LARGE-SCALE OBJECT RECOGNITION FOR EMBEDDED WEARABLE PLATFORMS

A Dissertation in
Computer Science and Engineering
by
Siddharth Advani

© 2016 Siddharth Advani

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2016
The dissertation of Siddharth Advani was reviewed and approved* by the following:

Vijaykrishnan Narayanan  
Distinguished Professor of Computer Science and Electrical Engineering  
Dissertation Co-Advisor, Co-Chair of Committee

Jack Sampson  
Assistant Professor of Computer Science and Engineering  
Dissertation Co-Advisor, Co-Chair of Committee

Lee Giles  
David Reese Professor of Information Sciences and Technology

Mary Beth Rosson  
Associate Dean and Professor of Information Sciences and Technology

Kevin Irick  
Founder, SiliconScapes, LLC  
Special Member

Chitaranjan Das  
Distinguished Professor of Computer Science and Engineering  
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.
Abstract

Visual object recognition has been an active thrust of research in the vast field of computer vision and neuroscience, both symbiotically closing the gap between man and machine. We, as humans, use years of evolution and a tightly integrated top-down and bottom-up visual system to be able to recognize a large set of objects with a very high degree of precision. When poor lighting, occlusion and varying pose create confusion or when new unidentifiable objects come into the picture, we use context to make an educated guess. For example, consider a foreign tourist looking for a restaurant in a busy city. Even though the tourist may not be able to understand the language of that place, by recognizing known objects such as “plate”, “cup”, “chair”, “bread”, he or she can come to a very quick conclusion about it being a restaurant.

As machines have evolved in their learning capabilities, technology has brought them closer to us allowing finer levels of interactions; initially from being hand-operated to then being hand-held and now being worn. However, current systems being deployed for vision-specific tasks are still either too power-hungry, require huge storage space or computationally take too long for them to be useful for any real-world scenario. Thus, as computing takes a new leap into this exciting world of Wearables, the need for smart cameras capable of supporting, engaging and
enhancing human capabilities has never been felt more acutely.

This thesis tackles one of the critical tasks of large-scale visual object recognition for embedded wearable platforms. A scalable architecture for visual object detection that is fast, accurate, light-weight and power efficient is first proposed. A Context-Aware Scalable Pipeline for Efficient Recognition - CASPER - which uses context in conjunction with a hierarchical visual recognition pipeline targeted for retail is then discussed. We are able to achieve high recognition rates across 62 object classes that are diverse in shape, size and color. The utility of using context in this visual pipeline is showcased and computing effort is reduced significantly without impacting accuracy at all. This allows for higher system throughput with limited area resources - an important factor in enhancing the capabilities of the next generation of wearable devices.
Table of Contents

List of Figures viii
List of Tables xi
List of Abbreviations xii
Acknowledgments xiii

Chapter 1
Introduction 1
1.1 The Goal ...................................................... 1
1.2 The Challenges ............................................. 2
1.3 Motivation .................................................. 3
1.4 Acknowledgments .......................................... 5

Chapter 2
Related Work 6
2.1 Object Recognition ........................................... 6
2.2 Visual Context .............................................. 9
2.3 Vision Accelerators ......................................... 11

Chapter 3
A Scalable Architecture for Real-Time Object Recognition 14
3.1 Algorithm Specifics ........................................... 14
  3.1.1 Performance Characteristics .......................... 15
  3.1.2 Performance Evaluation ............................... 16
3.2 Architecture ............................................... 17
  3.2.1 Raw HOG Accelerator ................................. 18
    3.2.1.1 Dominant Channel ................................ 18
    3.2.1.1.1 Function and performance .................. 18
# Document Contents

## Chapter 5

**Hardware Emulation of Visual Recognition Pipeline**

- **5.1 Coherent Accelerator Processor Interface**
- **5.2 Emulation of Visual Recognition Pipeline**
- **5.3 Discussion on Convolutional Neural Networks**

## Chapter 6

**Future Work**

- **6.1 Domain-specific Research**
  - **6.1.1 Robotics**
  - **6.1.2 Surveillance and Security**
- **6.2 Broad Topics of Research**
  - **6.2.1 Approximate Computing**
  - **6.2.2 Multi-sensor Fusion**

## Chapter 7

**Summary**

## Appendix A

**Datasets**

- **A.1 Inria Person Dataset**
- **A.2 Grocery Dataset**
  - **A.2.1 Wegmans**
  - **A.2.2 BING**
  - **A.2.3 Sophie**

## Appendix B

**Publications**

- **B.1 Refereed Conferences**
- **B.2 Refereed Journals**

## Bibliography
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A Pareto chart showing the most commonly occurring classes across 200 target classes in the ILSVRC 2015 competition.</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Frames per second for face (red) and person (green) detections. Scale factor of 1.1 was used for face detection and 1.05 for person detection.</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Face detection algorithm results on an image from PASCAL dataset.</td>
<td>12</td>
</tr>
<tr>
<td>3.1</td>
<td>HOG-Pedro detections for <em>Person</em> (purple bounding box) and <em>Car</em> (cyan bounding box) object classes.</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>An HD image frame is divided into cells. For each cell a HOG-Pedro (HOG-Dalal) array of 18 (9) orientation bins is computed. These cells are spatially grouped together as blocks. Each cell is then normalized by corresponding blocks that overlap it. An SVM is used learn a HOG template offline. A sliding window containing this template - termed as Region of Interest (RoI) - is run over the entire frame at different scales to produce the predicted outputs.</td>
<td>17</td>
</tr>
<tr>
<td>3.3</td>
<td>HOG Execution Breakdown. The algorithm spends around 40% time performing compute-bound operations and around 50% time performing cache accesses.</td>
<td>18</td>
</tr>
<tr>
<td>3.4</td>
<td>HOG Detection Time as a function of Number of Threads. Image size was Full HD (1920 $\times$ 1080). Multi-scale detection was enabled with a scale factor of 1.5.</td>
<td>19</td>
</tr>
<tr>
<td>3.5</td>
<td>System Architecture mapped to an FPGA. Bin values are generated by the Raw HOG Accelerator (RHA) which is then sent to the Normalized Classification Accelerator (NCA) for detection. Feature Compression block is bypassed when generating HOG-Dalal.</td>
<td>19</td>
</tr>
<tr>
<td>3.6</td>
<td>Dominant Channel Computation.</td>
<td>20</td>
</tr>
<tr>
<td>3.7</td>
<td>Gradient Features Computation.</td>
<td>21</td>
</tr>
<tr>
<td>3.8</td>
<td>Dynamic Histogram Computation.</td>
<td>22</td>
</tr>
</tbody>
</table>
3.9 Block Normalization ........................................ 24
3.10 Feature Compression ......................................... 25
3.11 SVM Accelerator ............................................. 26
3.12 Layout of HOG system design on a Virtex 7 690T FPGA. Xilinx Vivado 2014.4 was used for synthesis and place-and-route. 27
3.13 Speedup across various platforms. Multiscale detection was enabled with a scale factor of 1.5 across all platforms. 28
3.14 Speedup comparisons against CPU for multi-class. Multiscale detection was enabled with a scale factor of 1.5 for both platforms. 29
3.15 Objects detected using a test image from Inria dataset. Red (green) bounding box are results from CPU (GPU) library of OpenCV. 29
3.16 Objects detected using a test image from Inria dataset. Blue bounding box are results from FPGA. 30
3.17 Miss-rate vs False Positives per Image for different HOG models. 31
3.18 Four sub-floats .................................................. 31
3.19 Scalable architecture for many-class object detection 33
3.20 Detecting classes other than pedestrian. 33
3.21 (a1-a2) Groundtruth. (b1-b2) [71]. (c1-c2) [72]. (d1-d2) [73]. (e1-e2) HOG. Green (red) bounding boxes are valid (invalid) person detections based on an IOU>0.25. 35

4.1 A Context-Aware Scalable Pipeline for Efficient Recognition. 36
4.2 Example of ViCoNet’s graph structure. Objects are represented as nodes and when seen together in a visual scene are connected by edges. The weight of an edge between any two nodes is the probability of those two objects seen together. 38
4.3 Four objects seen together in ViCoNet. The weight of a node is the probability of occurrence and the weight of an edge is the probability of co-occurrence in the network. 39
4.4 CASPER Operation ............................................. 41
4.5 Two users shopping in a retail store. User 1 (red) visits the left half of the store while user 2 (pink) visits the right half of the store. 42
4.6 ViCoNet Performance on Test Dataset .............................. 45
4.7 Correlation Coefficient captured for ViCoNet ......................... 46
4.8 Orientation data captured from a wearable device while walking in different directions .......................... 47
4.9 Memory overheads for ViCoNet .................................. 49

5.1 Hardware and software stack used for emulation. ...................... 51
5.2 CAPI ......................................................... 52
5.3 CASPER on a chip ..................................................... 53
5.4 Throughput evaluations across various systems. .......................... 54

6.1 HMAX resilience to errors. Six classes from CalTech101 were used.
   We observe a roll-off in accuracy beyond 512 pixels in error. ............. 56
6.2 Interpolation techniques. In (a), four while in (b), 16 neighboring
   pixels are used for interpolating the value of the pixel at (x',y'). ....... 57

A.1 Training positive examples from the INRIA Person Dataset. Nega-tives
   provided are from natural scenes not containing any people. ............. 61
A.2 Panaromic image of an aisle captured at a Wegmans store. ............. 62
A.3 A few examples of our vision pipeline training dataset. A total of
   62 products were obtained from BING. There is inter-class as well
   as intra-class variance in terms of shape, size and color in the dataset. 62
A.4 High Definition video sequences were captured in different aisles
   of a Safeway store based in California. (a), (b) and (c) are frames
   extracted from a sequence taken in the Cleaning aisle. This temporal
   stream (d) is injected into our testing framework ......................... 63
A.5 Distribution of Test Data ............................................. 63
List of Tables

3.1 Resource Utilization of HD HOG Detector on Virtex 7 690T FPGA 27
3.2 Design Point Comparisons ................................. 30
3.3 Parameters for Multi-class Object Detection [15] ......... 32

4.1 Distribution of object classes across aisles ................. 43
4.2 System Evaluation ............................................ 48

6.1 Impact of Interpolation Techniques .......................... 58
List of Abbreviations

CASPER  Context Aware Scalable Pipeline for Efficient Recognition
CNN    Convolutional Neural Network
ESVM   Exemplar Support Vector Machine
HMAX   Hierarchical Model and X
HOG    Histogram of Oriented Gradients
NCA    Normalized Classification Accelerator
RHA    Raw HOG Accelerator
RLS    Regularized Least-Squares
RoI    Region of Interest
SVM    Support Vector Machine
ViCoNet Visual Co-occurrence Network
Acknowledgments

This thesis and my academic endeavors would not have been possible without the support and guidance of a number of people. I would like to start by thanking Prof. Vijay Narayanan, who gave me the opportunity to be a part of the Microsystems Design Lab and has been a huge pillar of support throughout my Ph.D. He has also been instrumental in my growth as an independent researcher. I would like to thank Prof. Jack Sampson, who joined Penn State a few years back and brought a breath of fresh ideas and methodologies that were critical in making my work competitive and appealing to the research community. I would also like to thank Prof. Mary Beth Rosson and Prof. Lee Giles who as part of my thesis committee have time and again challenged me and made the contributions of this thesis even stronger by helping me push the envelope. Finally, I would like to thank Dr. Kevin Irick who has been more than a mentor to me during my time in the lab. While working with him, he has lent valuable insight to the practical solutions to the problem at hand, and helped me cross-examine the pros and cons of other competing solutions. His technological “know-how” has been a huge help in handling many of the system-level challenges I faced during my thesis.

