Intelligent Vision Systems: Exploring the State-of-the-Art and Opportunities for the Future

Siddharth Advani*, Srinidhi Kestur*,‡, Vijaykrishnan Narayanan*
*The Pennsylvania State University, USA
‡Broadcom Corporation

***Abstract*— Vision and video applications are becoming pervasive in mobile and embedded systems. Consumer wearable devices require capabilities for real-time video analytics and prolonged battery lifetimes, which is further driving the need for innovative system designs with low-power, reliability and high performance. Further, the increasing resolution of image sensors in these mobile systems places an increasing demand on both the memory storage as well as the computational power. Such stringent requirements have given rise to accelerator-rich architectures in system-on-chips, where the primary computational burden is handled by dedicated hardware accelerators. In this paper we provide an overview of the current state-of-the-art in vision accelerators. We further discuss the opportunities to improve energy efficiency by minimizing Dynamic Random Access Memory (DRAM) refreshes and explore techniques to exploit algorithmic resilience for reduction in compute units while maintaining reliable system accuracy and performance.**

I. Introduction

The workings of the brain have intrigued researchers across various spectrums of science - from neuroscience to computer science. While the cortex still remains an enigma to the community, the visual cortex is a more finely understood system and many mathematical models mimicing the human visual system (HVS) have been proposed. Some of the early work in vision focused on understanding how primal capabilities of vision trigger higher modalities such as object recognition [1]. Progressively, object recogntion models based on the simple and complex cells in the cortex were developed [2]. To understand task-driven vision, attention models using salient features of a visual scene were proposed [3], [4]. More recent work focuses on understanding the impact of attention under the influence of multiple cues [5].

Most of these vision models are computationally intensive that require frequent accesses to memory due to large matrix operations that run either in a feed-forward or an iterative manner. Running these workloads in real-time is a necessary constraint that needs to be met by the underlying system, but is becoming a daunting challenge with increasing resolutions of display panels coupled with improved camera sensors.

Given the challenges and opportunities in neuromorphic computing, many have embarked upon developing systems for smart vision applications. Synopys recently launced EV544 - a Convolutional Neural Network (CNN) based processor [6]. The SpiNNaker project has evolved from being a massively parallel representation of the human brain to now being used as a tool to further advance studies in neuroscience and robotics [7]. In a similar league, the True North chip [8] meanders away from the traditional von Neumann architecture and uses 4096 parallel and distributed cores is an event-driven framework for solving problems in vision and audition.

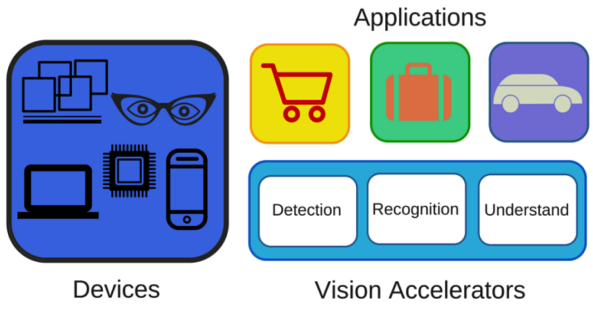


Fig. 1. Application-specific vision accelerator pipeline.

Fig. 1 illustrates the interaction between compute devices and vision accelerators when targeting various applications. A common vision pipeline involves parsing the visual scene and extracting objects or regions of interest (RoIs). This is carried out in the object detection stage. Once regions are extracted they are sent to a recognition stage to identify what the object is. Having figured out whether the object is of interest, further options can be explored. For example, if the object is a person, activity or pose estimation can be triggered. The application workload usually will decide the choice of the compute device. For example, if a user is in a retail store and would like to use a smart visual-assist device, a wearable small form-factor device would be ideal. However, if this is an automotive-assist system, a larger device may be engaged. If a security application is being deployed at an airport, then a large server-scale architecture would be needed to handle the sheer volume of data being generated every minute.

The main contributions of this paper are:

- To usher in the next wave of technology, we explore the current state-of-the-art in embedded vision accelerators and lay emphasis on key insights when designing such accelerators.
- With scaling technology paving the way for approximate computing, we exploit an increasingly powerful property of most vision algorithms - reliability to noise. We show that for an object recognition system using DRAMs for memory storage, we can reduce refresh rates by $4\times$ thus reducing refresh energy. We can also reduce the number of multipliers by $7\times$ in the preprocessing stage of the system. We maintain a 1% error bound on accuracy while exploiting reliability of the system.

