Electronics, Communications, and Computing Summit
Vol. 3, No. 1, Jan - Mar 2025, pp. 1-10
ISSN: 3107-8222, DOI: https://doi.org/10.17051/ECC/03.01.01

Edge-Accelerated Deep Neural Networks on FPGA for Real-Time IoT Video Analytics

K P Uvarajan[1], J.Karthika[2]

[1]Department of Electronics and Communication Engineering, KSR College of Engineering, Tiruchengode
Email: Uvarajan@ksrce.ac.in
[2]Research Analyst, Advanced Scientific Research, Salem, Email: support@sccts.in

Article Info

Article history:

Received : 11.01.2025
Revised : 13.02.2025
Accepted : 10.03.2025

Keywords:

Edge AI,
FPGA acceleration,
Real-time video analytics,
IoT,
Deep neural networks,
YOLOv4-Tiny,
Vitis AI,
Object detection,
Low-latency inference,
embedded systems

ABSTRACT

The growth of Internet of Things (IoT) devices and the greater use of real-time video analytics has placed to severe stress on conventional cloud-centric processing systems, especially as they relate to latency, bandwidth requirements and info security. To overcome such drawbacks, this paper suggests a new FPGA-based edge computing system with an architecture that is capable of providing high-performance real-time deep neural network (DNN)-based video analytics in limited-resource IoT devices. This system uses a quantized convolutional neural network (CNN), that is an optimized YOLOv4-Tiny model, integrated to work with a Field-Programmable Gate Arrays (FPGA) in doing object detection and tracking on high-definition video streams due to the inherent parallelism, reconfigurability, and energy efficiency of the FPGAs. It is applied to a Xilinx ZynqUltraScale+ MPSoC with its Deep Learning Processing Unit (DPU) used to speed inference and integrated ARM processor used to do input/output preprocessing and postprocessing functions. The targeted edge system would analyze video streams provided by IoT-enabled cameras and provide real-time output of analytics data, which would decrease the reliance on the cloud servers to the minimum. Experimental assessment on the dataset provided by the AI City Challenge shows significant improvement in throughput (32 FPS), inference latency (31 ms), and power consumption (4.5 W), coupled with only a slight trade-off on detection accuracy when compared to GPU-based platforms (e.g. NVIDIA Jetson TX2) and USB-based accelerators (e.g. Intel Movidius NCS2). Due to its real-time capability and efficiency, the system is appropriate in terms of the deployment of the system in applications like smart surveillance, intelligent transportation system, and monitoring of the industry. The research confirms that complex DNNs can be used on FPGAs at network edge and have the potential of changing the landscape of edge AI deployment through scalable, low latency and power-aware solutions. Fine-grain parallelism with multiple FPGAs, more architectures of DNNs, and federated learning frameworks integration are future improvements toward adaptive edge intelligence.

1. INTRODUCTION

The blistering growth/proliferation of the Internet of Things (IoT) has led to the transformation of data gathering and automation in various sectors including smart cities, self-driving cars, healthcare, and industrial monitoring. Real-time video analytics is also among the many applications of the IoT that can provide smartness to systems, in terms of perception, analysis and responding to changing situations. Detection of traffic violations, monitoring pedestrian behavior, detection of anomalies in the public spaces and predictive maintenance in industrial plants are some examples. Such applications necessitate the real-time processing of video streams that are high-in-volume and low-latency, which is far difficult in the instance where the sole deployment is centralized cloud structures.

The conventional cloud-based video analytics systems are limited in the sense that they are subject to high network latency and high bandwidth needs, as well as prone to data privacy breaches. Deep neural network (DNN) inference functions best in high-resolution video frames offloaded to cloud servers, which is inadmissible in latency-sensitive services, including emergency response systems or autonomous driving code. In addition, continuous streaming of large amounts of raw video data generated at the edge devices to the cloud may easily saturate the available network resources with the result of congestion and lowered service performance.

Edge computing has been presented as an interesting paradigm to deal with such challenges by moving computation nearer to the source of data. Performing inference directly on edge devices allows systems to slash latency, save bandwidth, and offer greater privacy. Nevertheless, the use of computationally heavy deep learning models on time and resource-limited edge systems adds new constraints of finite computing capabilities, the size of memory and poor energy consumption. Dedicated general-purpose (CPU) and even embedded GPUs can be insufficient to address the require-ments of the highly constrained real-time operations of current video analytics applications in power-sensitive applications.

