Project Report

All-in-one Self-adaptive Computing Platform for Smart City

30 June, 2023

Team Number: xohw23-114

Group Members: Hassaan ur Rahman, Syed Muhammad Shariq Ali, Morgan Greenhill Foteini Papadogianni, Akshay Gopinath

Supervisor: Dr. Xiaojun Zhai

Contents

Abstact

The "All-in-one Self-adaptive Computing Platform for Smart City" project presents a comprehensive solution for efficient and intelligent human detection in smart city environments. It utilizes the Xilinx Kria KV260 Vision AI Starter Kit, which combines the Zynq UltraScale MPSoC architecture with 4 GB of DDR4 memory for AI-accelerated applications such as smart cameras and face detection. The software component employs PetaLinux and Vitis Video Analytics SDK (VVAS) to create a streamlined development environment for seamless integration of hardware and software. By incorporating image classification models like YOLO and ResNet, the project enables real-time person detection from video input, contributing to enhanced security and safety in smart cities. Overall, the "All-in-one Self-adaptive Computing Platform for Smart City" project offers a comprehensive solution that combines powerful hardware, advanced software frameworks, and cutting-edge AI models for efficient and accurate human detection in smart city environments.

1. Introduction

In recent years, the concept of smart cities has gained significant attention as a means to address urban challenges and enhance the quality of life for residents. Smart city technologies leverage advanced computing systems and artificial intelligence to monitor and manage various aspects of urban environments, including transportation, energy consumption, public safety, and more. Among the key requirements for effective smart city solutions is the ability to accurately detect and analyze human activities and behaviors from video footage. The All-in-one Self-adaptive Computing Platform for Smart City project aims to develop a comprehensive solution for smart cities using a Raspberry Pi camera for human detection from video footage. The project incorporates hardware, software, and artificial intelligence components to create an efficient and adaptable system.

2. Methodology

2.1 Hardware

The project utilizes the Xilinx Kria KV260 Vision AI Starter Kit, which combines the power of the Zynq UltraScale MPSoC architecture with 4 GB of DDR4 memory. This hardware configuration is ideal for AI-accelerated applications and enables the development of smart cameras, AI face detection, and machine vision-based defect detection.

Figure 1 Kria KV260 Board

The kit has a MIPI CSI-2 interface which allows for easy connectivity and direct integration of the Raspberry Pi camera module and runs the Petalinux operating system. This OS is a version of Linux which has been customized for Xilinx FPGA devices.

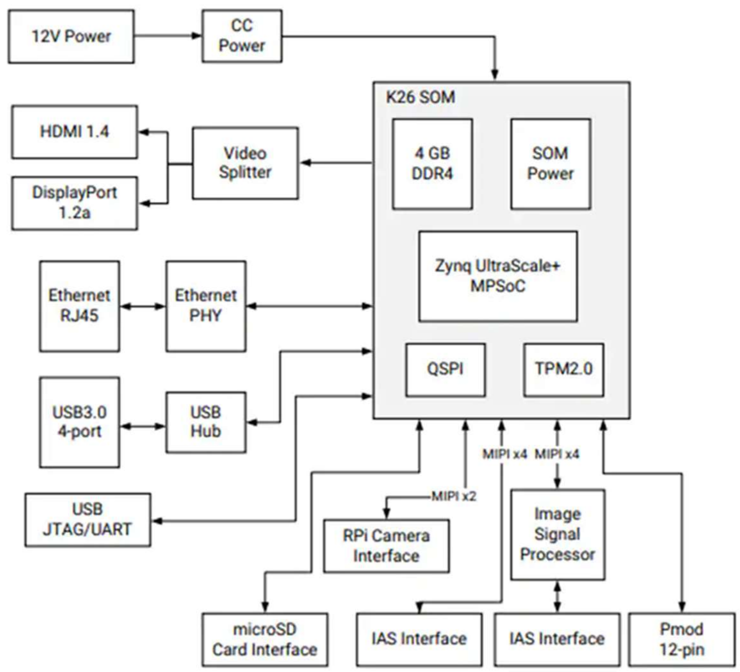


Figure 2 KV260 Block Diagram

The hardware platform provides the necessary computing resources and AI acceleration capabilities to support a range of applications in smart city environments. It serves as the foundation for implementing various AI-accelerated tasks, including smart cameras, AI face detection, and machine vision-based defect detection. The project uses a Raspberry Pi camera module to capture video footage. The camera module is connected to the Xilinx Kria KV260 board, allowing it to acquire real-time video input. By leveraging the power of the Xilinx Kria KV260 Vision AI Starter Kit, the project enables efficient processing and analysis of video footage captured by the Raspberry Pi camera. The kit's integration with the Zynq UltraScale MPSoC architecture ensures high-performance computing capabilities and enables real-time data processing.