I would like to thank all my MDL seniors especially Ahmed Maashri, Sung Ho Park and Srinidhi Kestur with whom I interacted on a one-on-one basis in the early phases of my Ph.D. They were patient with me and helped me to ramp up in the areas that I lacked sufficient knowledge. I would like to thank Matthew Cotter, Yang Xiao, Karthik Swaminathan and Nandhini Chandramoorthy, who as peers challenged my ideas and were strong sounding boards when working on my research problem and also gave me an opportunity to be a part of their research. I would like to thank other MDL students, especially Huichu Liu, Moon Seok Kim, William Tsai and Kaisheng Ma for their time and support whenever needed, especially when it came to co-ordinating online meetings or helping with a workshop or demonstration for visitors to the lab. The new students have been extremely driven and kept me on my toes, thus giving me a chance to keep innovating. It has been a pleasure working with Chris Lee, Peter Zientara, Geethanjali Esakkimuthu, Jinhang Choi,
Sumitha George, Srivatsa Srinivasa, Brigid Smith, Komala Madineedi, Gus Smith, Ikenna Okafor, Jake Eden and Vinayaka Krishna in the later part of my Ph.D. I would also like to thank Xueqing Li, who as a post-doc has been a constant source of support across various projects in the lab. The CSE administrative staff and IT team have been par excellence and I wish to thank them for their help.

I would like to acknowledge the collaborators who I have had the opportunity to work with. John Sustersic from the Applied Research Lab, Hong Wang from Intel Labs, Kevin Irick from SiliconScapes and the entire NSF Visual Cortex Team have contributed in some form or the other to the successful completion of this thesis. I would also like to acknowledge the contributions of visiting scholar Yasuki Tanabe, who with his rich research experience in industry gave me an opportunity to learn and apply some of the principles of system architecture and design during his stay in the lab. His work ethics and teachings will always remain with me as I venture out into industry trying to make a mark for myself.

I would like to thank the Ultrasonics group at Penn State, which I was a part of during my Masters at Penn State. My co-advisors Prof. Joseph Rose and Prof. Qiming Zhang were role models during my first innings of graduate life. I would like to thank my previous employer FBS who was extremely supportive of my decision to get a Ph.D. degree. Jason Van Velsor, Steve Owens and Mike Avioli were especially critical in developing my skills in applied R&D that have held me in good stead during my doctoral program.

This is the right time to acknowledge the constant support of my friend circle. Friends from Penn State, my high school group “PANGS” and my undergraduate group “Enjoimaadi” have been always been a call away in times of distress and joy. Many of my early corporate friends, especially Subbarao Nallamothu, Diptiman Singh, Dinesh Pattanayak and Shivani Bhatt have in some way shaped my academic destiny and their support will always be remembered.

This work would not have been possible by the infinite love and sacrifices of my parents. At some level I feel I am living their dreams and that in itself is a huge motivation and source of responsibility on my shoulders. My younger brother has been more than an elder brother to me. He has grown in maturity in leaps and bounds and has been a shoulder on which I have been able to rest on when many have doubted me. I would also like to thank my extended family here in the United States and back in India who have constantly motivated me to move forward in life and advised me whenever I have been in doubt.

Finally, I have derived great motivation from sporting stalwarts Sachin Tendulkar and Roger Federer. Their competitive edge and desire for excellence while still showing great humility and respect for the opponent are trademarks that I will aim to emulate as I venture into my professional career.
Chapter 1  
Introduction

With advancing technology, computing has undergone a metamorphosis time and again, thus enhancing our capabilities of interacting with the world. Progressing from desktops to hand-held devices, a new computing paradigm is making its way into our lives, that of wearable computing. Even though wearables have been around for over a couple of decades now [1], every technology has its own rate of adoption and has to wait its time before it reaches critical mass [2]. As devices become more interactive, the Human Computer Interface (HCI) becomes even more critical in immersive environments [3] and will have an impact on rates of adoption. An interesting study on how personal preferences can affect use and adoption was studied in the context of wearables by [4].

The hurdles that hindered the growth of these systems in the early parts of this millennium are now being considered as plausible opportunities that are sparking the interest of many. According to two different sources, the global wearable market will grow anywhere between four to ten times from 2013 to 2018 [5]. This is quite feasible given that the smartphone industry has grown four times between 2010 and 2015. However, given that this technology is still in its infancy, there are some key challenges, outlined below, which, if addressed, will spur growth even further.

1.1 The Goal

Cameras became ubiquitous with the advent of smartphone devices. As phone displays and camera sensors both scaled, newer hand-held devices integrated these cameras to enable smarter applications such as face detection and eye tracking. Popular social networks like Instagram (acquired by Facebook), Twitter and Snapchat
have millions of people sharing visual data via their mobile applications. As the digital world around us becomes increasingly visual and devices become ubiquitously wearable, cameras will evolve from image capture devices to moment capture devices. Further advances in computer vision are now demanding these camera-friendly wearable devices to be even more power efficient, have better performance and exhibit more powerful capabilities from the underlying technologies than their hand-held counterparts.

Most humans receive more than 85% of all sensory input via the head \[6\]. Some of these head-mounted wearables can be deployed as an assistant for the visually impaired to allow them to become employable in occupations such as tea picking or providing acupressure. Other kinds of watch and glove wearables can be used as personal health-monitoring systems, while the others will soon become a part of the trendy fashion industry.

In the context of streaming camera-based applications, recognizing objects is a critical early stage in the visual pipeline. For example, in a visual driving assist system, an approaching vehicle or a passing pedestrian needs to be identified to preempt any collisions. This needs to be carried out with minimal latency, minimum false positives, and maximum accuracy. On the other hand, a wearable visual prosthesis device needs to augment the visual cognition of the user in diverse and vastly unconstrained adversarial environments for extended periods of time. This thesis brings all these design constraints to the fore when focusing on large-scale visual object recognition targeted for smart camera-based wearables. Key design points of an embedded solution are articulated that achieves the necessary performance and accuracy while being capable of recognizing a large and diverse set of objects in a visual scene. Comparison against other state-of-the-art approaches are used to contrast and motivate the novelty of this work.

### 1.2 The Challenges

In computer vision, object detection is a highly computationally intensive task. To robustly detect an object in an image that may appear at arbitrary position and scale involves (1) extracting optimized features that aptly describe the object and (2) searching the image for the presence of particular configurations of the features that are indicative of the object’s presence. Multi-class object detection becomes
an even harder task since different objects exhibit high appearance variability in shape, color and size.

Designing such embedded systems for wearables compounds the problem even further. Not only, does the application have to be energy-efficient, it also has to be packed into a smaller form-factor while not compromising on quality of service. Addressing these design challenges while focusing on providing users the technology that can empower them rather than burden them [7] will fuel the next leg of innovation in wearables.

1.3 Motivation

The recent ImageNet Large Scale Visual Recognition Challenge (ILSVRC) required object detection of 200 different classes [8]. The provided training dataset, which is the largest available dataset for image recognition, was mined and the top 10 classes occurring in it were logged. As can be seen in Figure 1.1 20% of the data annotated contains the “person” class while the top ten classes (including “person”) contribute 35% of the entire dataset.

![Pareto chart](image)

**Figure 1.1** A Pareto chart showing the most commonly occurring classes across 200 target classes in the ILSVRC 2015 competition.

Drawing key insights from this, this thesis proposes to make the following contributions to the problem of large-scale domain-specific visual object recognition for embedded systems:

- Most visual scenes will have a variety of objects of interest that need to be detected at any given instant. However it is unlikely that all the objects need
to be detected in every visual scene. For example, consider an autonomous driving assistant system where potential target classes may be “car”, “pedestrian”, “traffic light”, “bicyclist” to name a few. An embedded architecture that can detect a limited number of such commonly occurring objects in a fast, accurate and power efficient manner while satisfying minimum area constraints is proposed.

- Contrary to most image datasets, there are certain real-world scenarios, where fine-grained object recognition is needed. For example, a retail store can have more than a thousand different products each occurring in different shape, size and color and when a shopper walks into a particular aisle, (s)he would take around 1-2 seconds on average to recognize a particular brand of product. A brute-face approach would be to run classifiers for each of the classes in the object space. What may not be very obvious is that these objects are seen together in a particular context. For example, in the cereal aisle, one is more likely than not to find a “Kellogg’s Frosted Flakes” cereal box rather than a “Barilla Penne” pasta box. As humans, we tend to bias our predictions based on context. To exploit this contextual awareness, a Bayesian model is proposed to encode context and is then integrated with an hierarchical extension of the aforementioned architecture to recognize objects in a domain-specific application like retail where the classes are diverse, yet have spatial dependence.

- An instantiation of the proposed vision system is emulated on an FPGA enabled with a Coherent Accelerator Processor Interface (CAPI) \cite{9} that treats the accelerators as peers in the system rather than as slaves. This platform enables hardware-software co-design of an end-to-end application.

The rest of this dissertation is organized as follows: Chapter 2 presents related work in the area of embedded vision in particular and visual object detection and classification in general. Chapter 3 focuses on the design of an embedded architecture that can detect a limited number of objects in a fast, accurate and power efficient manner while satisfying minimum area constraints. Chapter 4 extends the aforementioned design into a two tier visual classification pipeline and a context-aware approach is presented for recognizing objects in a domain-specific application like retail where the number of classes is large and diverse. Chapter 5
provides implementation specific details of the proposed vision system using a Coherent Accelerator Processor Interface (CAPI) to enable hardware-software co-design of the proposed solution. In Chapter 6, as part of future work, different areas of research are highlighted and corresponding opportunities are suggested. Finally, Chapter 7 summarizes the contributions of this dissertation.

1.4 Acknowledgments

Chapters 2, 3, 4 and 5 contain material from:


Chapters 2 and 6 contain material from:

Chapter 2 | Related Work

This chapter provides the reader a broad overview of the literature in the field of embedded visual object recognition. Section 2.1 gives an overview of object recognition and the different approaches that could have been taken. Section 2.2 gives an overview of visual context and its utility. Finally, in Section 2.3 a discussion on state-of-the-art computational accelerators for different vision systems is provided.

2.1 Object Recognition

Object recognition has been an active field of research in the computer vision domain for quite sometime now. Traditionally, pixel-level features have been used to describe different objects and these features are then used to train a classifier to accomplish autonomous recognition. Another approach is to use robust features that can be matched against a template model in a brute-force fashion. The Scale Invariant Feature Transform (SIFT) is one such feature that uses a Difference of Gaussian (DoG) approach to produce translation and rotation invariant feature vectors that can then be used to compare against a template \[10\]. HMAX is a biological model that processes visual input in a similar fashion as the earliest stages of the human visual system (HVS). Once extracted from the image, these HMAX features can then be used to train a Regularized Least-Squares (RLS) classifier \[11\] for recognizing objects. Support Vector Machines (SVMs) is another type of classifier that has shown to work very well with sparse features. In \[12\], the authors apply both these classifiers to a number of datasets and discuss the tradeoff in using either of them. Another compositional model - Hierarchical Object
Parsing (HOP) [13] - recognizes complex objects by combining simpler parts into complex representations.

The Histogram of Oriented Gradients (HOG) was first introduced for pedestrian detection in [14]. It was also shown to work well with other object classes as highlighted in [15]. Since then, numerous HOG based object detectors have been proposed in the vision community, especially for pedestrian detection. In [16], sixteen pedestrian detectors were evaluated on six datasets. Of these sixteen detectors, fourteen use some variant of gradient histogram. The work of [17] use a variant of the HOG model that extracts signed and unsigned orientations along with texton features and then projects these into a 31 dimension feature vector. The authors then train a latent support vector machine (LSVM) to detect different objects. In [18], the authors use the same feature extraction as [17], but use an Exemplar SVM (ESVM) as the classifier. In the ESVM framework every representative instance of a class has a single positive exemplar and many negative examples in the HOG feature space. Both [17] and [18] performed extremely well in the PASCAL Visual Object Classes (VOC) Challenge [19].

