International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

A REAL-TIME OBJECT DETECTION PROCESSOR USING AN XNOR-BASED VARIABLE-PRECISION COMPUTING UNIT ON FPGA

Mrs. P. MONIKA KANTHI[1], RAGALA KARTHIK[2]

[1]Assistant Professor, Department of Electronics and Communication Engineering, Ellenki College of Engineering and Technology, Patelguda, Patancharu, Hyderabad, Telangana.
smonikakanthi@gmail.com

[2]M.Tech Student, Department of Electronics and Communication Engineering (Embedded Systems), Ellenki College of Engineering and Technology, Patelguda, Patancharu, Hyderabad, Telangana.

Abstract - Convolutional neural networks (CNNs) achieve strong accuracy in object detection, but their heavy computation and memory demands make them difficult to run on embedded and mobile hardware. This work proposes a combined hardware-and-algorithm design that pairs a carefully regularized binarized neural network (BNN) with a compact, variable-precision FPGA processor. A new building block called DenseToRes is introduced to reduce the accuracy loss usually caused by aggressive 1-bit quantization. The supporting processor stores the entire trained network in on-chip memory instead of external DRAM and performs its multiply-accumulate (MAC) operations using an XNOR-based processing element that can flexibly handle 1-, 2-, 4- and 8-bit operand widths from one shared gate array. Implemented on a Xilinx FPGA, the design achieves real-time performance of 64.51 frames per second with 64.92% mean average precision (mAP) on the PASCAL VOC dataset, while consuming only 6.58 W. The results show that jointly optimizing network structure and hardware precision can enable accurate, real-time object detection without relying on off-chip memory.

1. Introduction

Deep learning has pushed computer vision to near-human accuracy in several tasks, powering applications such as autonomous driving and robotics. However, the underlying CNN models are computationally heavy, and running them efficiently on resource-limited embedded devices remains a persistent challenge. To address this, researchers have explored model compression techniques such as pruning and quantization, which reduce the number of parameters and arithmetic operations a network needs. Among these, reducing bit-width — the precision used to represent weights and activations — is especially effective because it lowers both memory traffic and switching activity in hardware.

Binarized Neural Networks (BNNs) take this idea to its extreme by representing weights and activations using a single bit, dramatically cutting energy and bandwidth requirements at some cost to accuracy. While ASIC accelerators built for such low-precision networks offer excellent efficiency, their high development cost and long design cycles make them unsuitable for smaller or rapidly changing projects. FPGAs, in contrast, allow flexible, reconfigurable hardware design at a fraction of the cost, making them well suited for embedded object-detection systems.

This project presents an FPGA-based, real-time object detection processor built for embedded use. A tightly regularized binarized backbone network is designed to minimize the accuracy loss typically associated with binarization, and it is paired with a reconfigurable-precision processing unit that keeps the full model on-chip, avoiding the latency and power cost of external memory access. The resulting system achieves 64.92% mAP at 64.51 frames per second while consuming just 6.58 W, demonstrating that combined hardware–algorithm design can deliver real-time, embedded-grade detection without compromising accuracy.

2. Literature Survey

Several detection frameworks have shaped the direction of this work. Girshick's Fast R-CNN improved on earlier region-proposal methods by sharing convolutional computation across proposals, greatly speeding up both training and inference while improving accuracy on PASCAL VOC. Ren et al.'s Faster R-CNN went further by introducing a Region

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

Proposal Network that shares features with the detection network, enabling near cost-free proposals, though its GPU-based pipeline (about 5 fps) remains too slow for embedded, battery-powered use.

Single-stage detectors offered a faster alternative. Liu et al.'s SSD removed the proposal stage entirely, predicting categories and box offsets directly from multiple feature-map scales, reaching 58 fps on a GPU. Redmon et al.'s YOLO similarly framed detection as a single regression problem solved in one network pass, reaching 45–155 fps depending on the variant. These single-shot, end-to-end approaches directly motivate the lightweight, grid-based detection head used in this project. Later versions such as YOLOv3 and YOLOv4 refined the loss formulation and combined multiple training tricks (e.g., cross-stage connections, Mosaic augmentation) to further boost speed and accuracy, and EfficientDet introduced compound scaling of depth, width and resolution for a favorable accuracy-to-parameter trade-off.