The rest of this paper is organized as follows: In Section II, we provide an overview of vision-based architectures and the corresponding state-of-the-art. Section III describes a robust object recogntion pipeline. Finally, we conclude with Section IV.

II. Vision Accelerators

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) [9] 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 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. 3 shows the faces detected when the face detection algorithm was run on an image from the popular PASCAL Visual Object Classes (VOC) Challenge dataset [10].

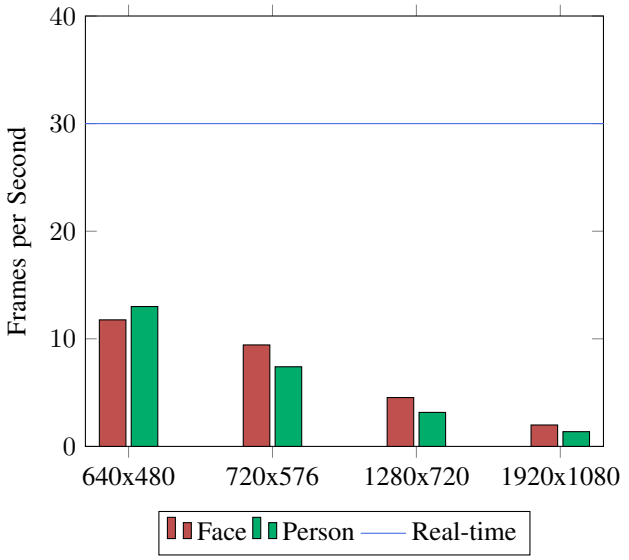


Fig. 2. Frames per second for face (red) and person (green) detections. Scale factor of 1.1 was used for face detection and 1.05 for person detection.

Digital signal processors (DSPs) have been a universally accepted alternative to general purpose CPUs for seamless multimedia processing. The Qualcomm Hexagon DSP instruction set architecture (ISA) contains numerous special-purpose instructions designed to accelerate key multimedia kernels such as sliding window filters [11]. Heterogeniety has often been considered as a viable solution to handle the increasingly varying nature of workloads that need to be run today [12]. 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

Fig. 3. Face detection algorithm results on an image from PASCAL dataset.

many small cores for low power and high throughput coupled with custom hardware accelerators was designed in [13]. In [14], the authors explored architectural heterogeniety by using customized data-flows for many vision-based applications targeted at retail, security, etc.

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, [15] proposed a GPGPU-accelerated intelligent learning model. [16] 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 computationaly intensive tasks such as face detection [17], pedestrian detection [18], object recognition [19] and object detection [20]. In [21], the authors propose a benchmarking suite - VISBench (Visual, Interactive, Simulation Benchmarks) - and find that a MIMD rather than a SIMD architecture gives better performance.

Even though Convolutional Neural Networks (CNNs) were explored in the early 1990s for vision applications [22], 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 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 [23]. This CNN would need more than 80 million operations and over 100,000 data transfers [24].

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. The reader is pointed to [25] for insights on deep architectures in general and [26] for CNN-based learning and their recent advances.

From a systems perspective, [27] mapped an earlier Convolutional Network based face-detection task onto custom hardware. [28] showed that a dynamically configurably coprocessor executing VLIW instructions can achieve significant speedup over CPU, GPU and FPGAs for a CNN-based architecture. More recently, [29] 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. [30] furthered this by proposing a training and inference accelerator capable of providing GPU-like bandwidth in ASIC-like power budgets.

III. Leveraging Reliability in Vision

Today, reliability is being explored at different layers of abstraction; from devices to memory to algorithms. At a circuit-level, [31] uses a conditional probability approach for modeling reliability in combinational circuits. Approximate computing is being considered one of the potential pathways in reducing energy inefficienices in applications that can tolerate some amount of error. [32] uses scalable effort hardware for the design of efficient hardware implementations for algorithms that demonstrate inherent error resilience. In [33], the authors propose an inexact neural network accelerator with significant energy savings. In this section, we evaluate the capabilities of a bio-inspired visual object recognition algorithm - HMAX - and leverage its potential to save power and reduce computational resources.

A. HMAX

The first few hundreds milliseconds of visual processing in primate cortex mostly follows a feedforward hierarchy and in [34], the authors proposed such a hierarchical visual object recognition model - HMAX - that consists of four layers of alternating units of *simple* **S** and *complex* **C** units. This model was further refined in [2] where the inputs to the second S layer i.e. "S2" are sparsified. For further insights, we point readers to [35] 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], [19]. All our experiments in this work are based on *hmin* - a publicly available representation of HMAX [37].