Different kinds of datasets have been used to benchmark the performance of these HOG-based object detection models. The Inria dataset used by the seminal HOG work [14] is a popular dataset that continues to be used for evaluating pedestrian detection systems. Newer and tougher datasets like the CalTech Pedestrian dataset [16] have also been introduced to train pedestrian models that account for more realistic scenarios that include occlusions and various human poses. Moving towards a more generic form of object detection, newer, larger and broader datasets like ImageNet [20] have become immensely popular. However it is only recently that large datasets that allow controlled noise into the data are being explored. This allows for the vision system to learn a variety of invariances typical to real-world scenarios [21].

Rather than use structured features, different data-driven approaches to characterize the features used to train the classifier have also been explored. Bag of Words (BoW) was a popular data-driven approach where conventional edge-like features were extracted and then clustered to form features in a new space that could make the classifiers task easier [22]. Even though Convolutional Neural Networks (CNNs) were explored in the mid 1990s for vision applications [23], they have resurfaced again after a long hiatus and become extremely popular in the past couple of
years. This successful comeback can be attributed to two major phenomena: (1) the existence of large amount of data (needed to train the network well) with the evolution of the digital era, and (2) the development of custom hardware (required for acceleration) now being used for CNNs. In the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) conducted in 2012, the winning team trained a CNN - “AlexNet” - consisting of five convolutional and three fully-connected layers. Importantly, the depth of the CNN is critical to its recognition capabilities since the authors found that removing any convolutional layer resulted in inferior performance [24]. So, “AlexNet” in its current form would need more than 80 million operations and over 100,000 data transfers [25].

More recent and advanced CNN architectures have 10 to 20 layers of Rectified Linear Units, hundreds of millions of weights, and billions of connections between units. An important insight recently unearthed by [26], is that the accuracies of CNNs can saturate after a few million images of training data. The reader is pointed to [27] for insights on deep architectures in general and [28] for CNN-based learning and their recent advances. From a systems perspective, [29] mapped an earlier Convolutional Network based face-detection task onto custom hardware, but found that the system was bandwidth-limited due to the numerous datafetch operations. This bottleneck would further exacerbate as more layers are added to the network. More recently, [30] recently proposed an architecture for CNNs and Deep Neural Networks (DNNs) that minimized memory transfers thus achieving high throughput with small area, power and energy footprint.

Industry too is manifesting some of its solutions geared towards machine learning and computer vision for application specific domains. Tensilica’s Vision P5 DSP running at 600MHz uses a CNN to process more than 850 traffic signs in one second, suggesting that deep CNNs will soon be commercially available for a number of embedded solutions [31]. For server-based applications, hardware accelerators designed for large-scale recognition was studied in [32]. In [33], the authors propose a winner-take-all (WTA) hash descriptor to achieve computational benefits when deploying 100,000 detectors. [34] developed a cloud server based system for fine-grained object recognition in retail. In the later two works, the Histogram of Oriented Gradients (HOG) was an integral part of the evaluation strategy albeit for desktop and server-type platforms.

Object detection is a critical early stage in such systems to help reducing the
number of computationally intensive tasks that are performed in the latter stages of a typical vision pipeline. In [35], a FPGA-based attention-driven detector is developed using bottom-up pixel features rather than using a sliding window approach. However the detected windows would then have to be fed to a feature extraction-classification pipeline to identify the objects as highlighted in [36]. In [37], the authors present a single-scale HOG based pedestrian detector using features as detailed in [14] while in [38], a multi-scale version is presented. In [39], a heterogeneous approach is taken to the multi-scale pedestrian detector and tradeoffs in power, speed and accuracy are evaluated. In contrast to these three HOG-based works, our work focuses on the challenges that a many-class object detection problem presents. These aspects of a general multi-object detector is not considered in [37–39].

2.2 Visual Context

Visual context was represented by [40], [41] in the form of a gist feature extracted from low-level pixel features. Both works used different characteristics of the image to extract these global gist features and then trained a classifier to recognize different scenes.

At a higher level of abstraction, [42] compute a bag-of-features (BoF) representation for all segments independently in a test image. Having predicted labels for these segments, a conditional random field (CRF) approach is used as a post-processing step to improve categorization accuracy. Rather than using region specific BoF, [43] use a context-specific BoF approach to handle the ambiguity involved when dealing with synonymy and polysemy. [44] propose a generative graphical model to coherently perform image level classification, individual object annotation and pixel level segmentation.

More recently and more closely related to the application domain our work focuses on, [45] explores domain-specific object recognition by proposing a framework designed for speed and scalability that takes advantage of the hierarchical grouping of objects within a retail environment. To perform a classification, they first perform a consensus-based, fine-grained object classification that will filter out categories of labels by the ranking of its output. This step takes advantage of the fact that retail environments are arranged hierarchically and allows their framework to remove
the least likely contenders for classification early on. Then, they use high-density pixel matching, a technique that does not require any training and is therefore quite scalable, to perform the classifications, and the output of this step is ranked using an energy function to determine the final labels. This work is important to us largely because the authors exploit spatial knowledge about a scene to refine their classification. The hierarchical organization of objects in [45] follows the form of “Food/Candy/Chocolate”, using an example from the paper. These relationships cannot necessarily be intuited directly from extractable features of the objects in question and must instead come from a higher-level understanding of object groupings.

In contrast, our work differs from the above work since the fundamental goal is to exploit the contextual relationships that exist in the world (more specifically retail) while simultaneously not bounding it to any particular algorithm for obtaining the said context. We thus build a layer of abstraction between our Bayesian model of context and the underlying classification system populating it. However, since this model builds context based on objects that are seen together in a scene, it structurally bears similarity to the seminal work by [46] where nodes represent web-pages on the Internet and edges represent links between these web-pages. The reader is pointed to [47] for a primitive approach to encoding context for object recognition that evolved into the work presented in this thesis.

Very recently, context-aware smartphones have been proposed too [48]. In [49], the authors use a neuromorphic accelerator based on Hebbian learning targeted for context-aware text recognition in documents. In other embedded vision work, [50] dynamically reconfigure a real-time object recognition pipeline based on the visual scene. Our work significantly differs from this since we focus on doing less work using context while maintaining accuracies. From a hardware-software perspective, [51] uses a Bayesian network for compiler auto-tuning. We propose a Bayesian network too, but again, the goal here is to build a mathematical framework to achieve the same or better accuracy while doing less work. As a motivation, the recently concluded Low-Power Image Recognition Challenge [52] demanded not only low-power but high accuracy when detecting and recognizing 200 object classes with both accuracy and power having equal weightage. Thus, a system designed with these constraints in mind while minimizing overheads beckons for significant all-round savings.
2.3 Vision Accelerators

With the advent of mobile computing, there has been considerable emphasis in the embedded domain for creating real-time vision systems operating under modest power budgets. With increasing image resolutions, achieving real-time performance for vision applications is becoming exceedingly difficult. As an example, we ran the multi-threaded face detection and person detection algorithms from the latest stable release of OpenCV (2.4.11) [53] using a general purpose CPU (8 Core, Intel i7-4770 3.40 GHz, 32 GB RAM, 60W) and compared the performance for different image resolutions. Fig. 2.1 shows the detection time in terms of frames per second for different image sizes ranging from VGA (640 × 480) to HD (1920 × 1080). As can be seen, performance deteriorates by more than 80% when moving from VGA to an HD image on the CPU. Fig. 2.2 shows the faces detected when the face detection algorithm was run on an image from the popular PASCAL Visual Object Classes (VOC) Challenge dataset [19].

Very Long Instruction Word (VLIW) based Digital Signal Processors (DSPs) have been a universally accepted alternative to general purpose CPUs for seamless multimedia processing. Similar to CPUs, DSPs are fundamentally operated at instruction-level granularity and makes for a good low power accelerator in mobile
System on Chips (SoC) to support the multi-core CPU. For example, Qualcomm’s Hexagon DSP instruction set architecture (ISA) contains numerous special-purpose instructions designed to accelerate key multimedia kernels such as sliding window filters \[54\]. Qualcomm’s latest Snapdragon 820 provides additional hardware support to the DSP via Hexagon Vector eXtensions (HVX) that include fixed point SIMD operations. Heterogeneity has often been considered as a viable solution to handle the increasingly varying nature of workloads that need to be run today \[55\]. Texas Instruments has a heterogenous multi-core DSP targeted for real-time vision applications based on their Keystone architecture. To tackle the diverse field of vision many other heterogenous architectures have been proposed that take advantage of customized flows for regular systolic operations while using the traditional Von Neumann architecture for handling control logic and other irregular data operations. A heterogenous server architecture consisting of many small cores for low power and high throughput coupled with custom hardware accelerators was designed in \[32\]. In \[56\], the authors explored architectural heterogeneity by using customized data-flows for many vision-based applications.
targeted at retail, security, etc.

Graphics Processing Units (GPUs) have evolved from being considered as workhorses for solving large scientific problems like weather forecasting \cite{57} to now being recognized as first-class citizens alongside general purpose processors \cite{58}. While extremely power-hungry, these devices are highly optimized for embarrassingly parallel Single Instruction Multiple Data (SIMD) applications like deep learning \cite{59}, and are now emerging as a popular choice in infotainment systems for high-end vehicles like Tesla Motors’ Model X.

Due to the capacity of the human vision system (HVS) for highly complex processing at very low power, many brain-inspired algorithms and architectures have been proposed to emulate the human visual cortex on different compute fabrics. Inspired by the structural and functional properties of the mammalian neocortex, \cite{60} proposed a GPGPU-accelerated intelligent learning model. \cite{61} described an end-to-end attention-recognition FPGA accelerator pipeline to emulate visual recognition in the visual cortex. Other vision accelerators have shown to be extremely performance-friendly for computationally intensive tasks such as face detection \cite{62}, pedestrian detection \cite{38}, object recognition \cite{63} and object detection \cite{35}. In \cite{64}, the authors propose a benchmarking suite - VISBench (Visual, Interactive, Simulation Benchmarks) - and find that a MIMD rather than a SIMD architecture gives better performance. In \cite{65}, the authors studied the quantitative energy benefits of vector acceleration.

In conclusion, this chapter has provided the reader an overview of the existing literature, which will provide context to the contributions of this thesis discussed in the chapters to follow.
Chapter 3  |  A Scalable Architecture for Real-Time Object Recognition

Our first step towards achieving fast and accurate object recognition is based on the Histogram of Oriented Gradients, which was initially conceptualized with the intention of detecting people in visual scenes \[14\]. While this version, referred to as HOG-Dalal in this chapter, was used for other classes as well, a variant was proposed in \[17\] which showed better discriminative properties for the multi-class problem. We refer to this as HOG-Pedro. Figure 3.1 shows objects detected using HOG-Pedro for two target object classes - Person and Car.

Section 3.1 gives an overview of the Histogram of Oriented Gradients algorithm and its performance characteristics. We sketch out the architecture for a HOG-based recognition scheme and highlight the salient features in Section 3.2. Section 3.3 provides a complete set of results by comparing our FPGA-based system with a CPU and GPU version. Given an area budget, we then showcase how the proposed architecture can be easily mapped to solve the multi-class problem in Section 3.4. Finally, we compare this sliding window approach to a more generic detection scheme and discuss the trade-offs between the two in Section 3.5.

3.1 Algorithm Specifics

Algorithm 1 gives a breakdown of the key steps in generating valid detections using HOG features. Figure 3.2 illustrates an example providing specific pictorial details about the algorithm. For HOG-Dalal, gradient magnitudes are accumulated across nine orientation bins for each cell. A cell is then normalized by four different
overlapping blocks thus giving a total of 36 feature elements per cell. For HOG-Pedro, gradient magnitudes are accumulated across 18 orientation bins for each cell. 18 Contrast-sensitive and nine contrast-insensitive features are accumulated along with four texton features to give a total of 31 feature elements per cell. Support Vector Machines (SVMs) are popular learning models that have shown to be very responsive to both these HOG features. A linear SVM is trained using the HOG features that are extracted from sufficient training data and this SVM can then later be used in the recognition stage.