On the hardware side, Sze et al. surveyed efficient DNN processing across platforms and dataflow strategies, informing the choice of an output-stationary dataflow used in this design to minimize on-chip buffer requirements. Pathak et al. surveyed CNN-based detection approaches broadly, providing a useful classification of region-proposal versus regression-based methods that helps position the present single-stage, grid-based design. Building on these ideas, this project focuses specifically on making a single-stage detector efficient enough, in both algorithm and hardware, to run entirely within an FPGA's on-chip resources.

3. Proposed Methodology

The design follows a combined hardware–algorithm optimization strategy with two goals: (1) keep the trained detection model small enough to fit entirely in on-chip memory, and (2) keep the processing hardware flexible enough to execute the mix of 1-, 2- and 8-bit operations that a well-regularized binarized network actually needs.

3.1 Backbone Network Architecture

The backbone network is built from three repeating blocks: an Initial layer, a DenseToRes layer, and a Transition layer. Each block combines binary convolution, batch normalization, and an RPReLU activation function that uses learnable per-channel biases to adjust its operating point. The DenseToRes layer forms dense connections between preceding layer outputs using full 3×3 binary convolutions, preceded by sparse 1×1 point-wise convolutions kept at 2-bit precision to limit accuracy loss. The Transition layer aligns channel counts between dense blocks; because heavily quantizing this layer was found to hurt accuracy significantly, its 1×1 alignment convolution is kept at higher precision, while its depth-wise 3×3 convolution retains a residual connection similar to the DenseToRes block. The Initial layer applies stride and max-pooling early on to reduce input resolution by 16×, lowering the computational load of all subsequent layers.

3.2 Training Strategy

Because binarized networks have limited representational capacity, training them directly on one-hot labels tends to produce poor activation distributions. Instead, the backbone is trained via knowledge distillation from a full-precision ResNet-34 teacher: first, a network with full-precision weights and binary activations is trained; its output distribution then supervises a second, fully binary network. The RAdam optimizer with a cosine learning-rate schedule is used throughout. An ablation study guided the final backbone configuration — combining dense connectivity, selective high-precision transition convolutions, and 2-bit residual activations recovered several percentage points of accuracy compared to a naively binarized baseline, at only a small increase in model size.

3.3 Detection Head and Loss Function

The detection head follows the backbone's structural style: after removing the backbone's classification layer, five additional dense layers (including two transition layers without max-pooling) are appended, with the final 1×1 detection convolution kept at higher precision. The network predicts five bounding boxes per grid cell across twenty PASCAL VOC classes, following a YOLOv2-style detection strategy. Training uses a binary cross-entropy loss over class scores and objectness confidence, following YOLOv3, which improved training stability compared to the mean-squared-error loss used in earlier YOLO versions.

3.4 Layer-wise Quantization

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

Since errors in the Initial, Transition and detection-head layers disproportionately affect final accuracy, these are executed at 8-bit precision with power-of-two scaling factors so that rescaling can be done through simple shift operations rather than multipliers. Batch-normalization parameters are folded into these 8-bit layers to save hardware, while parameters fused with fully binarized layers are left unquantized.

3.5 Processor Architecture and Dataflow

The processor consists of a dual-port SRAM feature-map memory feeding an array of 64 processing elements (PEs), each producing one output per cycle for 64 parallel outputs. Each PE holds local memory for its RPReLU parameters, batch-normalization parameters, and convolution weights. The entire network stays resident in on-chip memory, removing the latency and energy cost of external DRAM access. An output-stationary dataflow was chosen after comparing it with weight-stationary and row-stationary alternatives; for this design's parallelism, it needs roughly nine times less on-chip storage than row-stationary and about 1138 times less than weight-stationary, making it decisive for a resource-constrained FPGA despite requiring a fresh weight value for every input feature. The backbone is configured to add 64 channels after each DenseToRes layer so output channels can be evenly distributed across the 64 PEs.