B. Exploiting Resiliency for Power Benefits

So far we have surveyed the landscape of vision systems that enhance the performance and energy efficiency of the computational fabrics. However, memory is an integral part of most systems today and contributes between 10-30% of the overall power of embedded video systems and mobile phones [38]. Propeilled by the growth in big data, the demand for cloud storage is allowing for larger DRAM devices to be used today. However data in every DRAM cell needs to be refreshed periodically to maintain integrity. This is done by reading a row and writing it back from the sense amplifier. JEDEC standards [39] define a 64 ms refresh window in which a maximum of 8192 refresh commands are issued via an auto-refresh command by the memory controller. The increasing memory size has made DRAM memory refresh energy a matter of growing concern and new power-efficient techniques such as Low Power Auto Self Refresh, Temperature Controlled Refresh, Refresh Pausing, Fine Granularity Refresh and Data Bus Inversion have been introduced in recent memory standards such as DDR4 [39].

Modulating DRAM refresh based on the data characteristics has been proposed as early as 1998 [40]. In [41], the authors looked at reducing refresh power on multimedia workloads. Recently, in [42], the authors showed that refresh power can dynamically be changed based on autonomously tagging data with logical labels. Many recent works have looked at tackling the increasing refresh power in other different ways. [43] used software techniques while in [44], scheduling policies were proposed, and [45] used a refresh pausing mechanism to exploit idle memory cycles. Very recent work suggests a Flexible Auto-Refresh scheme to skip unneeded refreshes in DRAMs [46].

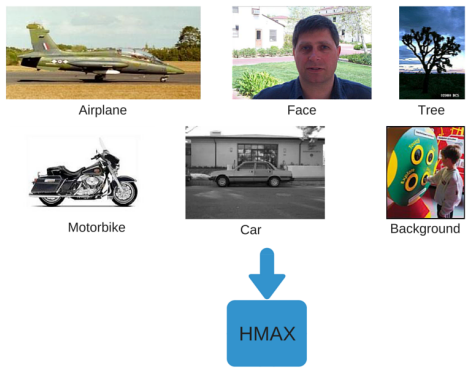
In this section, we showcase a data-dependent method to reduce refresh rates. More specifically, we explore the resiliency of HMAX to bit errors, which can then be used to choose the refresh rate for DRAM-based memory storage. As shown in Fig. 4(a), we choose six classes from the CalTech101 dataset [47] and obtain a recognition accuracy of around 78%. Next we introduce salt and pepper noise into 128 pixels of each testing image as shown in Fig. 4(b). This noise corresponds to 1024 cell failures in a 32GB DRAM device [43]. We find that the accuracy drops by less than 1%. Based on the cumulative cell failure probability curve provided in [43], this implies that by increasing the refresh period from 64ms to 256ms we can expect up to 1000 cell failures in a 32GB DRAM device. Under these constraints, HMAX will still maintain recognition rates within a 1% error bound, thus helping in reducing the overall refresh energy.

Fig. 5 illustrates the classification accuracy of HMAX as a function of the pixel errors introduced in each image. We find that HMAX is resilient to up to 512 pixel errors after which there is a roll-off in accuracy. We also inspected the accuracy function for each of the six classes independently and found that each class had its own sensitivity curve. This observation suggests that having a data-dependent differential refresh scheme as proposed in [42] would help in further reducing refresh energy.

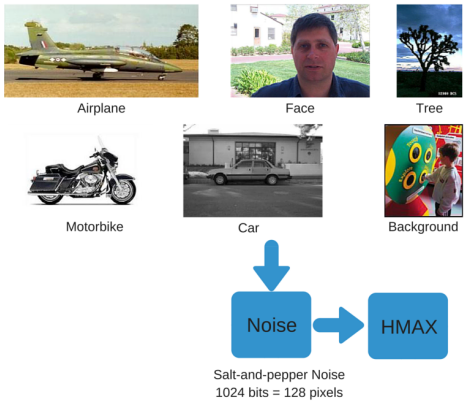
C. Exploiting Resiliency for Compute Benefits

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.