### 3.1.1 Performance Characteristics

We profiled the HOG-SVM object recognition in software using the latest stable release of OpenCV (2.4.11) [53] on a single thread of a general purpose CPU (8 Core, Intel i7-4770 3.40 GHz with 32 GB RAM). The OpenCV implements HOG-Dalal and uses loop unrolling and caching techniques to provide highly optimized performance on the CPU. Figure 3.3 highlights the fundamental bottlenecks and shows that the algorithm is almost equally compute-bound as is memory-bound.
Algorithm 1 HOG Detections

\textbf{procedure} \textsc{Detect}(\textit{frames}, \textit{mode})

\begin{algorithmic}
\State \textbf{for all} \textit{frames}_i \textbf{such that} \(1 \leq i \leq N\) \textbf{do}
\State \textbf{for all} \textit{channels}_j \textbf{such that} \(1 \leq j \leq 3\) \textbf{do}
\State \text{Compute gradient magnitude and orientation}
\State \text{Assign each pixel the orientation of the dominant channel magnitude}
\EndFor
\State \text{Divide} \textit{frames}_i \text{ into spatial cells}
\State \textbf{for all} \textit{cells} \textbf{do}
\State \text{Bin gradient magnitudes based on orientation}
\EndFor
\State \text{Group} \textit{cells} \text{ into overlapping blocks}
\State \textbf{for all} \textit{blocks} \textbf{do}
\State \text{Compute normalization factor}
\EndFor
\State \text{Apply normalization to each overlapping cell}
\State \text{Accumulate values as a feature vector}
\If {\textit{mode} == \textit{Pedro}}
\State \text{Compress features}
\EndIf
\State \text{Apply linear SVM at all scales and locations}
\State \text{Fuse overlapping detections}
\State \text{Output top detections}
\EndFor
\end{algorithmic}

\textbf{end procedure}

across various image sizes.

\subsection{3.1.2 Performance Evaluation}

Having ascertained the hot-spots in the algorithm, we evaluated absolute execution times as a function of number of CPU threads on the same 8 core machine. As shown in Figure 3.4, the detection time scales logarithmically with the number of parallel threads which suggests a high degree of inherent parallelism. The detection latency far exceeds the minimum requirement of 33 milliseconds to maintain 30 frames per second (fps) real-time throughput constraints. This motivates our investigation of hardware acceleration.
Figure 3.2 An HD image frame is divided into cells. For each cell a HOG-Pedro (HOG-Dalal) array of 18 (9) orientation bins is computed. These cells are spatially grouped together as blocks. Each cell is then normalized by corresponding blocks that overlap it. An SVM is used learn a HOG template offline. A sliding window containing this template - termed as Region of Interest (RoI) - is run over the entire frame at different scales to produce the predicted outputs.

3.2 Architecture

In this section, we discuss the key design points of the visual object detection pipeline. Figure 3.5 illustrates the end-to-end system beginning with the PCIe host data interface.

Two hardware accelerators are integrated into the system - one to extract the
Figure 3.3 HOG Execution Breakdown. The algorithm spends around 40% time performing compute-bound operations and around 50% time performing cache accesses.

raw HOG features - Raw HOG Accelerator (RHA) - and the other to carry out region-based normalization followed by the detection using SVMs - Normalized Classification Accelerator (NCA). For a single class detection, these two accelerators can be coupled directly to each other without needing to use off-chip memory as a staging buffer. In this configuration, histogram bin contents produced by the RHA stream directly into the NCA. However for reasons explained later, we decouple these two accelerators to support multi-class object detection in a single frame.

### 3.2.1 Raw HOG Accelerator

The Raw HOG Accelerator (RHA) consists of dominant channel computation, gradient computation and dynamic histogram computation, each of which are explained in detail below.

#### 3.2.1.1 Dominant Channel

**3.2.1.1.1 Function and performance** The dominant channel computation module determines, at a pixel granularity, which of the color channels has the most
**Figure 3.4** HOG Detection Time as a function of Number of Threads. Image size was Full HD (1920 × 1080). Multi-scale detection was enabled with a scale factor of 1.5.

**Figure 3.5** System Architecture mapped to an FPGA. Bin values are generated by the Raw HOG Accelerator (RHA) which is then sent to the Normalized Classification Accelerator (NCA) for detection. Feature Compression block is bypassed when generating HOG-Dalal dominant gradient. It thus operates as a data compression mechanism by throwing out the redundant channels for every pixel and keeping the best one for further processing.

**3.2.1.1.2 Architecture** As shown in Figure 3.6, the module consists of a convolution engine for each channel. The difference operators \([-1 \ 0 \ 1]\) and \([-1 \ 0 \ 1]^T\) are
inputs to the convolution engine and the gradients $I_x$ and $I_y$ are computed using the same window buffer in a streaming manner. The outputs of the convolution engine are squared and egressed to an accumulator to determine the gradient magnitude. A comparator tree determines the dominant channel for the active pixel which is ultimately the output of the module. This module is agnostic of the color space it is operating on and thus can be used for HSV or YUV domains too and is also scalable to more than three channels.

![Figure 3.6 Dominant Channel Computation](image)

### 3.2.1.2 Gradient Features

#### 3.2.1.2.1 Function and performance

Similar to the dominant channel module, the gradient features module also operates in a pipelined streaming fashion. As shown in Figure 3.7, the gradient magnitude is obtained using the square root operator while the quantized gradient orientation is computed using the arctangent
operator. Both operators work in parallel and the overall latency of this module is 48 cycles due to the square root module. Four bytes are allocated for each output entity i.e. gradient magnitude and gradient orientation.

3.2.1.2.2 Architecture The square root operator is computed using CORDIC while the arctangent operator computes 18 dot-product values for different quantized angles and assigns the orientation to the quantized angle with maximum response. It should be pointed out that the dot-product coefficients are symmetric and hence can be re-used thus utilizing fewer DSP resources.

![Figure 3.7 Gradient Features Computation](image)

3.2.1.3 Dynamic Histogram

3.2.1.3.1 Function and performance Once computed, the gradient features are binned according to their orientation. Histograms are generated at a cell granularity with 18 bins per cell in the range from 0 to $2\pi$. In order to support histogram operations across varying cell sizes, we introduce a virtual histogram mechanism based on a pre-configured page table. In this protocol we allocate 256 histogram tags that can be used to bin into a global histogram indexed based on the group-id in the page table. We are thus able to achieve a high degree of configurability while being able to reuse the shared histogram resources for cells that have exceeded their lifetime and are no longer active.

3.2.1.3.2 Architecture Figure 3.8 illustrates the core architecture of this module. The Cell Index Compute module determines the cell-id for each pixel.
This cell-id is used to look up a pre-configured page table to read an associated group-id. Based on the group-id, the past neighbors of the cell are checked to see if a matching group-id is present. In case a match occurs, then the histogram tag of that neighbor propagates to the current cell and used to bin the current gradient feature. If no match exists, then a new histogram tag is generated which is then written into the page table for other pixels belonging to that cell to refer to. The gradient feature for that pixel is then binned into the new histogram. A controller triggers when the last row of a cell is being processed. At that time, the group ids of the future neighbors of the current cell are checked to determine if the current virtual histogram will live beyond the current cell. If not, then that particular histogram can be retired and the tag released for reuse. If there is a group-id match, then the current virtual histogram remains active.

Figure 3.8 Dynamic Histogram Computation

3.2.2 Normalized Classification Accelerator

The Normalized Classification Accelerator (NCA) consists of block normalization, feature compression and SVM matching modules, each of which are explained in detail below.
3.2.2.1 Block Normalization

3.2.2.1.1 Function and performance  Local normalization of descriptor vector is essential to reduce effects of illumination and contrast variations in images that can affect recognition accuracy. In [16], an $L_2$-norm based block normalization scheme is evaluated and achieved 27% miss rate reduction at $10^{-4}$ false positives per window. In block normalization, a block is defined as a unit of $2 \times 2$ cell. A cell is therefore normalized by four $L_2$-norm of four blocks. The Block Normalization module shown in Figure 3.9 is designed to process this normalization for each cell in raster-scan order and it can output a normalized bin value every cycle.

3.2.2.1.2 Architecture  The Cell $L_2$ sub-module receives bins and outputs sum of square of HOG bins for each cell. Calculated cell $L_2$ is transferred to Block $L_2$ sub-module and stored in Cell $L_2$ Line Memory for future use.

In the Block $L_2$ sub-module, $L_2$ of the lower-right cell is received from the Cell $L_2$ sub-module, $L_2$ of the upper-right cell is loaded from the Cell $L_2$ Line Memory and block $L_2$ is calculated using $L_2$ of upper-left and lower-left cached from previous block calculation. The SQRT module calculates block $L_2$-norm. To realize normalization of bins by a multiplier instead of a divider, reciprocal of denominator i.e. the normalization scale is calculated in the INV sub-module. Since the results are not required for each clock cycle in this design, both the SQRT and INV sub-modules are implemented without multipliers and 16 clock cycles are used to generate a 16 bit square root and 16 bit scale respectively. The calculated scale is then kept in the Scale Line Memory. The Norm sub-module loads a bin from the Bin Line Memory and a scale from the Scale Line Memory every cycle and calculates a normalized bin. Since a single cell belongs to four adjacent blocks, each bin in a cell is loaded four times and normalized by four scales.

3.2.2.2 Feature Compression

3.2.2.2.1 Function and performance  A scheme introduced in [17] compresses features and reduces feature descriptor dimensions. In a system which needs to detect a wide variety of objects, we have to hold several SVMs in memory. The reduction of feature descriptor dimensions reduces total amount of memory required for the SVMs. The maximum size of an RoI supported by SVM accelerator
described later in Section 3.2.2.3 is 32 × 32 cells (256 × 256 pixels). This mode is always operational in the degenerate case. In a many-class system, we save 2 BRAMs for every class that uses this compression method.

This Feature Compression module shown in Figure 3.10 is implemented to receive a normalized bin every cycle and outputs compressed HOG features.

3.2.2.2 Architecture The Block Normalization module generates 72 dimension features for each cell by normalizing 18 dimension HOG features by 4 blocks. This Feature Compression module compresses 72 dimensions into 31 dimensions. 18 dimension features are generated by accumulating each bin for blocks (Contrast Sensitive Feature) while 9 dimension features are generated by coupling 18 bins into 9 bins and accumulating the 9 bins for each block (Contrast Insensitive Feature). 4 dimension features are generated by accumulating bin values for each block (Texture Energy Feature). To realize on-the-fly compression, several registers are used in this sub-module.

3.2.2.3 Support Vector Machine

3.2.2.3.1 Function and performance Sum of dot products between the image feature descriptor of an RoI and a trained linear SVM can be considered as a likelihood of the target object. This system is designed to achieve 64 × 128 pixel RoI scan processing with 8 pixel sliding window on a Full HD (1920 × 1080) video sequence running at 30 fps. However, since the dimension of the feature descriptor
is vast, a significant number of dot production computations are required. To process this, an accelerator shown in Figure 3.11 is implemented, where a feature descriptor of up to 36 dimensions for each cell is supported. For a scalable system, a wide variety of objects needs to be detected, and hence various RoI sizes need to be supported. This module is designed to support RoI sizes of up to \(32 \times 32\) cells (256 \(\times\) 256 pixels for a fixed \(8 \times 8\) pixel per cell configuration).

### 3.2.2.3.2 Architecture

A feature buffer is implemented as a double buffer to hold data which is received while the other side is used for dot product calculation. Either the normalized feature described in Section 3.2.2.1 or the compressed feature described in Section 3.2.2.2 can be used as input.