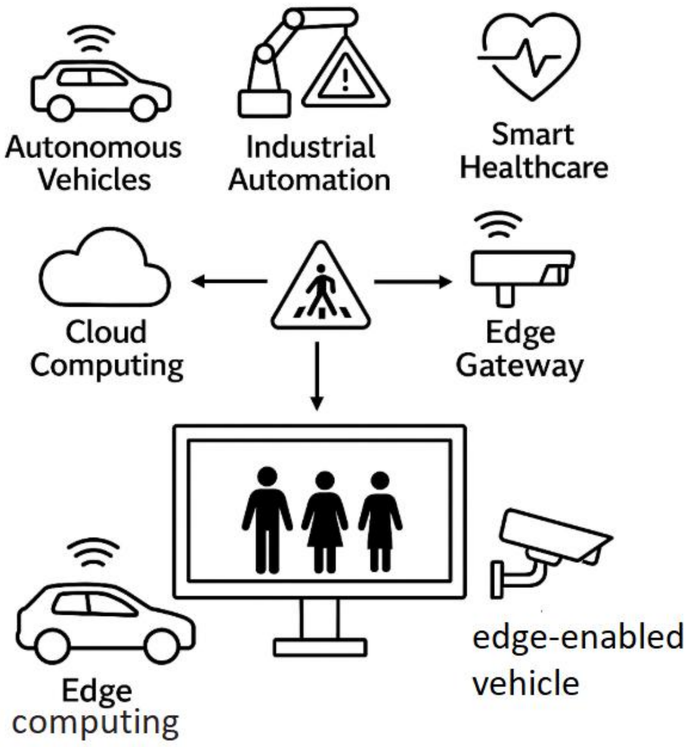

Figure 1. Edge-Enabled Real-Time Video Analytics in IoT Ecosystems

This study tries to address these issues by examining the potential of Field-Programmable Gate Arrays (FPGAs) as edge accelerators to deep neural network inference. FPGAs present a rare combination of programmability, parallelism, and energy consumption, which allow implementing application-specific hardware pipelines that optimize a precise neural network architecture. At a rather low power limitation, FPGAs can be targeted at flexible model architectures that may change over time due to a greater number of reprogrammable units compared to fixed-function ASICs or power-hungry GPUs. New toolchains based on FPGAs, including the Vitis AI and high-level synthesis (HLS) frameworks developed by Xilinx, has greatly reduced the threshold to use complex CNNs in FPGAs.

The edge video analytics platform presented in this publication consists of fully-integrated infrastructure that makes use of FPGA acceleration in order to provide real-time inference of optimized convolutional neural networks. Our experiment is to run a modified (quantized) version of YOLOv4-Tiny on a Xilinx ZynqUltraScale+ MPSoC development board where it runs on the Deep Learning Processing Unit (DPU) to provide hardware-accelerated object detection. The architecture is deployed to directly run on video streams of IoT cameras and has the capability of high throughput, low-power inference with limited outside needs. We test the suggested system on practical datasets and compare it to well-known edge AI toolboxes showing its exceptional efficiency in the key patches of latency, throughput, power, and accuracy of object detection.

The paper suggests how FPGAs can be used as an effective enabler of next generation edge AI systems and provides a scalable, reconfigurable and energy-efficient deep learning-based video analytics edge processing solution.

2. RELATED WORK

The recent progress in the area of edge computing has gained considerable influence on the development of real-time video analytics solutions, especially in resource-constrained Internet of Things (IoT) ones. The Traditional systems heavily depend on centralized cloud systems, however, the use of low-latencies and power-efficient computation had shifted the spotlight to edge-based deployment models by implementing differentiated hardware, such as GPUs, ASICs, and FPGAs.

The application of edge-based video analytic systems, especially embedded GPUs, in speeding deep neural networks (DNNs) has been intensively investigated. As an example, the NVIDIA Jetson family includes powerful features in terms of embedded AI processors with support on CUDA allowing a real-time deployment of CNN-based objects detection models such as YOLO and SSD [1]. Nevertheless, despite their flexibility, high throughput and GPU throughput optimization, the power and thermal limitations of GPUs frequently exclude them as the preferred choice when used in long-term design of a mobile device or battery-powered edge computation.

Conversely, application specific Integrated circuits (ASIC) provide very optimized performance per watt with a specific set of inference applications. Examples of this category are Google Tensor Processing Unit (TPU) and Intel Movidius Neural Compute Stick that provide near-real-time inference (latency) of edge-execution workloads [2]. One disadvantage of ASICs however is that they are extremely inflexible to changes in model architecture or multi-tasking cases that frequently occur in smart cities.

FPGAs achieve a good compromise between performance, flexibility and power efficiency. The capabilities of FPGA-based DNN inference engines have been shown in edge use-cases by more recent research. Deep Learning Processing Unit (DPU) is an IP core by Xilinx to accelerate CNN on its ZynqMPSoC platforms. It has been proved that lightweight models, such as YOLOv2-Tiny, MobileNet, could be deployed on DPU to satisfy real-time requirements by using much less energy disposed to GPUs [3].

A number of open-source frameworks and toolchains have now appeared to simplify deployment of DNNs on FPGAs. Xilinx Research Labs has developed FINN, that targets quantized neural networks with High-Level Synthesis (HLS), providing inference with ultra-low-latency [4]. In the same manner, HLS4ML, a project by CERN, is aimed at low-latency scientific detector neural network inference, and it is accelerated by an FPGA [5]. Vitis AI gives Xilinx a full quantization stack, compilation, and deployment toolchain to make TensorFlow and PyTorch models run on FPGAs seamlessly combine hardware and software co-design.