3.6 XNOR-Based Variable-Precision Processing Element

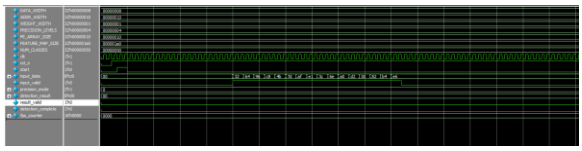
The core innovation is a reconfigurable processing element that serves both the fully binarized majority of the network and its few higher-precision layers using a single arithmetic unit, avoiding the need for a separate high-precision datapath. For 1-bit activations and weights, multiplication reduces to a bitwise XNOR operation, with the result obtained by a simple bit-count instead of an arithmetic multiplier. This identity extends to multi-bit signed operands through an equivalence substitution, allowing higher-precision multiplication to also be carried out with XNOR gates under uniform positive quantization. The resulting MAC unit uses 64 parallel XNOR gates in a shift-add structure that can be time-multiplexed across several configurations: 64 simultaneous 1b×1b MACs, 32 simultaneous 2b×1b MACs, six 4b×4b MACs, or a single 8b×8b MAC — all from the same gate array, without extra multiplexing hardware. Each processing element pairs this variable-precision MAC with a fixed-precision 16b×8b MAC for batch normalization, and since all scaling factors are restricted to powers of two, rescaling throughout the design uses only shift operations.

3.7 Data and Memory Path

The on-chip feature-map memory holds up to 21,632 512-bit words, with a feature fetcher supplying 64 8-bit values per PE according to each layer's precision setting. Data transfer patterns vary by precision — for example, 1-bit operations consume one 64-value word per cycle, while 16-bit operations spread a 512-bit word across 64 cycles. Weight fetchers mirror this behavior for convolution weights. Because the processor computes in an output-stationary order, a result buffer temporarily holds each PE's completed output until its target memory address becomes free, at which point it overwrites input data that is no longer needed.

4. Results and Discussion

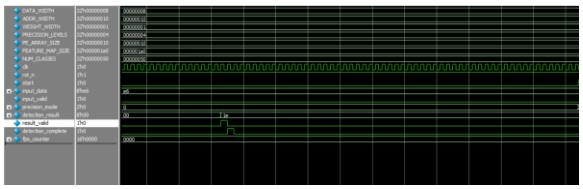
A PyTorch reference implementation of the network was trained on four NVIDIA RTX 2080 Ti GPUs, first pre-training the backbone on ImageNet using distillation, then fine-tuning the full detector (including the detection head) on PASCAL VOC 2007/2012 using quantization operators that mirror the target hardware's arithmetic. After pre-training, the backbone reached 65.54% top-1 accuracy on the ImageNet validation set, and the full detector reached 64.92% mAP on the PASCAL VOC 2007 test set with a model size of only 1.80 MB.



On the target Xilinx FPGA, the complete processor sustains real-time throughput of 64.51 frames per second while consuming just 6.58 W of power, at the same 64.92% mAP achieved in software. These results indicate that the combination of a regularized binarized backbone (via the DenseToRes block and selective layer-wise quantization) with a reconfigurable, XNOR-based processing element allows the system to match the detection accuracy of larger, less specialized networks while operating comfortably within embedded power and memory budgets. Storing the entire model in on-chip memory,

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

rather than off-chip DRAM, is a key contributor to both the low power draw and the consistent frame rate, since it removes the latency and energy overhead of external memory access that typically limits FPGA-based deep-learning accelerators.



5. Conclusion

This project presents an embedded-friendly, real-time object-detection processor built around a co-designed binarized neural network and FPGA processing element. The DenseToRes backbone, trained through full-precision-to-binary distillation and refined through ablation studies, achieves competitive detection accuracy using far fewer parameters and operations than comparable detectors. The accompanying processor removes the need for external DRAM by keeping the full network in on-chip memory and performing most of its arithmetic through an XNOR-based variable-precision MAC unit that flexibly supports 1-, 2-, 4- and 8-bit operand combinations from a single gate array. On the target FPGA, the system reaches 64.92% mAP at 64.51 frames per second while consuming only 6.58 W, comparing favorably with prior FPGA-based object-detection systems in accuracy, hardware cost, and memory bandwidth. Future work could extend the mixed-precision ablation to larger backbone networks and explore adaptive, per-layer precision selection at runtime.

References

[1] R. Girshick, "Fast R-CNN," Proc. IEEE Int. Conf. Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1440–1448.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," Advances in Neural Information Processing Systems (NeurIPS), vol. 28, 2015, pp. 91–99.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," Proc. European Conf. Computer Vision (ECCV), Amsterdam, Netherlands, 2016, pp. 21–37.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779–788.

[5] A. R. Pathak, M. Pandey, and S. Rautaray, "Application of deep learning for object detection," Procedia Computer Science, vol. 132, pp. 1706–1717, 2018.

[6] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," Proceedings of the IEEE, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[7] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.

[9] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Seattle, WA, 2020, pp. 10781–10790.