In an input video stream, a cell can belong to several RoIs. For example, if we use \(8 \times 16\) cell (64 \(\times\) 128 pixel) RoI size, a cell can belong to 128 RoIs in different positions. To realize FullHD 30 fps sliding window processing, input features of a cell are re-used for every RoI that the cell may belong to (up to \(32 \times 32\) RoIs). 8 instances of dot product and adder tree are implemented for this processing. Every cycle, dot products of 9 features are calculated in parallel for 8 RoIs in horizontally different positions using these 8 instances. Cycles equal to the RoI’s height in a cell are used for RoIs in vertically different positions. If the RoI width is larger than 8 cells, extra cycles will be used to calculate other RoIs in horizontally and
vertically different positions. After 9 features are reused for every RoI that the cell may belong to, this module will load other 9 features of a cell and process it in the same manner. After processing all features in a cell, 9 features of next cell held in the other side of feature buffer will be processed. The Dot Product Buffer is used to hold the sum of dot products of RoIs which are being processed. Since the module supports raster scan processing of FullHD, 240 cells in horizontal and RoIs up to vertically 32 different positions are needed to be held in the buffer. With a margin for a wider image, a 32bit×32 × 256 buffer is implemented. After the sum of dot product is calculated for all features within an RoI, the result from this module is written out to memory.

![Figure 3.11 SVM Accelerator](image)

**Figure 3.11 SVM Accelerator**

### 3.3 Results

We synthesized the design using Vivado 2014.4 mapped to a Virtex 7 690T FPGA. Table 3.1 shows the resource utilization for both single scale and multi-scale detection. In the multi-scale approach, each scale is computed iteratively using the same pipeline to avoid the need of BRAM-heavy pixel buffers. Thus, at a scale granularity, computation is purely streaming. We synthesize a configurable scale generator and include the resources utilized by that block for the multi-scale version.
of the system. Figure 3.12 illustrates the layout of the entire single-scale object detector system which consists of the PCIe core, RHA, NCA, Memory and Router as the major blocks.

Table 3.1 Resource Utilization of HD HOG Detector on Virtex 7 690T FPGA

<table>
<thead>
<tr>
<th></th>
<th>Flip Flops</th>
<th>LUTs</th>
<th>BRAMs</th>
<th>DSP48s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Scale</td>
<td>102929 (12%)</td>
<td>130329 (30%)</td>
<td>275 (19%)</td>
<td>158 (4%)</td>
</tr>
<tr>
<td>Multi-Scale</td>
<td>113507 (13%)</td>
<td>137361 (32%)</td>
<td>375 (26%)</td>
<td>202 (6%)</td>
</tr>
</tbody>
</table>

Figure 3.12 Layout of HOG system design on a Virtex 7 690T FPGA. Xilinx Vivado 2014.4 was used for synthesis and place-and-route.

We compared the performance of our design against CPU and GPU implementations of HOG-Dalal. The CPU version was evaluated as explained in Figure 3.4 with 8 threads on a 8 core machine. The GPU used was Nvidia GeForce GTX590
and we evaluated the OpenCV GPU library for HOG-Dalal on it. As shown in Figure 3.13 for a Full HD video stream, in multi-scale mode we show 5x performance improvement over the CPU and 2x performance improvement over the GPU. We also compared our FPGA implementations against a CPU implementation in multi-class mode for a Full HD stream. As can be seen in Figure 3.14, we can achieve over 20x improvement over the CPU for 5 classes. We were not able to make comparisons with the GPU since we used the pre-compiled GPU library that does not support multi-class.

Figure 3.13 Speedup across various platforms. Multiscale detection was enabled with a scale factor of 1.5 across all platforms.

Figure 3.15 shows objects detected for the Person class in a test image obtained from the Inria Pedestrian dataset. Red (green) bounding boxes depict people detected by the CPU (GPU). Correspondingly, Figure 3.16 shows objects detected by the FPGA in HOG-Pedro mode.

Our evaluation (refer Table 3.2) indicates that we achieve ASIC like performance on Full HD video while supporting multiple scales and multiple classes. Our design is the first to support multiple classes by provisioning a specialized block that produces compressed HOG features common to all classes. Furthermore our design operates at 100 MHz, since power is a dual goal and further performance improvements are
Figure 3.14 Speedup comparisons against CPU for multi-class. Multiscale detection was enabled with a scale factor of 1.5 for both platforms.

Figure 3.15 Objects detected using a test image from Inria dataset. Red (green) bounding box are results from CPU (GPU) library of OpenCV.

possible. There are other efforts akin to [67] that implement HOG on FPGAs. While some of these implementations achieve even higher frame-rates [68], none of them support multi-class processing and some offload portions of computations to a host machine [69]. In terms of power, our FPGA design consumes 7W, an order
Figure 3.16 Objects detected using a test image from Inria dataset. Blue bounding box are results from FPGA.

Table 3.2 Design Point Comparisons

<table>
<thead>
<tr>
<th>System</th>
<th>This Work</th>
<th>[67]</th>
<th>[38]</th>
<th>[37]</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Classes</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>No. of Scales</td>
<td>6</td>
<td>-</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>FPS</td>
<td>24</td>
<td>33</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>Technology</td>
<td>28nm Virtex7</td>
<td>40nm Virtex6</td>
<td>45nm CMOS</td>
<td>65nm CMOS</td>
</tr>
<tr>
<td>Clock</td>
<td>100 MHz</td>
<td>-</td>
<td>270 MHz</td>
<td>42.9 MHz</td>
</tr>
<tr>
<td>Image Size</td>
<td>1080 × 1920</td>
<td>1080 × 1080</td>
<td>1080 × 1920</td>
<td>1080 × 1920</td>
</tr>
<tr>
<td>Bit Precision</td>
<td>16</td>
<td>13</td>
<td>9</td>
<td>-</td>
</tr>
<tr>
<td>Platform</td>
<td>FPGA</td>
<td>FPGA</td>
<td>ASIC</td>
<td>ASIC</td>
</tr>
</tbody>
</table>

of magnitude less than our baseline CPU (~60W) or GPU (~300W).

We also evaluated our fixed point design on the INRIA Test Dataset and we achieve comparable accuracy compared to the OpenCV floating-point software version as shown in Figure 3.17. All three models (dashed lines) were trained using LibSVM [70] on the INRIA training dataset and the 288 test images were non-overlapping with this training dataset. More details about the dataset can be found in Appendix A.1. As a baseline, we also plotted the pre-trained OpenCV detector (solid line). The plot suggests that further refinement in training can
Figure 3.17 Miss-rate vs False Positives per Image for different HOG models.

Figure 3.18 Learnt SVM models: (a) HOG-OpenCV; (b) Self-trained HOG-OpenCV; (c) Hardware HOG-Dalal; (d) Hardware HOG-Pedro.
help reduce miss-rates. Figure 3.18 shows the learnt SVM model for all the HOG features evaluated.

### 3.4 Case Study

For many-class object detection, the visual object detection pipeline outlined in the previous sections needs to be modified minimally. For every image frame, the RHA needs to generate the raw HOG features only once. These are then sent to external memory using an on-chip memory controller. For each class a class-specific NCA is instantiated which is configured with (i) the kind of feature it needs - HOG-Dalal or HOG-Pedro, (ii) the kind of block normalization it needs and (iii) the RoI size of that class. For \( m \) classes that we need to support we instantiate \( m \) NCA modules in parallel. Since the raw HOG features are broadcast to each NCA via external DDR3 memory, assuming a modest average bandwidth of 1 GB/s, we can easily support transferring the raw HOG features for a Full HD frame for up to 8 classes. The individual SVM weights of each class are also read from external memory but rather than continuously reading them on-the-fly for every frame, we fetch them only once at the start of the first frame, and store them in an on-chip buffer. Since we support a max RoI size of \( 32 \times 32 \) cells, we would need a maximum of 3 additional BRAMs per NCA for this.

<table>
<thead>
<tr>
<th>Class</th>
<th>Window Size</th>
<th>Bins</th>
<th>Range</th>
<th>Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>64 × 128</td>
<td>9</td>
<td>0-180</td>
<td>L2-Hys</td>
</tr>
<tr>
<td>Car</td>
<td>104 × 56</td>
<td>18</td>
<td>0-360</td>
<td>L1-Sqrt</td>
</tr>
<tr>
<td>Stop Sign</td>
<td>64 × 64</td>
<td>18</td>
<td>0-360</td>
<td>L2-Hys</td>
</tr>
<tr>
<td>Bus</td>
<td>120 × 80</td>
<td>18</td>
<td>0-360</td>
<td>L1-Sqrt</td>
</tr>
<tr>
<td>Bicycle</td>
<td>104 × 64</td>
<td>18</td>
<td>0-360</td>
<td>L2-Hys</td>
</tr>
</tbody>
</table>

In terms of on-chip area, each NCA requires approximately 63 BRAMs, 80 DSPs, 15000 FFs and 22000 LUTs. Based on these numbers, and keeping 40% of area available for meeting Place-and-Route constraints, we can target up to 5 classes using the Virtex 7 690T.
We choose a driving-assist system as our choice of domain for a case-study in scalability here. Table 3.3 shows the HOG parameters used for detecting 5 of the 20 classes provided in the Pascal challenge relevant to this domain. As can be seen, each class requires different RoI sizes, orientations and block normalization. Figure 3.19 illustrates the multi-object pipeline that can be used for such a scenario. Figure 3.20 shows a stop-sign detected on a sample image based on the parameters chosen for training the SVM.

Figure 3.19 Scalable architecture for many-class object detection

Figure 3.20 Detecting classes other than pedestrian.
3.5 Discussion

In this section, we discuss a contrasting object recognition scheme and its trade-offs when compared to a sliding-window approach. In this scheme, the classification is decoupled from the detection and a generic object detection based on region proposals is used to feed the classification stage with candidates to classify. Thus, the probability of recognizing an object \( P(O) \) is given by Equation 3.1 where \( P(D) \) is the probability of detecting the object and \( P(C) \) is the probability of classifying it.

\[
P(O) = P(D) \times P(C)
\] (3.1)

The detection models propose a large number of candidate regions (over a 1000 at times), where different candidates have different confidence scores of having encompassed an object. Figure 3.21 shows the detections from three such generic models when evaluated on a couple of images from the INRIA test dataset. Figure 3.21(a) provides the groundtruth i.e. bounding box annotated with the class name. Figure 3.21(b) is a visual saliency model based on information maximization [71]. Figure 3.21(c) is another visual saliency model based on primate features like color, intensity and orientation [72]. Finally, Figure 3.21(d) is an objectness detector based on binary gradients [73].

As can be seen, these models are scalable and provide valuable hints to where most objects are in the image, but fail to hone in on the complete object regularly enough. We use the Intersection Over Union (IOU) metric [52] to quantify the detection capabilities. As per the standard, an IOU>0.5 is considered a valid detection. Since the INRIA annotations are tightly bound around the subject, we relax this constraint to 0.25 for visualization. We assume that we have a perfect classifier i.e. \( P(C) = 1 \) for [71] [73] and compare the results with our HOG-SVM scheme. It should be pointed out here that these generic detection schemes are used as-is for the purpose of comparison. These models are extremely performance-friendly and scalable to a large number of classes. With further algorithmic tuning, they can be deployed for application-specific tasks such as surveillance where a majority of objects that make their way into the visual frame demand attention.
Figure 3.21 (a1-a2) Groundtruth. (b1-b2) [71]. (c1-c2) [72]. (d1-d2) [73]. (e1-e2) HOG. Green (red) bounding boxes are valid (invalid) person detections based on an IOU>0.25.
In Chapter 3, a fast and efficient solution for object detection was proposed that could target up to five object classes given a constrained area and power budget. In this chapter, we present CASPER - a Context-Aware Scalable Pipeline for Efficient Recognition that addresses the challenges encountered when transcending a handful of object classes. To optimally tune our system, we decouple the visual recognition task into two phases - a detection phase which is capable of generating good regions of interest followed by a classification phase. The focus of this work is the later with an effort towards optimizing classification accuracy for a significantly large number of object classes. Figure 4.1 gives an overview of the pipeline.