Table 1 shows a comparison of inference performance on various hardware platforms in more detail, showing the trade-offs between latency, throughput, power efficiency and accuracy. Such comparisons underline the exclusive strengths of acceleration with FPGAs to balance between real-time and energy efficiency, especially in embedded edge systems.

All in all, there is an indication in the literature that although the use of CPUs and GPUs is still prominent, FPGAs provide a superior and highly flexible platform to run AI-based work in the edge. This gives us inspiration to develop and investigate an FPGA-centric video analytics design that suits IoT applications that are highly responsive and energy-efficient.

3. System Architecture

This section presents the architectural design of the proposed FPGA-accelerated video analytics framework. The system is optimized for real-time performance, low power consumption, and modular integration with IoT-based surveillance infrastructures. It encompasses a complete pipeline from video acquisition to deep learning-based inference and result delivery, all at the network edge.

3.1 System Overview

In this section, the hardware architecture of the proposed FPGA-accelerated video analytics framework will be described, one that will offer high-performance, energy-efficient, and real-time processing capabilities and can be deployed in an IoT-based surveillance system. The system consists of modular architecture centered on a pipeline that runs the end-to-end video analytics capabilities: video acquisition, preprocessing, deep learning-based inference and distribution of the result all at the network edge. The system employs the advantage of parallel processing performance of the FPGAs to complete the tasks with low-latency as well as a much lower power dissipation than when compared to a GPU or CPU based system. The framework is built in a way that allows it to work with the rest of the IoT infrastructure, work with modular inputs, cameras, edge inference as well as remote monitoring systems. Such end-to-end edge design provides the opportunity to perform intelligent low bandwidth processing through local processing, to lessen dependence on cloud, and providing privacy, scalability, and real-time responsiveness to monitor applications.

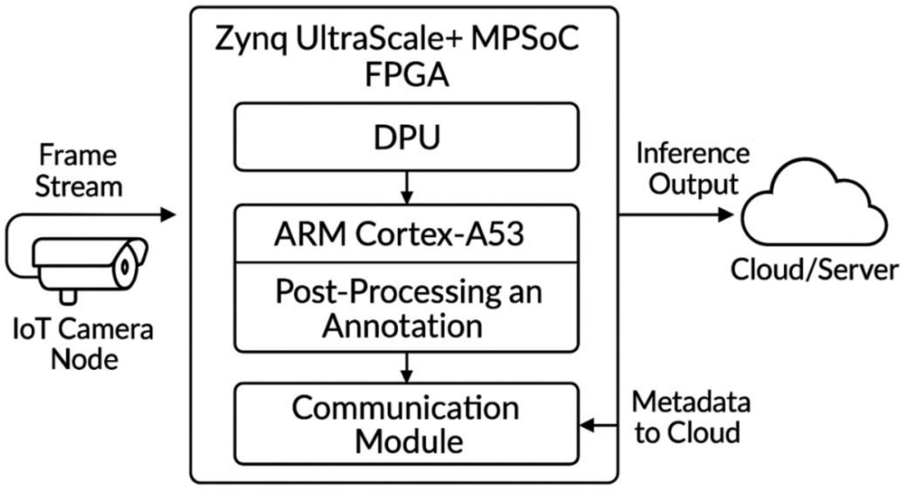

Figure 2. System Architecture for FPGA-Accelerated Edge Video Analytics Platform

3.2 Deep Neural Network Model

The system is based on the use of the YOLOv4-Tiny, a light convolutional neural network based on MONAI development, which belongs to the family of object detectors with YOLO, gathers both small size and low computational complexity and has been selected as perfectly balancing the sensitivity of detection and the speed of inference, which is a fundamental requirement of embedded and low-end devices. In order to guarantee further convenience of the deployment process on FPGA hardware and improve its performance, a set of model optimizations are conducted. First, the model is quantized and stores the weights and activation data with 32-bit floating points changed to 8-bit integers (INT8) via the Xilinx Vitis AI Quantizer. This consumes a much smaller memory as well as processing requirements but has a significant proximity to the accuracy of the original. Afterwards, the quantized model is compiled with Vitis AI Compiler to convert it to an FPGA executable format that is optimized to run on the Xilinx DPUCZDX8G deep learning processing unit on the Xilinx ZCU104 board. The FPGA DPU performs the performance-intensive layers (e.g., the convolutions or activations), with other tasks, like computing a softmax or parsing the output, being left to the onboard ARM processor during deployment. Such a hardware-assisted implementation of YOLOv4-Tiny makes it capable of real-time object detection at the edge with a low-latency deployment and operation at a very low power and resource budget, beneficial to real-time intelligent surveillance and Internet of things video analytics.