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 256×256) with a scale factor of $2^{1/4}$ and uses a bicubic



(a) Baseline



(b) Noise

Fig. 4. HMAX resilience to errors. Six classes from CalTech101 were used. We use (a) as our baseline. Since HMAX operates only on grayscale image, we convert each image to grayscale (8 bits) before processing. In (b) we introduce salt and pepper noise in 128 pixels as shown.

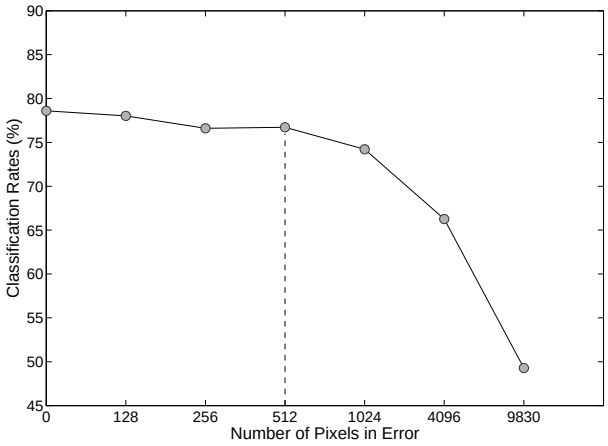


Fig. 5. HMAX resilience to errors. Six classes from CalTech101 were used. We observe a roll-off in accuracy beyond 512 pixels in error.

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.

Many architectures have been proposed to support linear and non-linear interpolation techniques [48]. 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 [49]. This process, derived from first principles, is shown in (1) and requires eight multiplications. Fig. 6(a) illustrates an interpolated pixel using four neighboring pixels.

$$\begin{aligned} I(x', y') = & I(x, y) \times (1 - \Delta x) \times (1 - \Delta y) + \\ & I(x + 1, y) \times \Delta x \times (1 - \Delta y) + \\ & I(x, y + 1) \times (1 - \Delta x) \times \Delta y + \\ & I(x + 1, y + 1) \times \Delta x \times \Delta y \end{aligned} \tag{1}$$

Using bicubic interpolation, the same interpolated pixel $I(x', y')$ is given by (2) where R_c denotes a bicubic interpolation function [50]. The computation requires 56[1] multiplications in all and Fig. 6(b) shows the interpolated pixel using 16 neighboring pixels.

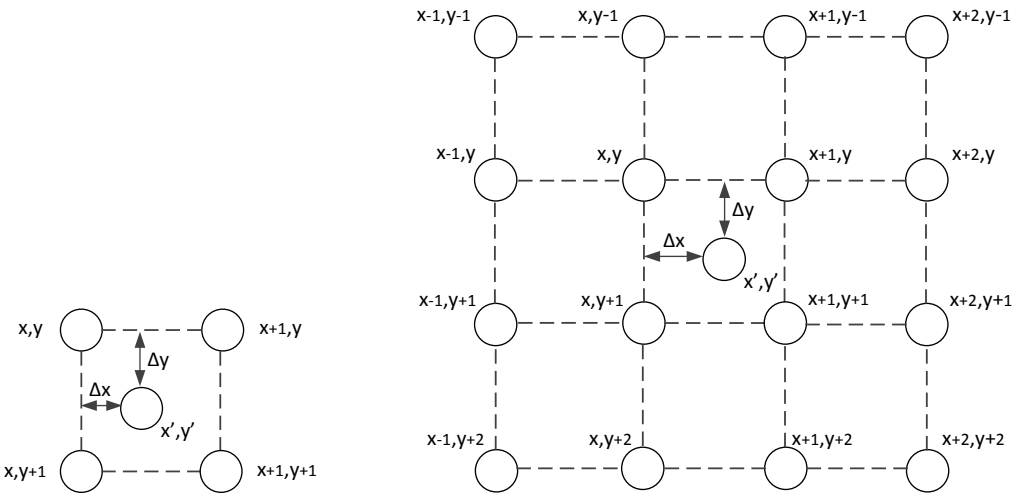
$$I(x', y') = \sum_{m=-1}^{2} \sum_{n=-1}^{2} I(x+m, y+n) R_c(m - \Delta x) R_c(-(n - \Delta y)) \tag{2}$$

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 [2]. 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 I. These savings have a significant impact on area and performance of the accelerated system.

TABLE I. IMPACT OF INTERPOLATION TECHNIQUES

System	Algorithm	Accuracy	Multipliers
HMAX	Bicubic	54%	56
HMAX	Bilinear	53%	8

[1]Further optimizations are possible using shift and add operators but this would still require signficantly more multipliers than the bilinear version.