**Figure 4.1** A Context-Aware Scalable Pipeline for Efficient Recognition.

Section 4.1 highlights a hardware-accelerated hierarchical vision pipeline targeted for fine-grained object recognition. Section 4.2 provides details of a Visual
Co-occurrence Network (ViCoNet), which is a Bayesian model used to encode context. In Section 4.3, we explain in detail the system-level operation of CASPER. Section 4.4 describes the methodology that we adopt to evaluate the utility of our context-based system. In Section 4.5, we conclude the chapter by comparing our results with other recognition schemes and discuss system level implications of our design too.

4.1 A Hierarchical Vision Pipeline

The vision pipeline is a two-tier classification system consisting of an HMAX accelerator coupled with a trained Regularized Least Squares (RLS) classifier followed by a HOG accelerator coupled with an ensemble of Exemplar Support Vector Machines (ESVMs). The efficacy of using such a hierarchical approach to large-scale object recognition was highlighted in [74] and integrated into an elaborate multi-algorithm framework [75].

4.1.1 HMAX

The first few hundred milliseconds of visual processing in primate cortex mostly follows a feedforward hierarchy and in [76], the authors proposed such a hierarchical visual object recognition model - HMAX - that consists of an image pyramid followed by four layers of alternating units of simple S and complex C units. This model was further refined in [11] where the inputs to the second S layer i.e. “S2” are sparsified. For further insights, we point readers to [77] where the claim made is that a hierarchical MAX-like operation is a key mechanism for object recognition in the cortex. HMAX while being computationally expensive is highly parallel and various accelerator designs have been proposed for embedded real-time applications [36, 63].

4.1.2 Exemplar SVMs

Exemplar SVM (ESVM) is a specific implementation of a Support Vector Machine (SVM) that is trained on a single positive example of a class, as opposed to many positives of that class. Each positive example is then called an exemplar of that class. In [18], the authors use HOG features in this ESVM framework and show that a collection of these exemplars prove to have high discriminative capabilities.
For brevity, during the rest of the paper, the HMAX-RLS system will be referred to as HMAX and the HOG-ESVM system as ESVM.

4.2 ViCoNet

Figure 4.2 Example of ViCoNet’s graph structure. Objects are represented as nodes and when seen together in a visual scene are connected by edges. The weight of an edge between any two nodes is the probability of those two objects seen together.

4.2.1 Model

Let $A$ and $B$ be two distinct types of items that are seen together. We can then model ViCoNet using Bayes’ theorem as given by Eq. 6.1 where $P(A)$ and $P(B)$ are the probabilities of $A$ and $B$ occurring independently and $P(A \cap B)$ is the joint probability of $A$ and $B$ occurring together.

\[
P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{4.1}
\]

Reorganizing Eq. 6.1 we get Eq. 6.2

\[
P(A \cap B) = P(B) \times P(A|B) \tag{4.2}
\]

Eq. 6.2 can be further simplified in terms of number of occurrences as Eq. 4.3 where $n(A)$ and $n(B)$ are the number of occurrences of item $A$ and item $B$ respectively in the set $S$ and $n(S)$ is the total number of items in the set $S$.  

38
\[ P(A \cap B) = \frac{n(B)}{n(S)} \times \frac{n(A)}{n(S) - 1} \] (4.3)

Figure 4.3 Four objects seen together in ViCoNet. The weight of a node is the probability of occurrence and the weight of an edge is the probability of co-occurrence in the network.

Every object \( O_i \) in a visual scene is thus represented as a node in ViCoNet with the node weight being \( P(O_i) \) and an edge between \( O_i \) and \( O_j \) signifies that \( O_i \) and \( O_j \) are seen together. The weight of the edge is \( P(O_i \cap O_j) \). Nodes with self-loops indicate that at least two instances of the same item exist in the network. Figure 4.3 shows an example of such a network where four items A, B, C and D are all seen with each other.

4.2.2 Operations

We envision ViCoNet to be able to operate in different modes: a learning mode, a context-building mode and a querying mode.

4.2.2.1 Learning

Learning can be static (offline) or dynamic (online) or both. For the purposes of this work, we constrain ViCoNet to learn only offline, i.e. relationships are built statically using a pre-annotated dataset. Every time a new object is seen in a visual scene, a new node is created in ViCoNet. As the object is seen with other objects, edges are formed and weights adapted. During this learning phase, we compute probabilistic weights of nodes and edges. Thus, once learning is complete, ViCoNet
represents a weighted undirected graph. Given a query, we can then determine which nodes are most likely to be seen together and more importantly how likely are they to be seen together.

4.2.2.2 Context-building

Context-building is fundamentally an online mode for ViCoNet where a fixed number of temporal predictions are used to build context about a particular visual scene. Conceptually, it is akin to walking into an unknown location blind-folded and once given visual access to the surroundings, being able to very quickly come up with a likelihood of the location. In this mode, no feedback is taken from ViCoNet, so it is imperative that this mode is fast, yet reliable.

4.2.2.3 Structure

Based on the proposed model and the envisioned operations, we structure ViCoNet as a three-pronged data-structure: one, an adjacency matrix of size $N \times N$ as shown in Eq. 4.4

$$E = \begin{bmatrix} e_{11} & e_{12} & \ldots & e_{1N} \\ e_{21} & e_{22} & \ldots & e_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ e_{N1} & e_{N2} & \ldots & e_{NN} \end{bmatrix} \quad (4.4)$$

where the value $e_{ij}$ is the probability of object $i$ and $j$ been seen together. Objects not connected to one another will have corresponding entries set to 0. Also, since it is as likely to see object $i$ with $j$ as it is to see object $j$ with object $i$, $E$ will be symmetric, so it would only be necessary to store the results in one diagonal half of the matrix.

ViCoNet also maintains node weights in a vector of size $N$ as shown in Eq. 4.5.

Each node weight is the probability of occurrence of an object in the graph.

$$V = \begin{bmatrix} v_1 \\ v_2 \\ \ldots \\ v_N \end{bmatrix} \quad (4.5)$$

Finally, ViCoNet keeps track of the current context by maintaining a context queue. Based on the contents of the queue, ViCoNet provides a probabilistic estimate of seeing each of the $N$ items. This queue is of significantly small length.
(m) compared to the number of items needed to be recognized using it. ViCoNet can also dynamically decide whether the stored context is stale and new context needs to be built by flushing the queue.

4.3 System Operation

As described in [74], the HMAX-ESVM pipeline operates as follows: Each test object (belonging to one of $N$ classes) is evaluated by HMAX, which produces a list of $k < N$ potential candidates. These $k$ candidates are then sent to ESVM to identify the target class. This two-tier hierarchy thus improves performance while also achieving higher accuracy.

To simulate our system, we encapsulate the aforementioned vision pipeline and ViCoNet in a larger test vehicle - CASPER - that can be thought of as a device containing the corresponding units. We envision users wearing such a smart-camera enabled device while shopping. We assume that a fraction of objects in a store will be of little to no interest to the shopper who likely has a distinct list of items to find. We would like to use these objects for ViCoNet’s context-building phase. As the user navigates through the store and stops to look at products, we would like to bring in context to the recognition task at hand. This is the querying phase of ViCoNet. Our test vehicle initializes into context-building mode and after a few predictions (also used to build context), enters into querying mode. An illustration of the underlying flow of information in these two phases can be seen in Figure 4.4.

![Figure 4.4 CASPER Operation](image-url)
4.4 Methodology

As shown in Figure 4.5, a retail store is an ideal example of perpetually-changing context since a user is constantly moving around different parts of the store. Thus, ViCoNet, having built context, needs to be capable of switching context dynamically so as to not drive the recognition pipeline into false predictions. For example, say a shopper is in the Cereal aisle, and based on predictions during context-building, ViCoNet provides a higher probability estimate of seeing Kix Cereal near Cheerios Cereal. Now, as the shopper moves into the Cleaning aisle, there is a drastic context-switch and ViCoNet needs to be able to quickly adapt to the new surroundings else the vision pipeline will be incorrectly informed about a higher likelihood of seeing Kellogs Cereal near Tide Laundry Detergent. Such context-switches are handled by ViCoNet by maintaining a record of the HMAX and ESVM confident scores. For every instance that all the entries of the context queue suggest low confidence based on these scores, a counter is incremented and if it is greater than \( m/2 \), the queue is flushed and ViCoNet switches to context-building mode.

![Figure 4.5](image.jpg)

Figure 4.5 Two users shopping in a retail store. User 1 (red) visits the left half of the store while user 2 (pink) visits the right half of the store.

4.4.1 Training

To begin with, we train ViCoNet using a set of 11 panoramic images taken at a local grocery store which have been annotated using the LabelMe tool \[78\]. Each image corresponds to a particular aisle in the store. As shown in Table 4.1, there are a total of 62 unique object classes annotated within these images. ViCoNet
is populated by reading in these annotations: one node in ViCoNet is created for each unique item name, and a count is kept of the number of appearances of each object. Each annotation additionally contains X and Y coordinates, so an edge is drawn between two nodes if they have been spatially seen together in the image, and a count is additionally kept of these appearances, allowing us to track what objects and object relationships are most frequent. This is the static learning phase of ViCoNet discussed earlier.

<table>
<thead>
<tr>
<th>Aisle</th>
<th>Number of Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cereal</td>
<td>4</td>
</tr>
<tr>
<td>Cleaning</td>
<td>8</td>
</tr>
<tr>
<td>Coffee</td>
<td>4</td>
</tr>
<tr>
<td>Condiments</td>
<td>5</td>
</tr>
<tr>
<td>Cookies</td>
<td>8</td>
</tr>
<tr>
<td>Dental</td>
<td>4</td>
</tr>
<tr>
<td>Juice</td>
<td>3</td>
</tr>
<tr>
<td>Pasta</td>
<td>2</td>
</tr>
<tr>
<td>Sauce</td>
<td>7</td>
</tr>
<tr>
<td>Soda</td>
<td>14</td>
</tr>
<tr>
<td>Storage</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>62</strong></td>
</tr>
</tbody>
</table>

Table 4.1 Distribution of object classes across aisles

To train the components of the vision pipeline, we obtained over 4800 images of 62 products ranging from Barilla Pasta to Tide Laundry from BING. More details about the training datasets can be found in Appendix A.3. We then created a dataset in PASCAL format so as to train classifiers using HMAX and HOG features. HMAX and HOG features were independently extracted from over 4800 images. A single RLS classifier was then trained using HMAX features, while the HOG features were used to train the ESVMs. A total of 2259 Exemplar models are formed using this training dataset. Thus, for $N = 62$, we would be running $E_{avg} = 37$ Exemplars per class to evaluate a particular test RoI. Using the baseline HMAX-ESVM framework this would mean running 740 ESVMs for $k=20$ which is computationally still a very daunting task.
4.4.2 Testing

Our testing happens on a sequences of images captured from a popular retail store (different from the one ViCoNet was learnt on) containing the 62 object classes we trained on. A total of 139 images, grouped into temporal sequences, are used for evaluation. Each group (scene) contains within it a number of products (RoIs) which have been annotated using the LabelMe toolkit [78]. Over 1000 RoIs are evaluated across all 62 classes, where each class consists of a minimum of 5 test RoIs. More details about the test dataset can be found in Appendix A.4.