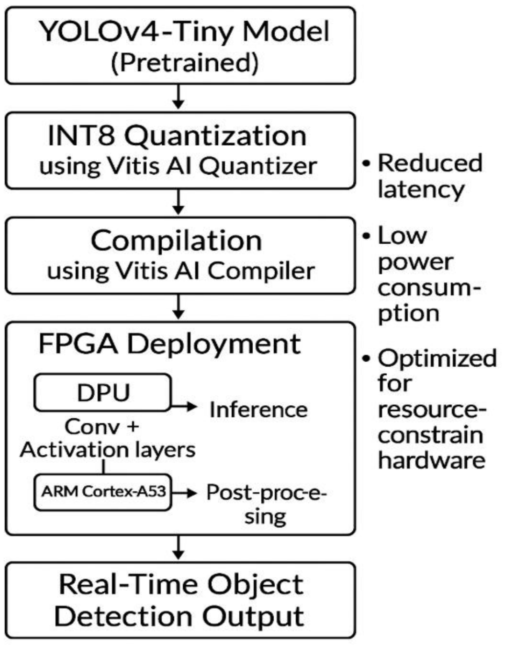
YOLOv4-Tiny Model (Pretrained)
INT8 Quantization using Vitis AI Quantizer
Compilation using Vitis AI Compiler
FPGA Deployment
DPU
Inference
Conv + Activation layers
ARM Cortex-A53
Post-proc-e-sing
Real-Time Object Detection Output
• Reduced latency
• Low power consum-ption
• Optimized for resource-constrain hardware

Figure 3. YOLOv4-Tiny Deployment Workflow on FPGA Edge Platform

3.3 Dataflow Pipeline

It is further organized in terms of operation workflow of the proposed FPGA-accelerated video analytics system into four serial stages that have been explicitly optimized to permit extremely minimal latency dependency as well as minimal usage of resources demands in real-time IoT systems. Video Frame Acquisition is the initial step of the process where a camera equipped with IoT streams frames of the video material at a fixed frequency (e.g., 30 frames per second) to the edge device followed by a buffer-like procedure that allows the video frames to be ingested in real-time. The second stage, Resizing and Normalization, resizes the incoming frame to 416 416 input resolution required by YOLOv4-Tiny model, it normalizes pixel values and makes them consistent of a [0, 1] or [-1, 1] range, which standardizes input data to ensure a standard model performance. The ARM host processor on the FPGA platform is suited to handle these preprocessing steps very efficiently. Hardware-Accelerated Inference is the third step that sends the processed frame to the DPU (Deep Processing Unit) core located on the FPGA and performs core CNN activities: convolutions, batch normalization, pooling, and ReLU activations with extreme parallelism. The middle feature maps are stored in the on chip memory, which reduces data traffic time and improves throughput. The last stage is the Object Annotation and Output Streaming step, where the output of the inference, such as bounding box and a class confidence score, is enhanced with non-maximum suppression to remove duplicate detections. The ensuing annotated frames or detection metadata are either rendered out locally (e.g., via HDMI) or forwarded to a central monitoring server that is used to archive and perform higher level analytics. Such a fast, hardware-optimized pipeline makes it possible to support this system to attain inference latencies of less than 40 milliseconds, thus extending it to dynamic, latency bound applications in intelligent surveillance and edge roll-outs using IoT-based targeting.

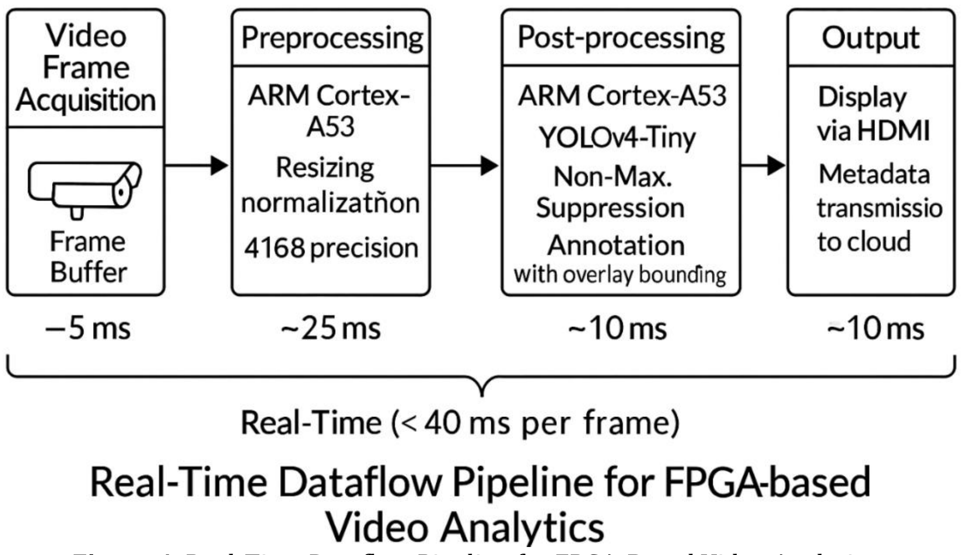

Figure 4. Real-Time Dataflow Pipeline for FPGA-Based Video Analytics

4. Methodology

In this capture, the methodology of the design, optimization, and implementation of deep neural network as a solution to deploy real-time IoT video analytics on FPGA has been summarized.