(a) Bilinear Interpolation

(b) Bicubic Interpolation

Fig. 6. Interpolation techniques. In (a), four while in (b), 16 neighboring pixels are used for interpolating the value of the pixel at (x',y').

IV. Conclusion

In this paper we extensively survey a plethora of vision-based systems targeted for real-time applications. With shrinking technology, reliability will become exceedingly challenging and approximate computing will have more of a role to play when designing compute systems. We exploit the inherent robustness in vision algorithms and show that it can help to reduce both power and compute resources, both of which are valuable when designing such systems. Future work includes evaluation of 3D stacked DRAM technology for reliable vision applications.

Acknowledgements

This work is supported in part by NSF Expeditions: Visual Cortex on Silicon CCF 1317560. The work is also supported through infrastructure provided by NSF Award 1205618.

References

[1] D. Marr, "Early Processing of Visual Information," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 275, no. 942, pp. 483–519, 1976.

[2] J. Mutch and D. G. Lowe, "Object Class Recognition and Localization using Sparse Features with Limited Receptive Fields," *IJCV*, 2008.

[3] N. D. B. Bruce and J. K. Tsotsos, "Saliency Based on Information Maximization," *Advances in Neural Information Processing Systems*, vol. 18, pp. 155–162, 2006.

[4] L. Itti and C. Koch, "Computational Modelling of Visual Attention," *Nature Reviews Neuroscience*, 2001.

[5] M. Bay and B. Wyble, "The Benefit of Attention is not Diminished when Distributed Over Two Simultaneous Cues," *Attention, Perception, and Psychophysics*, vol. 76, no. 5, pp. 1287–1297, 2014.

[6] J. Campbell and V. Kazantsev, "Using an Embedded Vision Processor to Build an Efficient Object Recognition System," in *DesignWare IP White Papers*, May 2015.

[7] S. Furber, F. Galluppi, S. Temple, and L. Plana, "The SpiNNaker Project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[8] A. Cassidy *et al.*, "Real-Time Scalable Cortical Computing at 46 Giga-Synaptic OPS/Watt with 100x Speedup in Time-to-Solution and 100,000x Reduction in Energy-to-Solution," in *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*, Nov 2014, pp. 27–38.

[9] OpenCV, http://opencv.org/.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, June 2010.

[11] L. Codrescu *et al.*, "Hexagon DSP: An Architecture Optimized for Mobile Multimedia and Communications," *Micro, IEEE*, vol. 34, no. 2, pp. 34–43, Mar 2014.

[12] E. Chung, P. Milder, J. Hoe, and K. Mai, "Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs?" in *Microarchitecture (MICRO), 2010 43rd Annual IEEE/ACM International Symposium on*, Dec 2010, pp. 225–236.

[13] R. Iyer *et al.*, "CogniServe: Heterogeneous Server Architecture for Large-Scale Recognition," *Micro, IEEE*, May 2011.

[14] N. Chandramoorthy *et al.*, "Exploring Architectural Heterogeneity in Intelligent Vision Systems," in *International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2015.

[15] A. Nere, A. Hashmi, and M. Lipasti, "Profiling Heterogeneous Multi-GPU Systems to Accelerate Cortically Inspired Learning Algorithms," in *IPDPS*, 2011.

[16] S. Kestur, D. Dantara, and V. Narayanan, "SHARC : A Streaming Model for FPGA Accelerators and its Application to Saliency," in *Design, Automation, and Test in Europe*, 2011.

[17] D. Hefenbrock, J. Oberg, N. Thanh, R. Kastner, and S. Baden, "Accelerating Viola-Jones Face Detection to FPGA-Level Using GPUs," in *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on*, May 2010, pp. 11–18.

[18] A. Suleiman and V. Sze, "Energy-efficient HOG-based Object Detection at 1080HD 60 fps with Multi-Scale Support," in *IEEE Workshop on Signal Processing Systems*, Oct 2014.

[19] A. Maashri *et al.*, "Accelerating Neuromorphic Vision Algorithms for Recognition," in *DAC*, 2012, pp. 579–584.

[20] S. Bae *et al.*, "An FPGA Implementation of Information Theoretic Visual-Saliency System and Its Optimization," in *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, May 2011.

[21] A. Mahesri, D. Johnson, N. Crago, and S. Patel, "Tradeoffs in Designing Accelerator Architectures for Visual Computing," in *Microarchitecture,*

2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on, Nov 2008, pp. 164–175.