4.5 Results

To evaluate and quantify the benefits of using ViCoNet, we first ran HMAX and ESVM independently on the test dataset. HMAX predicts the right class with an overall accuracy of around 55%, while ESVM gives around 72%. Figure 4.6 a) shows the accuracy of HMAX as a function of top $k$. For ViCoNet to be effective, it is imperative that during the context-building phase it is being provided with accurate predictions. We thus choose $k = 20$ corresponding to an expected accuracy of approximately 90%, i.e. 90% of the time HMAX includes a correct prediction in the list of $k = 20$ that it provides ESVM. We are thus able to maintain an overall accuracy of around 70% using this HMAX-ESVM pipeline with the advantage of not having to run exemplars for all $N = 62$ classes.

Having established a baseline, we now run the ViCoNet-enabled pipeline. At initialization, ViCoNet starts in context-building mode. We set ViCoNet’s context queue length to $m=8$ which experimentally proves to be optimum. Too short a queue does not allow enough context to be built and too long a queue degrades performance. A counter keeps track whether the context queue is full. It must be pointed out here that only high confident predictions are used to fill the queue. These confidence scores are obtained from ESVM classifications and we set the confidence threshold to $t=0.075$ so as to filter out the very low confidence predictions. After the first $m$ confident predictions from the visual pipeline, ViCoNet switches to querying mode. During the querying mode, a continuous stream of temporally related data is being evaluated and as new predictions are made, the context queue is updated in a first-in-first-out manner. As discussed earlier, there are sharp context switches.
at aisle changes. These are dynamically handled by ViCoNet by maintaining a record of the HMAX and ESVM confident scores and flushing the queue when a context-switch is detected. We ran 10 iterations of the simulations equivalent to 10 different users walking through the same store but navigating through different sections and using different snapshots for building context. ViCoNet achieves a 45% improvement in performance by paring down $k$ to 11 from 20. Along with these significant computational reductions, we achieve a small improvement in accuracy of 3%. The reduction in computational complexity is critical to support this recognition operation in embedded wearable camera platforms.

### 4.5.1 Context-Switches

As discussed earlier, ViCoNet handles context-switches by flushing the context queue and rebuilding context. We observe that during a spatially-related stream, sometimes a flush occurs just because of continuous low confidence from the HMAX-ESVM visual pipeline. To highlight the importance of detecting context-switches correctly, we ran the simulation for each spatial stream independently. For each stream, context was built starting with an empty queue. ViCoNet now achieves a 50% improvement in performance and a 7% improvement in accuracy. The reason behind this improvement is that the more flushes that are made, the more ViCoNet has to resort to context-building and thus more of the base pipeline is used.
Asymptotically, we are tending towards the baseline. So, ideally if a context switch is detected the optimum i.e. the exact number of times, then ViCoNet will be the most effective. To identify the ideal number of context switches, we computed the correlation coefficient of the first half of the queue with the second half for both static and dynamic context-switch runs of ViCoNet. Figures 4.7 a) and b) show the variation of the correlation coefficient during these runs. A constant correlation coefficient tending towards 1 indicate windows of constant context while a sudden drop in the correlation coefficient indicates that the queue was flushed. As can be seen the ideal case as a lot fewer fluctuations compared to the current dynamic version of ViCoNet.

(a) ViCoNet - Actual Context Switch  (b) ViCoNet - Ideal Context Switch

Figure 4.7 Correlation Coefficient captured for ViCoNet

To be able to improve the detections of context switches, other sensors such as the accelerometer and gyroscope, commonly available in most wearables, can be used along with visual context. For a planogram as shown in Figure 4.5, detecting sharp changes in orientation would be a good trigger for flushing the context queue. We recorded the readings from the orientation sensor of a popular wearable device for different walking directions - straight, left arc and right arc. As shown in Figure 4.8, the sensor is highly sensitive to changes in orientation and also has different signatures for different directions. This exposes a pathway for further improvement of dynamic context switches in ViCoNet.
4.5.2 Comparisons

We compared our performance with other state-of-the-art methods like Deep Convolutional Neural Network (DCNN). We trained a DCNN using our training data set described in Appendix A.3. It must be noted that our dataset is not large enough to train DCNNs from scratch. Therefore, we used an approach called fine-tuning which uses a model already trained using a very large dataset and then resumes training using a smaller dataset. This approach is known as an efficient way to get DCNNs trained with small datasets. More precisely, we used the Caffe [79] implementation as first proposed in [24], trained using 1.2 million images of ImageNet LSVRC-2010 [20] to classify 1,000 categories. We modified the output layer to classify our 62 target classes and then trained with the images in our training dataset. To accelerate the training process we use a modern GPU (NVIDIA Tesla M2070, 1.15GHz processor clock, 6GB GDDR5, 225W) [80]. We achieve around 60% total accuracy on our test dataset. We would expect the accuracy of the DCNN to improve with more training data and/or training the entire DCNN from scratch. Transfer learning is another technique that can be deployed when the target dataset is significantly smaller than the base dataset [81]. In general, using CNNs for any application requires huge amounts of training data and also an in-depth analysis of the structure, layers and depth of the CNN. However, exploring these options is beyond the scope of this work. It should also be noted that ViCoNet, as a context engine, is agnostic to the choice of the classifier and it should improve on any underlying classification system.

Conceptually, ViCoNet provides a gist of the current scene. We evaluated the
gist model of [40] in the context of aisles to compare how well it fares in this task. For the purpose of this evaluation, we took random snapshots of our panoramic aisle images and divided it into training and testing partitions. We extracted the gist feature from 30 such training images per aisle and trained a non-linear SVM using [70]. In this framework, gist provides a 51% accuracy of predicting an aisle correctly. Thus, a gist-enabled pipeline would suffer a drastic drop in accuracy on average compared to that achieved by ViCoNet. A comparison of the accuracy and the average number of classifiers needed for different variants of our pipeline is tabulated in Table 4.2. Figure 4.6 b) shows the per class accuracies for ViCoNet. Some of the classes like “26” (Fresca Soda Bottle), “41” (Mug Root Beer Soda Bottle) and “56” (Sierra Mist Soda Pack) have an accuracy of zero. This is because the base recognition pipeline is unable to produce high confidence scores for these classes compared to other class scores. Thus even if ViCoNet is able to lower the confidence of the other competing classes based on context, the target class prediction score is so low that the system fails. In such a scenario, we can use a more generic prediction like Soda rather than the fine-grained prediction to improve the overall efficacy of the system.

<table>
<thead>
<tr>
<th>System</th>
<th>Avg. Accuracy</th>
<th>Avg. No. of Classifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMAX</td>
<td>55%</td>
<td>1</td>
</tr>
<tr>
<td>ESVM</td>
<td>72%</td>
<td>62*Exemplars</td>
</tr>
<tr>
<td>HMAX-ESVM</td>
<td>70%</td>
<td>20*Exemplars</td>
</tr>
<tr>
<td>ViCoNet-Ideal</td>
<td>77%</td>
<td>10*Exemplars</td>
</tr>
<tr>
<td>ViCoNet</td>
<td>73%</td>
<td>11*Exemplars</td>
</tr>
</tbody>
</table>

Table 4.2 System Evaluation

4.5.3 Memory Overheads

We evaluated the storage overheads of ViCoNet based on the data structures used to support it. Each entry in the adjacency matrix $E$ and the self-occurrence vector $V$ uses 32 bits for storing the weights without loss of any data due to precision. The HMAX and ESVM scores in the context queue are represented using 8 bit precision. For our current dataset, we would need less than 10KB for $N=62$ object classes. Figure 4.9 outlines the scalability of ViCoNet as the number of objects increases. Based on this plot, we observe that less than 50MB is needed for up to
4000 objects.

Figure 4.9 Memory overheads for ViCoNet

4.6 Conclusions

In this chapter, the significance of context in rich multimedia temporal streams was highlighted. We used retail as our environment for evaluation due to the deep and diverse nature of classes and the drastic changes in the landscape of context it has to offer. A large dataset of 62 classes was created and components of our system were trained using thousands of retail images. Evaluations were then carried out on temporal image streams from real-world grocery aisles. To the best of our knowledge, this is the first work that fuses context with pixel-level features for an targeted for embedded object classification system.
Chapter 5  |  Hardware Emulation of Visual Recognition Pipeline

Accelerators are becoming mainstream in commercial products. Chipmakers are quickly ramping up their existing frameworks to enable accelerators to communicate easily with the general purpose processor. In an end-to-end vision application, these accelerators can take advantage of the huge parallelism available to improve performance while any latency-sensitive computation can still run on the processor. This chapter prototypes a dedicated hardware implementation of the proposed visual object recognition pipeline. The hardware-accelerated pipeline can be easily integrated with ViCoNet implemented as a multithreaded software program. Area and performance metrics are used as figure of merits to compare this solution with other competing strategies. Critical architectural insights such as workload characterization in a multi-accelerator heterogenous domain can also be drawn using this framework as a vision benchmark. For emulation purposes, we use a CAPI-enabled POWER8 machine as our platform which has multiple PCIe slots for FPGAs and/or GPUs as shown in Figure 5.1.

5.1 Coherent Accelerator Processor Interface

Chapter 3 showed the HOG-SVM accelerators in a traditional slave-style paradigm. As shown in Figure 5.2a, in this paradigm, data transfer between the accelerator and memory takes place via the PCIe interface. This introduces tremendous amounts of overhead moving data from the operating system’s virtual address space to the accelerator and back. As server workloads are becoming more computational
intensive, data transfer can become a major overhead. Accelerator-based designs are becoming ubiquitous and IBM’s latest POWER8 introduces a CAPI interface that allows the accelerators to be treated as peers in the system as shown in Figure 5.2b. Thus the accelerator can be treated as an additional hardware thread running alongside the other threads in the processor. This hardware thread in the form of an FPGA or a GPU can be reconfigured and customized based on the domain the server is deployed.

5.2 Emulation of Visual Recognition Pipeline

As discussed in Chapter 4, the pipeline is a hierarchical two stage system consisting of HMAX and HOG as features and RLS and SVM classifiers. HMAX is a two-layer structured CNN where the weights are fixed. The “S1” is a convolution layer, “C1” is a cross-scale pooling layer, “S2” is another convolution layer followed by “C2” - a max-pool layer. Earlier HMAX systems [36,82] have focused on accelerating “S2” - the compute bottleneck of HMAX, while running the other layers in software. In this work, the “C1” architecture outlined from [83] is verified and integrated into
HMAX and is the first end-to-end HMAX accelerator emulated on a single FPGA. The HOG-SVM stage is inherited from Chapter 3.

To exactly stress the impact of having reduced the workload of ESVM using ViCoNet, we synthesized, placed and routed HOG-SVM on a CAPI-enabled Xilinx Virtex 7 690T FPGA and observe that it can hold four SVM modules as shown in Figure 5.3a. Operating at a frequency of 100 MHz, assuming each RoI is of size $256 \times 256$ we are able to process all 2259 exemplars for the retail dataset at a throughput of 0.33 RoIs/second. Note that the prediction accuracy is 72%. Next, we mapped HMAX on a second 690T FPGA as shown in Figure 5.3b and we can achieve a net throughput of 1 RoI/second on a single FPGA, but the accuracy is 55%.

In the HMAX-ESVM pipeline, with $k=20$, the number of exemplars to be computed reduces significantly and the net throughput is now around 1 RoI/second, but we would need two FPGAs now rather than one. With the context model of ViCoNet we reduce $k$ further to 11 and with some minor changes to the
HMAX architecture, we can achieve an improvement in performance of around 1.9 RoIs/second. Figure 5.4 highlights an iso-area comparison of throughputs of these different system configurations. Given two FPGAs, with CASPER we can enjoy a throughput of close to 2 RoIs/sec (similar to HMAX), but more importantly we can achieve the best accuracy of 73%.