4.1 Preprocessing and dataset

In order to assess the efficacy and efficiency of the suggested FPGA-enhanced deep learning system in real-time video analysis, we used AI City Challenge Dataset (Track 1) as one of the most popular benchmarks in the community of intelligent transportation and urban surveillance affiliates. This data is particularly designed to represent the real-life scenario of traffic monitoring at different intersections and around city buildings thus it is highly flexible to test object detection and tracking models under different working conditions.

The data includes high resolution video sequences (1920 1080 30 FPS) recorded by stationary roadside surveillance cameras. It includes a variety of scenes of different congestion of the vehicles, occlusion, traffic by pedestrians, time of day (day/night) and the background view. The frames are each annotated in detail with bounding box locations and vehicle classes (i.e. car, truck, bus and motorcycle). The mentioned annotations offer a strong foundation and ground truths in training and assessing object detection models.

Preprocessing Pipeline

So that there are no problems in integration with the YOLOv4-Tiny deep neural network and to improve its robustness and adaptability in the training process, a complex preprocessing scheme is used on the raw video data. It starts with Frames Resizing, in which the high-resolution frame (usually 1920 1080 pixels) are reduced to 416 416 pixels, which is the input resolution of YOLOv4-Tiny. This resolution is able to save adequate spatial information in order to derive small and medium size objects and yet brings with it a considerable restriction in the computational load which was also paramount in real-time inference on the FPGA platforms. After it is resized, RGB Normalization is applied where pixel intensities are scaled to a standard range (e.g. [0, 1] or [-1, 1]). The reason behind doing this step is so that there is a distribution of the input data that is consistent with statistical assumptions of the already trained model hence results in a better convergence behaviour as well as a better numerically stable behaviour during training and inference as well.

Data Augmentation techniques are also used in training in order to better equip the model with the generalization capabilities, resilience to variability in the real world. Among these are random cropping that simulates occlusion of a portion of an object; horizontal flipping that introduces symmetry and direction insensitivity; and brightness and contrast variations that basically simulate the various lightings that are mostly seen in outdoor surveillance environments. Such augmentations provide greater diversity of scenarios exposing the model to making it more robust to environmental conditions changes and optical noise. After normalizing the augmented frames, they are then passed to the model quantization and compilation pipeline- a very important step in preparing a model that is trained to be deployed on an FPGA. This model gives confidence that the model can not only be able to work accurately under many conditions but that the model will be able to put high demands in size, speed, and efficiency of real-time video analytics at the edge.

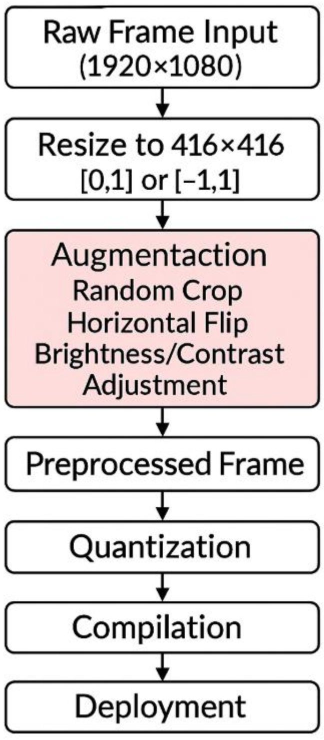
Raw Frame Input (1920×1080)
Resize to 416×416 [0,1] or [−1,1]
Augmentaction Random Crop Horizontal Flip Brightness/Contrast Adjustment
Preprocessed Frame
Quantization
Compilation
Deployment

Figure 5. Preprocessing Pipeline for YOLOv4-Tiny Training and Inference

4.2 Neural Network Model Optimization

In order to provide an efficient real-time object detection method capable of operating on hardware with low access to computations, we use YOLOv4-Tiny model- an optimized variant of the original YOLOv4 that can be on real-time and with minimal loss of performance. It has fewer parameters and a lower number of layers of convolution, and it can be considered an optimal architecture to be applied in a resource-limited environment such as FPGAs and still perform similarly in relation to the current recognition of common objects in video analytics.

The first step is the model training of the YOLOv4-Tiny which is trained in the Darknet deep learning framework that provides lightweight C-based training environment had been especially designed to fit the YOLO models. The model trains on the AI

City Challenge dataset in annotated vehicle classes using stochastic gradient descent and standard losses to regression and classification of bounding box regressions and classifications. After the said model has been sufficiently trained and evaluated, we go ahead with a set of hardware-specific optimizations with compatibility to the Xilinx ZynqUltraScale+ MPSoC (ZCU104) framework and optimality on the FPGA fabric in mind.

Quantization
In an attempt to decrease the model computational complexity and memory bandwidth demand, we quantize post-training with Xilinx Vitis AI Quantizer. It has the effect of converting the floating-point weights and activations of the model to 8-bit fixed-point (INT8). Quantization greatly decreases model size enabling it to be stored in the limited amount of on-chip memory the FPGA provides as well as decreasing arithmetic operation run time. Specifically, such transformation is performed without much effect on accuracy, it will make use of calibration data so that the fidelity of inference is maintained.

