Real-Time Lightweight Object Detection Pipelines on PYNQ FPGA for Edge Vision Deployment

Enrique J Morales Mateo, Suxia Cui, Lujun Zhai, Prairie View A&M University,
Zhigang Xiao, Satilmis Budak, Alabama A&M University,
Mohammadreza Hadizade, Central State University

Abstract

❖ This work investigates the feasibility of running lightweight object detection on FPGA-based edge platforms. Object detection is a fundamental capability in modern computer vision systems, supporting applications such as autonomous vehicles, robotics, smart surveillance, and edge AI. However, deploying deep learning–based detection models on resource-constrained platforms remains challenging due to high computational demands and energy consumption. Field-programmable gate arrays (FPGAs) provide a promising solution by enabling low-power, real-time acceleration of vision workloads at the edge. This project explores the feasibility of implementing and optimizing object detection pipelines using the PYNQ FPGA platform, with an emphasis on hands-on semiconductor workforce training and practical edge AI deployment. Inspired by recent demonstrations of real-time edge detection on PYNQ, we extend FPGA-based image processing toward more advanced detection tasks, including lightweight convolutional neural network (CNN) inference and hardware-accelerated preprocessing. The study investigates how FPGA programmable logic can support key components of object detection, such as feature extraction, edge-enhanced filtering, and model acceleration. The project also evaluates trade-offs between software-based Python execution on embedded ARM processors and hardware-based acceleration using FPGA overlays. The long-term goal is to develop an accessible framework for integrating computer vision with FPGA-based acceleration while contributing toward efficient, real-time object detection at the edge.

Introduction

❖ Real-time object detection has become essential in applications such as autonomous systems, robotics, and smart surveillance. These systems rely on fast and accurate detection to interact with their environment, often under strict power and hardware constraints. Traditional implementations using GPUs provide high performance but are not always suitable for edge deployment due to their cost and energy consumption.

❖ As a result, there is growing interest in running object detection directly on embedded and low-power platforms. This project explores the feasibility of deploying lightweight object detection models on FPGA-based systems, specifically using the PYNQ FPGA platform. The goal is to understand how hardware-software co-design can support efficient, real-time vision processing at the edge.

Motivation

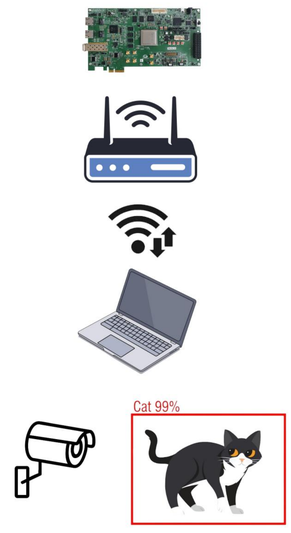
❖ Real-time object detection is critical for applications such as robotics, autonomous systems, and smart surveillance, where decisions must be made quickly and efficiently. However, most deep learning models require high computational power, making them difficult to deploy on low-power edge devices. This creates a need for efficient solutions that can deliver fast detection without relying on GPUs.

❖ Platforms like the PYNQ FPGA offer a promising approach by combining software flexibility with hardware acceleration. In this project, we explore how lightweight models like Tiny YOLO can be deployed on FPGA-based systems to enable real-time object detection at the edge.

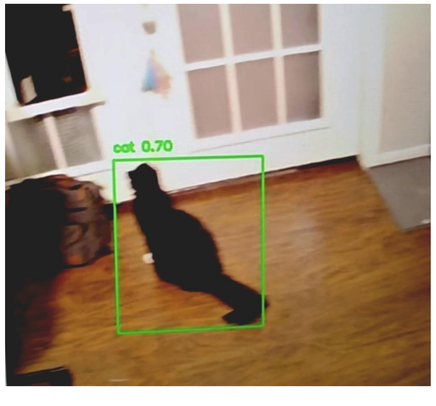
Methodology

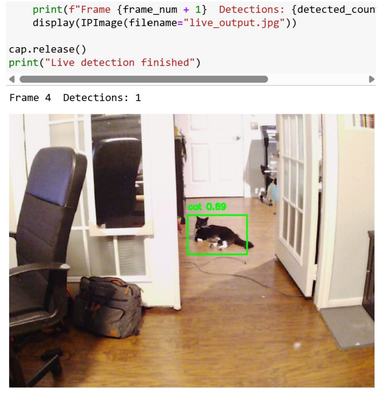
❖ This project implements a lightweight object detection pipeline on the PYNQ FPGA platform by combining software execution on the embedded ARM processor with FPGA-based acceleration. The system is first configured by setting up the PYNQ environment and accessing the Jupyter interface. Input images are then loaded and preprocessed through resizing and formatting before being passed into the detection stage. Object detection is performed using Tiny YOLO, a lightweight convolutional neural network executed on the ARM processor to generate bounding boxes and class predictions. In parallel, the FPGA programmable logic is explored for accelerating preprocessing tasks such as filtering and other parallel operations. The final output is displayed as annotated images with detected objects, providing a functional end-to-end edge vision pipeline.

System Overview of FPGA-Based Object Detection Pipeline



Object Detection Results Using Tiny YOLO





Optimization Strategies

❖ To address the computational demands of continuous live detection, optimization strategies are being explored to improve real-time performance. Running object detection on every video frame introduces significant processing overhead, particularly on resource-constrained hardware.

❖ As an initial approach, the system processes discrete frames by capturing images at fixed time intervals instead of performing continuous video inference. This reduces computational load while maintaining periodic detection capability. Ongoing work focuses on further optimizing the pipeline and exploring FPGA-based acceleration to enable more

Results

❖ The implemented system successfully demonstrates object detection using Tiny YOLO on the PYNQ FPGA platform, producing labeled outputs with bounding boxes and confidence scores across multiple test images. The results confirm that the end-to-end pipeline is functional and capable of detecting objects in different positions and distances within a scene.

❖ Initial testing shows that while detection is reliable, continuous real-time processing introduces noticeable computational overhead, particularly when processing every frame in a live video stream. By transitioning to interval-based image capture, the system maintains consistent detection performance while reducing processing demand.

❖ These results highlight both the feasibility of deploying lightweight object detection on FPGA-based edge systems and the need for further optimization to achieve efficient real-time performance.

Discussion

❖ The results demonstrate that lightweight object detection on FPGA platforms is feasible using a hybrid hardware–software approach. However, reliance on software execution limits performance for computationally intensive tasks. These findings highlight the importance of effective workload partitioning between the ARM processor and FPGA logic. Overall, achieving efficient real-time performance requires both model optimization and deeper integration of hardware acceleration.

Acknowledgement

❖ This research is partially supported by the National Science Foundation under Grant CNS-2411979, EES-2436203; Apple NSI; and the Prairie View A&M University Faculty RISE Program.