Figure 5.3 CASPER on a chip: (a) HMAX; (b) ESVM

5.3 Discussion on Convolutional Neural Networks

Using an Android-based library [84], we ran a Convolutional Neural Network (CNN) - “Alexnet” - onto a handheld device [85] enabled with a quad-core 1.2 GHz Cortex-A53, 1GB RAM and found that on average a classification takes between 1-2 seconds. The configuration of this phone is very similar to one of the
Figure 5.4 Throughput evaluations across various systems.

latest smart glasses in the market \cite{86} and we expect similar performance. From a memory point of view, this modest 8-layer CNN has over a million parameters. Each parameter can be stored using 2 bytes thus requiring around 200MB, which is around 25% of the RAM. It should be highlighted that very recent work use sophisticated compression techniques on “AlexNet” that reduces the weight storage by 35% without loss of accuracy \cite{87}. Deploying CNNs for any application requires huge amounts of training data and also an in-depth analysis of the structure, layers and depth of the CNN which is beyond the scope of this work.
Chapter 6  
Future Work

Machine vision systems are moving beyond the realm of academia into real-world systems and need to be resilient to extreme conditions such as poor lighting, low resolution, etc. This thesis highlighted retail and automobiles as two application spaces where vision systems can be deployed. Moving forward there are a number of other domains where these systems will penetrate and showcase their efficacy. In Section 6.1 we emphasize some of the key areas where the aforementioned system - CASPER - can be deployed. In Section 6.2 we look at some of the exploratory areas of research for vision-based systems.

6.1 Domain-specific Research

6.1.1 Robotics

While earlier work in robotics treaded carefully when depending on visual information for path planning, we are seeing a sudden growth in vision-based robotics. Algorithms like Visual Simultaneous Localization and Mapping are becoming increasingly popular in autonomous systems too and we will soon see the marriage of machine learning with machine kinematics that will spur the next leg of technology forward.

6.1.2 Surveillance and Security

Surveillance is of prime importance at airports, universities and other such public places. Deploying smart cameras enabled with real-time capabilities to quickly
identify threats is an important area of focus. While introducing wearables into our daily lives can make us more effective, there is always a risk of divulging important information making us vulnerable to cyber-threats. Thus, detection of such malware in the coming generation of “Internet of Things” is imperative and needs to be an important layer of the device stack.

6.2 Broad Topics of Research

6.2.1 Approximate Computing

So far we have explored the landscape of vision systems that enhance the performance and energy efficiency of the computational fabrics. In [90], the authors study some deep learning models and found that distortions like blur and noise reduce their fidelity. Thus, as system architects we need to understand the tradeoffs in performance and accuracy while designing such systems. In this section we explore the potential savings in computational work needed to be done while not compromising on accuracy. In embedded systems, compute resources come at a cost. Saving a few resources can enable fitting a design in a particular form-factor or may cause the design to overflow into the next larger generation of devices.

![Classification Rates (%)](image)

**Figure 6.1** HMAX resilience to errors. Six classes from CalTech101 were used. We observe a roll-off in accuracy beyond 512 pixels in error.
Image reconstruction is an important processing technique in image processing and computer vision applications. Most object recognition algorithms use a multi-scale pyramid to make it scale invariant. For example, HMAX uses an image pyramid having 11 scales (including base scale of size $256 \times 256$) with a scale factor of $2^{1/4}$ and uses a bicubic interpolation technique to generate the image pyramid. The input image is first converted to grayscale and then passed through this image pyramid before computing the “S1” layer of HMAX.

![Diagram](image.png)

**Figure 6.2** Interpolation techniques. In (a), four while in (b), 16 neighboring pixels are used for interpolating the value of the pixel at $(x', y')$.

Many architectures have been proposed to support linear and non-linear interpolation techniques [91]. Given a pixel at $(x, y)$ with intensity $I(x, y)$, an interpolated pixel at $(x', y')$ at an offset of $(\Delta x, \Delta y)$ from $(x, y)$ (where $0 < \Delta x, \Delta y < 1$) can be computed using bilinear interpolation. By definition, this involves computing two linear interpolations in the $x$ direction and one linear interpolation in the $y$ direction [92]. This process, derived from first principles, is shown in (6.1) and requires eight multiplications. Figure 6.2(a) illustrates an interpolated pixel using four neighboring pixels.
\[ I(x', y') = I(x, y) \times (1 - \Delta x) \times (1 - \Delta y) + \]
\[ I(x + 1, y) \times \Delta x \times (1 - \Delta y) + \]
\[ I(x, y + 1) \times (1 - \Delta x) \times \Delta y + \]
\[ I(x + 1, y + 1) \times \Delta x \times \Delta y \]

(6.1)

Using bicubic interpolation, the same interpolated pixel \( I(x', y') \) is given by (6.2) where \( R_c \) denotes a bicubic interpolation function [93]. The computation requires 56\(^1\) multiplications in all and Figure 6.2(b) shows the interpolated pixel using 16 neighboring pixels.

\[ I(x', y') = \sum_{m=-1}^{2} \sum_{n=-1}^{2} I(x + m, y + n) R_c(m - \Delta x) R_c(-(n - \Delta y)) \]

(6.2)

We explored the capability of HMAX to correctly recognize objects using bilinear interpolation in the image pyramid. We used all 101 classes of CalTech101 for this purpose. It should be noted that using the original bicubic interpolation technique, we achieve 54% accuracy on the said dataset. This is in confirmation with the results shown in [11]. We then ran the experiment using bilinear interpolation and found the impact of this is a 1% loss in accuracy. Thus, if the image pyramid is implemented using bilinear interpolation, we would need eight multipliers instead of 56 multipliers. We tabulate our results in Table 6.1. These savings have a significant impact on area and performance of the accelerated system.

<table>
<thead>
<tr>
<th>System</th>
<th>Algorithm</th>
<th>Accuracy</th>
<th>Multipliers</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMAX</td>
<td>Bicubic</td>
<td>54%</td>
<td>56</td>
</tr>
<tr>
<td>HMAX</td>
<td>Bilinear</td>
<td>53%</td>
<td>8</td>
</tr>
</tbody>
</table>

### 6.2.2 Multi-sensor Fusion

Depth cues are an integral part of any vision task. Current depth sensing in outdoor environments is done via active LIDAR devices [94]. New depth sensors such as...\(^1\)Further optimizations are possible using shift and add operators but this would still require significantly more multipliers than the bilinear version.
passive light-field cameras [95], active cameras such as Kinect 2 [96] or near-field cameras such as RealSense [97] can be utilized to refine the depth estimation, which can solve the segmentation problem in region of interest generation. Integrating other temporal sensor information such as ultrasonics can help in validating potential obstacles when building autonomous vehicles. In Chapter 4 we have already demonstrated the utility of monitoring the gyroscope and the accelerometer on a wearable device. A holistic approach to combining different sensor information for inference will make the wearables of the future more effective.
Chapter 7  
Summary

In this chapter we summarize the contributions of this thesis.

An embedded system for large-scale object recognition targeted for wearables is presented. Domain-specific constraints were considered in the design-space exploration of computer vision and machine learning algorithms. Two domains, namely retail and the automobile industry were considered where such an intelligent wearable visual aid could be deployed. Relevant data occurring in these two domains was collected and annotated. A simulation environment was created where evaluations were conducted both in terms of accuracy and performance. Since real-time is a critical constraint, bottlenecks in the pipeline were accelerated. The hardware accelerated pipeline was then emulated on a computing platform that treats these accelerators as peers rather than slaves in the system. Recent works in embedded vision study the implications of coupling vision accelerators with light-weight cores in a heterogeneous system-on-chip [56]. This work would orthogonally augment such architecturally rich design-space evaluations. This work will also serve as a pathway to benchmark the benefits of emerging low power devices such as coupled oscillators.
Appendix A
Datasets

A.1 Inria Person Dataset

The Inria Person dataset was introduced with [14] and is publicly available at
http://lear.inrialpes.fr/data/ Figure A.1 shows a few sample positives from
the training images.

![Training positive examples from the INRIA Person Dataset.](image)

Figure A.1 Training positive examples from the INRIA Person Dataset.
Negatives provided are from natural scenes not containing any people.

A.2 Grocery Dataset

This dataset consists of 62 different classes focused on objects that can be found in
a grocery store. All the data was annotated using [78]. There are three sources to
build this dataset, each used for a specific purpose as outlined below.
A.2.1 Wegmans

A set of 11 panoramic images was taken at Wegmans - a local grocery store - where each image corresponds to a particular aisle in the store. Figure A.2 shows the Cleaning aisle. The data from such an aisle is then used to train ViCoNet as outlined in Chapter 3.

Figure A.2 Panoramic image of an aisle captured at a Wegmans store.

A.2.2 BING

Classifiers trained on features extracted from images need a large amount of data to train them. To train our feature-based vision pipeline, we obtained many images of products ranging from Barilla Pasta to Tide Laundry from BING. A few candidate examples are shown in Figure A.3. We then created a dataset in PASCAL format so as to train classifiers using HMAX and HOG features. Overall 4842 images across 62 products were used for training.

Figure A.3 A few examples of our vision pipeline training dataset. A total of 62 products were obtained from BING. There is inter-class as well as intra-class variance in terms of shape, size and color in the dataset.
A.2.3 Sophie

For testing purposes, we used data taken in a different store - Safeway - based in California. Figure A.4 shows an example of data captured in the Cleaning aisle of the store. 1019 RoIs extracted from 139 image frames are evaluated across all 62 classes, where each class consists of a minimum of 3 test RoIs. As can be seen, some of these products tend to be occluded, some further behind in the shelf thus having poor lighting. This makes it a fairly challenging dataset to test on. Figure A.5 shows the distribution of the data across all classes.

![Figure A.4](image1)

(a) Frame 34  (b) Frame 96  (c) Frame 134  

(d) Test Stream

**Figure A.4** High Definition video sequences were captured in different aisles of a Safeway store based in California. (a), (b) and (c) are frames extracted from a sequence taken in the Cleaning aisle. This temporal stream (d) is injected into our testing framework.

![Figure A.5](image2)

**Figure A.5** Distribution of Test Data
Appendix B  
Publications

B.1 Refereed Conferences


- Matthew Cotter, Siddharth Advani, Jack Sampson, Kevin Irick, and Vijaykrishnan Narayanan, "A Hardware Accelerated Multilevel Visual Classifier


B.2 Refereed Journals


Acoustics, Speech and Signal Processing, IEEE Transactions on, vol. 29, no. 6, 

[94] M. M. Aasted, R. J. Dise, T. A. Baugher, J. R. Schupp, P. H. Heinemann, and 
Louisville, Kentucky, August 7-10, 2011. American Society of Agricultural 
and Biological Engineers, 2011, p. 1.


Vita

Siddharth Advani

Siddharth Advani is a Ph.D. student in the department of Computer Science and Engineering at the Pennsylvania State University. He joined the department in 2011 and is co-advised by Prof. Vijaykrishnan Narayanan and Prof. Jack Sampson in the Microsystems Design Laboratory.

His research is focused on developing real-time design flows for visual assist systems. Specifically, as part of his Ph.D. thesis, he looked at object detection and classification using traditional computer vision features and developed a low power object detection/recognition pipeline targeted for wearables [ICCAD 2014, FPL 2015]. He then proposed a Bayesian graphical model to encode visual context in a scene that can enable this image recognition pipeline to make smarter predictions [ESTIMEDIA 2015]. He also looked at some key properties (like resiliency to noise and approximate computing) that vision systems have to offer [INIS 2015].

During the course of his Ph.D., he also had the opportunity to collaborate with people from academia (UCLA and USC) and industry (IBM, Intel, SiliconScapes and Toshiba). He interned at Intel Labs between June 2013 and December 2013 with the Microarchitecture Research Lab headed by Hong Wang, where he was supporting the team in enabling their SoCs and also prototyped some vision accelerators on Xilinx FPGAs. As part of service to the community, he has served as a technical reviewer for journals and conferences including IEEE TCAD, IEEE TIM and DAC. He has also been involved in numerous outreach activities such as the Exploration U - an annual STEM event at State College and the Young Scholars Program at Penn State.