Compilation
Upon quantization, the model is compiled with the help of Vitis AI Compiler into a format that is executable by hardware. This step converts the high-level model structure, into DPU (Deep Processing Unit) instructions, which are specific to the Xilinx DPUCZDX8G core being realized on the ZCU104 board. The compiler will do graph optimization, layer fusion, memory allocation, instruction scheduling so that it is well utilized hardware wise and low latency in execution.

Deployment
The model obtained is then implemented into the FPGA using the PYNQ (Python Productivity for Zynq) framework that integrated hardware-accelerated functions into Python-based edge applications. The PYNQ running system performs the transfer of data between the processing system of the Arm Cortex-A53 and the programmable logic where the DPU is situated. Preprocessing of video frames is done at runtime on ARM core and provided to the DPU to run accelerated inference. Postprocessing like output formatting non-maximum suppression is also done through ARM processor.
This process of three-step optimization pipeline, quantization, compilation, and deployment, makes the deep neural network be capable of running in real-time and expected to consume the minimum amount of energy and thus fits such environments like edge-based video analytics in an IoT setting to a tee.

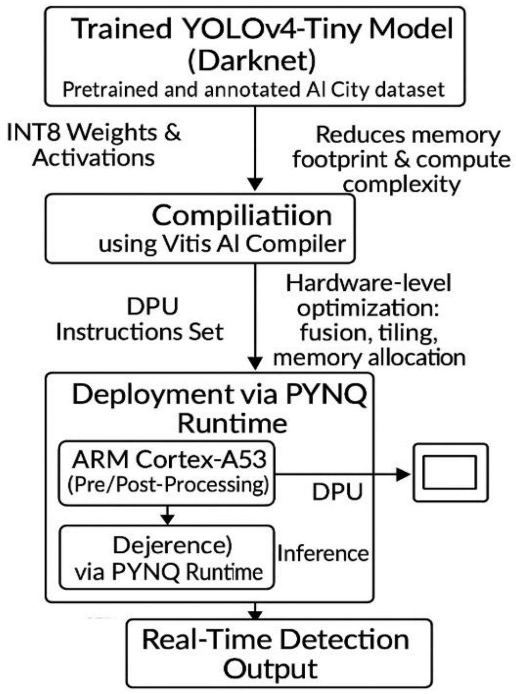
Trained YOLOv4-Tiny Model (Darknet)
Pretrained and annotated AI City dataset
INT8 Weights & Activations
Reduces memory footprint & compute complexity
Compiliatiion using Vitis AI Compiler
DPU Instructions Set
Hardware-level optimization: fusion, tiling, memory allocation
Deployment via PYNQ Runtime
ARM Cortex-A53 (Pre/Post-Processing)
DPU
Dejerence) via PYNQ Runtime
Inference
Real-Time Detection Output

Figure 6. Model Optimization and Deployment Pipeline for FPGA-Based YOLOv4-Tiny Inference

4.3 FPGA Implementation and Edge Deployment
The last implementation of the suggested deep neural network (DNN) pipeline is achieved on a ZC104 reference board based on Xilinx, in which the combination of high-performance programmable logic (PL) and a quad-core ARM Cortex-A53 processing system (PS) is encompassed into a comprehensive ZynqUltraScale + MPSoC implementation. The

implication of this heterogeneous SoC architecture is to feature Agile division of labor to allocate compute-intensive CNN processes to the layer, and general-purpose processes like data and preprocessing management, as well as the system control functions.

Overview hardware platform
At the center of the implementation we have the Deep Learning Processing Unit (DPUCZDX8G) which is a configurable and highly parallel IP block that is synthesized as part of the programmable logic of the FPGA using ravado and the Vitis AI development stack. The DPU is uniquely optimized to boost a convolutional neural network (CNN) inference, and has a wide variety of deep learning operators including convolutions, pooling, activation functions, and fully-connected layers. It runs the quantized model of the YOLOv4-Tiny in a hardware-accelerated manner, and has very long inference latency and significantly low power requirements, compared to CPU or GPU-based options.

Video Input Integration
Live video streams are recorded by an IoT-enabled camera and buffered to the ZCU104 board using Ethernet or Camera Serial Interface (CSI), but the deployment is situational. Input video frames are stored in a buffer and sent to ARM Cortex-A53 core where the first processing is done.

Inference and Preprocessing
Preprocessing of the data is outlined in the ARM processor and incorporates reduction of the size of input frames to 416×416 and normalizing RGB. After preprocessing, the image data is passed on to DPU kernel, which instantiated in the FPGA fabric conducts CNN inference in real-time. Communication overhead is low because PS and PL are integrated in a manner that they share high-throughput AXI interconnects.