[22] S. Lawrence, C. Giles, and A. C. Tsoi, "Convolutional Neural Networks for Face Recognition," in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, Jun 1996, pp. 217–222.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[24] XCell, "Machine Learning in the Cloud: Deep Neural Networks on FPGAs," Available: http://issuu.com/xcelljournal/docs/xcell_journal_issue_92/46?e.

[25] Y. Bengio, "Learning Deep Architectures for AI." Now Publishers, 2009.

[26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," in *Nature*, May 2015, pp. 436–444.

[27] C. Farabet, C. Poulet, J. Han, and Y. LeCun, "CNP: An FPGA-based processor for Convolutional Networks," in *International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2009.

[28] S. T. Chakradhar, M. Sankaradass, V. Jakkula, and S. Cadambi, "A Dynamically Configurable Coprocessor for Convolutional Neural Networks," in *ISCA*, 2010, pp. 247–257.

[29] T. Chen, J. Wang, Y. Chen, and O. Temam, "DianNao : A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning," in *ASPLOS*, 2014.

[30] Y. Chen *et al.*, "DaDianNao: A Machine-Learning Supercomputer," in *MICRO*, Dec 2014, pp. 609–622.

[31] C. Chen and R. Xiao, "A Fast Model for Analysis and Improvement of Gate-Level Circuit Reliability," *Integration, the VLSI Journal*, 2015.

[32] V. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. Chakradhar, "Scalable Effort Hardware Design: Exploiting Algorithmic Resilience for Energy Efficiency," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, June 2010, pp. 555–560.

[33] Z. Du, A. Lingamneni, Y. Chen, K. Palem, O. Temam, and C. Wu, "Leveraging the Error Resilience of Neural Networks for Designing Highly Energy Efficient Accelerators," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 34, no. 8, pp. 1223–1235, Aug 2015.

[34] T. Serre, L. Wolf, and T. Poggio, "Object Recognition with Features Inspired by Visual Cortex," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, June 2005, pp. 994–1000 vol. 2.

[35] M. Riesenhuber and T. Poggio, "Hierarchical Models of Object Recognition in Cortex," *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.

[36] S. Kestur *et al.*, "Emulating Mammalian Vision on Reconfigurable Hardware," in *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, April 2012.

[37] J. Mutch, http://cbcl.mit.edu/jmutch/hmin/.

[38] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," in *Usenix Annual Technical Conference*, 2010.

[39] "JEDEC DDR3 and DDR4 SDRAM Standard," 2012.

[40] T. Ohsawa, K. Kai, and K. Murakami, "Optimizing the DRAM Refresh Count for Merged DRAM/logic LSIs," in *ISLPED*, 1998.

[41] S. Liu, Pattabiraman K., T. Moscibroda, and B. Zorn, "Flikker: Saving DRAM Refresh-power through Critical Data Partitioning," in *ASLPOS*, 2011.

[42] S. Advani *et al.*, "Refresh Enabled Video Analytics (REVA): Implications on Power and Performance of DRAM Supported Embedded Visual Systems," in *Computer Design (ICCD), 2014 32nd IEEE International Conference on*, Oct 2014, pp. 501–504.

[43] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.

[44] J. Stuecheli, D. Kaseridis, H. Hunter, and L. John, "Elastic Refresh: Techniques to Mitigate Refresh Penalties in High Density Memory," in *MICRO*, 2010.

[45] P. Nair, C. Chou, and M. Qureshi, "A case for Refresh Pausing in DRAM memory systems," in *HPCA*, 2013.

[46] I. Bhati, Z. Chishti, S.-L. Lu, and B. Jacob, "Flexible Auto-refresh: Enabling Scalable and Energy-efficient DRAM Refresh Reductions," in *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, 2015, pp. 235–246.

[47] R. Fergus and P. Perona, "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories," in *IEEE Conference on Computer Vision and Pattern Recognition Workshop on Generative-Model Based Vision*. Ieee, 2004, pp. 178–178.

[48] S. Kestur, K. Irick, S. Park, A. Al Maashri, V. Narayanan, and C. Chakrabarti, "An Algorithm-Architecture Co-Design Framework for Gridding Reconstruction using FPGAs," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, June 2011, pp. 585–590.

[49] J. Russ, "The Image Processing Handbook." CRC Press, 2011.

[50] R. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 29, no. 6, pp. 1153–1160, Dec 1981.