, M. Taghizadeh, *et al.*, "FPGA simulation of linear and nonlinear support vector machine," *Journal of Software Engineering and Applications*, vol. 4, no. 05, p. 320, 2011.

[42] R. A. Patil, G. Gupta, V. Sahula, and A. S. Mandal, "Power aware hardware prototyping of multiclass SVM classifier through reconfiguration," in *2012 25th International Conference on VLSI Design*, IEEE, 2012, pp. 62–67.

[43] A. Carrington, "Kernel methods and measures for classification with transparency, interpretability and accuracy in health care," 2018.

[44] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[45] N. Attarmoghaddam, K. F. Li, and A. Kanan, "FPGA implementation of crossover module of genetic algorithm," *Information*, vol. 10, no. 6, p. 184, 2019.

[46] M. Hemmati, M. Biglari-Abhari, S. Berber, and S. Niar, "HOG feature extractor hardware accelerator for real-time pedestrian detection," in *2014 17th Euromicro Conference on Digital System Design*, IEEE, 2014, pp. 543–550.

[47] M.-E. Ilas, "Improved binary HOG algorithm and possible applications in car detection," in *IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME)*, IEEE, 2017, pp. 274–279.

[48] N. Attarmoghaddam and K. F. Li, "An area-efficient FPGA implementation of a real-time binary object detection system," in *International Conference on Network-Based Information Systems*, Springer, 2020, pp. 127–139.

[49] S. Xie, Y. Li, Z. Jia, and L. Ju, "Binarization based implementation for real-time human detection," in *23rd International Conference on Field Programmable Logic and Applications*, IEEE, 2013, pp. 1–4.

[50] N. Attarmoghaddam and K. F. Li, "An area-efficient FPGA implementation of a real-time multi-class classifier for binary images," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 4, pp. 2306–2310, 2022.

[51] F. Takarli, A. Aghagolzadeh, and H. Seyedarabi, "Combination of high-level features with low-level features for detection of pedestrian," *Signal, Image and Video Processing*, vol. 10, no. 1, pp. 93–101, 2016.

[52] Z. Xiao, P. Xu, X. Wang, L. Chen, and F. An, "A multi-class objects detection coprocessor with dual feature space and weighted softmax," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 9, pp. 1629–1633, 2020.

[53] S.-H. Bae, J.-H. Bae, A. Muqeet, M. S. Monira, and L. Kim, "Cost-efficient super-resolution hardware using local binary pattern classification and linear mapping for real-time 4k conversion," *IEEE Access*, vol. 8, pp. 224 383–224 393, 2020.

[54] A. Arunmozhi and J. Park, "Comparison of HOG, LBP and Haar-like features for on-road vehicle detection," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, IEEE, 2018, pp. 0362–0367.

[55] G. Gan and J. Cheng, "Pedestrian detection based on HOG-LBP feature," in *2011 Seventh International Conference on Computational Intelligence and Security*, IEEE, 2011, pp. 1184–1187.

[56] J. Zhang, K. Huang, Y. Yu, and T. Tan, "Boosted local structured HOG-LBP for object localization," in *CVPR 2011*, IEEE, 2011, pp. 1393–1400.

[57] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *IEEE 12th International Conference on Computer Vision*, IEEE, 2009, pp. 606–613.

[58] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors)," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[59] X. Wang, T. X. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in *IEEE 12th International Conference on Computer Vision*, IEEE, 2009, pp. 32–39.

[60] T. Sledeviè, A. Serackis, and D. Plonis, "FPGA-based selected object tracking using LBP, HOG and motion detection," in *2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, IEEE, 2018, pp. 1–5.

[61] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "Architectural study of HOG feature extraction processor for real-time object detection," in *2012 IEEE Workshop on Signal Processing Systems*, IEEE, 2012, pp. 197–202.

[62] M. Hahnle, F. Saxen, M. Hisung, U. Brunsmann, and K. Doll, "FPGA-based real-time pedestrian detection on high-resolution images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 629–635.

[63] T. P. Cao and G. Deng, "Real-time vision-based stop sign detection system on FPGA," in *2008 Digital Image Computing: Techniques and Applications*, IEEE, 2008, pp. 465–471.