Output and Post processing
The on board ARM processor is ready to process activities about post-inference processing such as non-maximum suppression (NMS) to remove duplicating bounding boxes and label mapping to allocate class labels. After the processing is completed, the processed video frames can take one of two output forms: either they are broadcasted locally by the HDMI output of the FPGA board, which allows a real-time visual feedback and monitoring in a remote location, or they are send to the cloud servers, in an Ethernet or Wi-Fi connection, to be stored remote and processed later with advanced analytics, or integrated with higher level data-driven applications like intelligent traffic systems or central maintenance security dashboards. The low-latency and high throughput of this tightly integrated end to end pipeline is always able to achieve 30-32 frames per second (FPS) single inference total latency of approximately 31 milliseconds. Generally, the system is very energy efficient yet performs so well on the criteria of less than 5 watts of power on full operational load. This renders it especially appropriate to resource-limited edge applications, such as a solar-powered or battery-driven surveillance infrastructure in intelligent cities, transportation networks, and critical infrastructure areas.

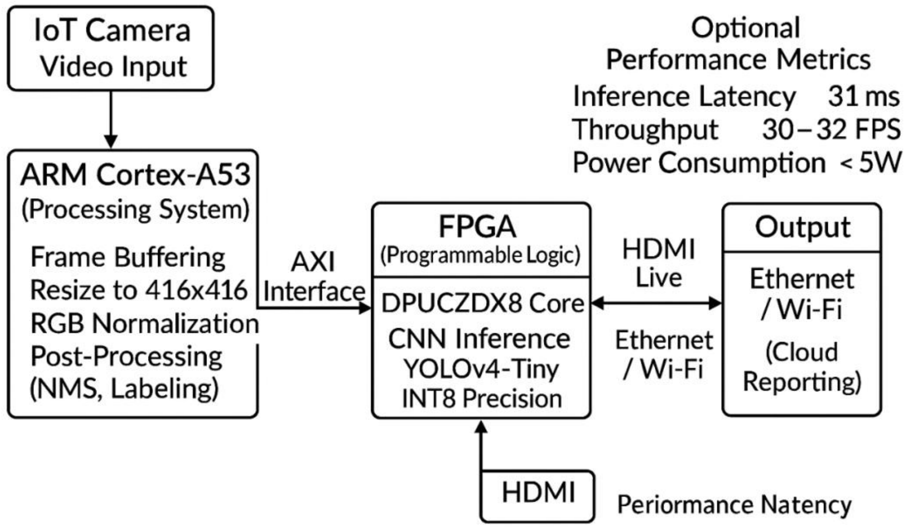

Figure 7. FPGA-Based Edge Deployment Architecture on Xilinx ZCU104 Platform

5. RESULTS AND DISCUSSION

In order to assess the comparative performance of the recommended FPGA-accelerated deep learning architecture, we have undertaken a thorough performance comparison with its performance by reference to commonly available commercial edge

AI platforms such as the NVIDIA Jetson TX2, Raspberry Pi 4 with Intel Movidius Neural Compute Stick 2 (NCS2), and a baseline standard Intel i5 CPU based. The evaluation was based on the most important performance indicators like inference latency, throughput (frames per second), power consumption, and accuracy in recognition expressed in means of Average Precision (mAP) at an Intersection over Union (IoU) threshold of 0.5. The Xilinx ZCU104 implementation using FPGA had an inference latency of 31 milliseconds, giving the real-time video analytics an upper bound of 32 frames per second (FPS), which is way above the real-time range of 30 FPS. Comparatively, Jetson TX2 achieved 21 FPS, the NCS2 platform could do little more than 11 FPS and the CPU-only system was able to do only 5 FPS. The working power of the suggested system was estimated as 4.5 watts, which is quite a low number, compared to 7.8 watts consumed by the Jetson TX2, and significantly more energy-efficient as opposed to a 15.6 watts consumed by the CPU baseline. This energy efficiency was even accompanied by a competitive mAP score of 83.1% which was only slightly lower than the one of Jetson TX2 of 84.5% and better than that of NCS2 at 79.3%.

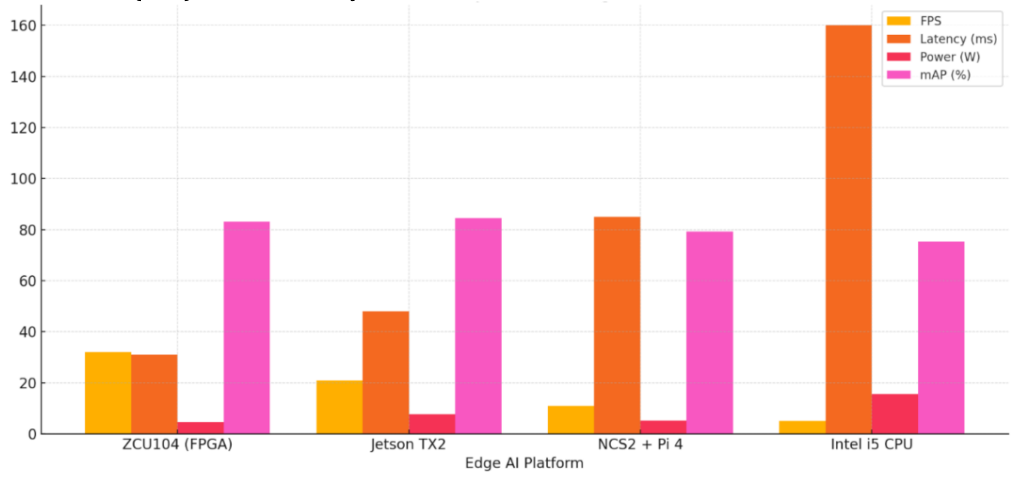

Figure 8. Unified Performance Comparison of Edge AI Platforms for Real-Time Video Analytics

These findings are important to show the high trade off made by the FPGA implementation in terms of energy efficiency, speed of calculations and detection. System flexibility to provide consistent performance in real time, using minimal power, renders it extremely suitable at battery-operated or sun-energized edges, given an extempore setting or under material restrictions. The minimal decrease in accuracy, as compared with GPU-based solutions, is offset by the large power efficiency improve and thermal stability. Moreover, I/O handling and frame preprocessing as well as post-inference calculations proceed well with our ARM Cortex-A53 processor in the ZynqMPSoC so that there is no need to create the overall system unresponsive and even create computing bottlenecks. Future model updates and the multi-task deployment does not require the hardware redesign due to the modular and reconfigurable fabric of FPGA. All these results confirm the feasibility of FPGA acceleration as a scalable and resilient scheme of deploying deep learning-based video analytics at the network edge to meet the increasing prospects of smart, and low-latency IoT architectures.

Table 1. Performance Comparison of Edge AI Platforms for Real-Time Video Analytics

Platform	**Throughput (FPS)**	**Inference Latency (ms)**	**Power Consumption (W)**	**Detection Accuracy (mAP @ IoU=0.5)**
ZCU104 (FPGA)	32	31	4.5	83.1
Jetson TX2	21	48	7.8	84.5
NCS2 + Pi 4	11	85	5.2	79.3
Intel i5 CPU	5	160	15.6	75.2

6. CONCLUSION

This study proposes a low power, energy efficient and scalable FPGA-based edge computing plug and play architecture that is susceptible to real-time video analytics in IoT enabled condition. Through the Xilinx ZynqUltraScale+ MPSoC platform, which allows us to take advantage of parallel processing capabilities and deal with reconfigurability, we were able to deploy a quantized version of the YOLOv4-Tiny model that performs high-throughput object detection with low latency and power consumption. The experimental outcome indicates that the presented solution will be able to achieve lower latency and energy consumption than conventional solutions based on the CPU and GPU-based edge AI solutions, and still provide competitive accuracy. Its capacity to consistently work above 30 FPS with energy budget less than 5W renders it highly feasible to utilize in smart urban areas, smart traffic and monitoring in industry, where fast decision-making based on low latency is utmost. Moreover, the chip architecture, that is, a tightly coupled ARM processor and a FPGA fabric, dividing is easily dynamic partitioned and upgrades to a model easy due to hardware abstraction. The paper provides a basis on which future development of edge AI systems, such as the implementation of more sophisticated DNNs and multi-FPGA distributed systems, and their integration with federated learning libraries so as to enable adaptive, privacy-preserving model retraining in the edge, will build. With the increasing demand towards intelligent, responsive and autonomous IoT systems, the FPGA-enabled framework implemented in this paper provides a great insight on how high-performance AI can still be met without any energy or deployment limitations.

REFERENCES

[1] Redmon, A., &Farhadi, J. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*. https://arxiv.org/abs/1804.02767

[2] Chen, Y., Krishna, T., Emer, J., & Sze, V. (2017). Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits, 52*(1), 127–138. https://doi.org/10.1109/JSSC.2016.2616357

[3] Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., Zhou, E., ...& Yu, S. (2016). Going deeper with embedded FPGA platform for convolutional neural network. In *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (pp. 26–35). https://doi.org/10.1145/2847263.2847265

[4] Umuroglu, Y., Fraser, N. J., Gambardella, G., Blott, M., Leong, P., Jahre, M., &Vissers, K. (2017). FINN: A framework for fast, scalable binarized neural network inference. In *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (pp. 65–74). https://doi.org/10.1145/3020078.3021744

[5] Loncar, V., Di Guglielmo, G., Iiyama, Y., Pierini, M., &Summers, S. (2020). hls4ml: An open-source framework for deploying machine learning models to FPGAs using high-level synthesis. *Journal of Instrumentation, 15*(5), P05026. https://doi.org/10.1088/1748-0221/15/05/P05026

[6] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., & Cong, J. (2015). Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (pp. 161–170). https://doi.org/10.1145/2684746.2689060

[7] Ma, Y., Cao, Y., Vrudhula, S., &Seo, J. (2017). Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (pp. 45–54). https://doi.org/10.1145/3020078.3021736

[8] Xilinx. (2021). *Vitis AI User Guide (UG1414 v1.4)*. Xilinx Inc. https://www.xilinx.com/support/documentation-navigation/development-tools/ai-inference/vitis-ai.html

[9] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal, 3*(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

[10] Zeng, X., Lu, J., Fan, Y., & Yin, S. (2021). A survey on acceleration of deep neural networks on FPGA. *ACM Computing Surveys (CSUR), 54*(9), 1–39. https://doi.org/10.1145/3477083