[64] Y. Zhang, W. Cao, and L. Wang, "Implementation of high performance hardware architecture of face recognition algorithm based on local binary pattern on FPGA," in *2015 IEEE 11th International Conference on ASIC (ASICON)*, IEEE, 2015, pp. 1–4.

[65] M. Vergara, A. Wolf, and M. Figueroa, "A texture-based architecture for face detection in IR images on an FPGA," in *Electro-Optical and Infrared Systems: Technology and Applications XI*, International Society for Optics and Photonics, vol. 9249, 2014, p. 92490L.

[66] Z. Ullah, "LH-CAM: Logic-based higher performance binary cam architecture on FPGA," *IEEE Embedded Systems Letters*, vol. 9, no. 2, pp. 29–32, 2017.

[67] S. Kumar, J. Manikandan, and V. Agrawal, "Hardware implementation of support vector machine classifier using reconfigurable architecture," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, 2017, pp. 45–50.

[68] S. Saurav, R. Saini, and S. Singh, "FPGA based implementation of linear SVM for facial expression classification," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, 2018, pp. 766–773.

[69] M. Papadonikolakis and C.-S. Bouganis, "A novel FPGA-based SVM classifier," in *2010 International Conference on Field-Programmable Technology*, IEEE, 2010, pp. 283–286.

[70] C. Kyrkou and T. Theocharides, "A parallel hardware architecture for real-time object detection with support vector machines," *IEEE Transactions on Computers*, vol. 61, no. 6, pp. 831–842, 2011.

[71] M. Cutajar, E. Gatt, I. Grech, O. Casha, and J. Micallef, "Hardware-based support vector machine for phoneme classification," in *Eurocon 2013*, IEEE, 2013, pp. 1701–1708.

[72] D. Anguita, S. Pischiutta, S. Ridella, and D. Sterpi, "Feed-forward support vector machine without multipliers," *IEEE Transactions on Neural Networks*, vol. 17, no. 5, pp. 1328–1331, 2006.

[73] M. Ruiz-Llata, G. Guarnizo, and M. Yébenes-Calvino, "FPGA implementation of a support vector machine for classification and regression," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2010, pp. 1–5.

[74] B. Mandal, M. P. Sarma, K. K. Sarma, and N. Mastorakis, "Implementation of systolic array based SVM classifier using multiplierless kernel," in *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, 2014, pp. 35–39.

[75] N. H. Weste and D. Harris, *CMOS VLSI design: a circuits and systems perspective*. Pearson Education India, 2015.

[76] F. An, X. Zhang, A. Luo, L. Chen, and H. J. Mattausch, "A hardware architecture for cell-based feature-extraction and classification using dual-feature space," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 3086–3098, 2017.

[77] N. Dalal and B. Triggs, *INRIA person dataset*, 2005. [Online]. Available: `http://lear.inrialpes.fr/data/hu-man`.

[78] A. Opelt and A. Pinz., *Graz 01 data set*, 2004. [Online]. Available: `http://www.emt.tugraz.at/%E2%88%BCpinz/data/GRAZ%2001/`.

[79] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.

[80] D. Comaniciu, "An algorithm for data-driven bandwidth selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 281–288, 2003.

[81] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[82] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET curve in assessment of detection task performance," National Institute of Standards and Technology Gaithersburg MD, Tech. Rep., 1997.

[83] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[84] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view traffic sign detection, recognition, and 3D localisation," *Journal of Machine Vision and Applications*, vol. 25, no. 3, pp. 633–647, 2014.

[85] Xilinx, *Vivado Design Suite*. [Online]. Available: `https://www.xilinx.com/products/design-tools/vivado.html`.

[86] Y. S. Abu-Mostafa, "The Vapnik-Chervonenkis dimension: Information versus complexity in learning," *Neural Computation*, vol. 1, no. 3, pp. 312–317, 1989.

[87] Xilinx, *Kintex UltraScale FPGA*. [Online]. Available: `https://www.xilinx.com/products/boards-and-kits/kcu105.html`.

[88] V. Ngo, A. Casadevall, M. Codina, D. Castells-Rufas, and J. Carrabina, "A high-performance HOG extractor on FPGA," *arXiv preprint arXiv:1802.02187*, 2018.

[89] M. Komorkiewicz, M. Kluczewski, and M. Gorgon, "Floating point HOG implementation for real-time multiple object detection," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, IEEE, 2012, pp. 711–714.

[90] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 658–666.