2.2 Software

The software component of the project includes PetaLinux and Vitis Video Analytics SDK (VVAS) [1]. PetaLinux provides a Linux distribution optimized for Xilinx FPGA platforms, enabling the customization and configuration of the operating system. VVAS offers a comprehensive framework for developing video analytics applications, providing tools and libraries to streamline the development process. The Vitis Video Analytics SDK (VVAS) can be used alongside Petalinux to create powerful AI applications.

PetaLinux, a Linux distribution optimized for Xilinx devices, is utilized in the project. It provides a robust and customizable software environment that allows for easy integration of the hardware and software components. PetaLinux ensures that the software system is compatible with the Xilinx Kria KV260 Vision AI Starter Kit, enabling smooth communication and utilization of the hardware resources.

The project also incorporates the Vitis Video Analytics SDK (VVAS), which serves as a powerful video analytics framework. VVAS provides a set of tools, libraries, and APIs that facilitate the implementation of advanced video processing and analytics algorithms. It allows for efficient and real-time analysis of video footage captured by the Raspberry Pi camera. By utilizing PetaLinux and VVAS, the project creates a streamlined development environment for the smart city application. These software components enable seamless integration between the hardware and software, ensuring optimal utilization of the hardware resources and efficient processing of video data.

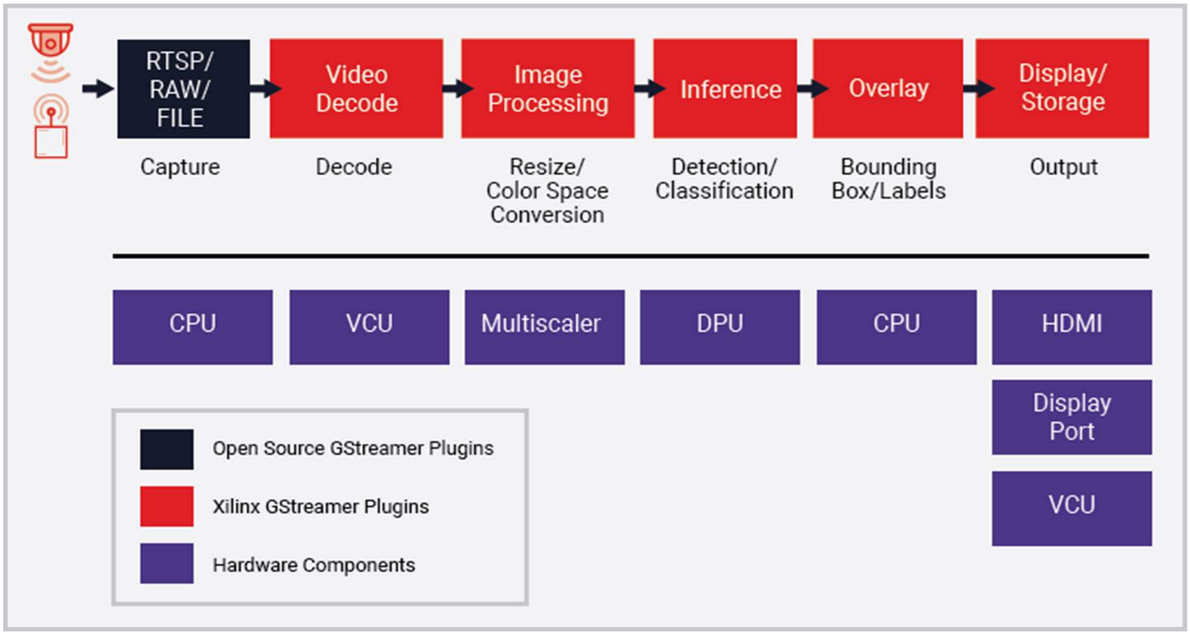


Figure 3 Xilinx VVAS

2.3 Artificial Intelligence

Image Classification Models: The project utilizes two popular image classification models, YOLO (You Only Look Once) and ResNet. These models have been widely adopted for object detection tasks, including human detection.

YOLO is a real-time object detection model that operates by dividing the input image into a grid and predicting bounding boxes and class probabilities for each grid cell. It offers a good balance between accuracy and processing speed. ResNet: ResNet is a deep convolutional neural network architecture known for its superior performance in image classification tasks. It is used as a complementary model to improve the accuracy of human detection.

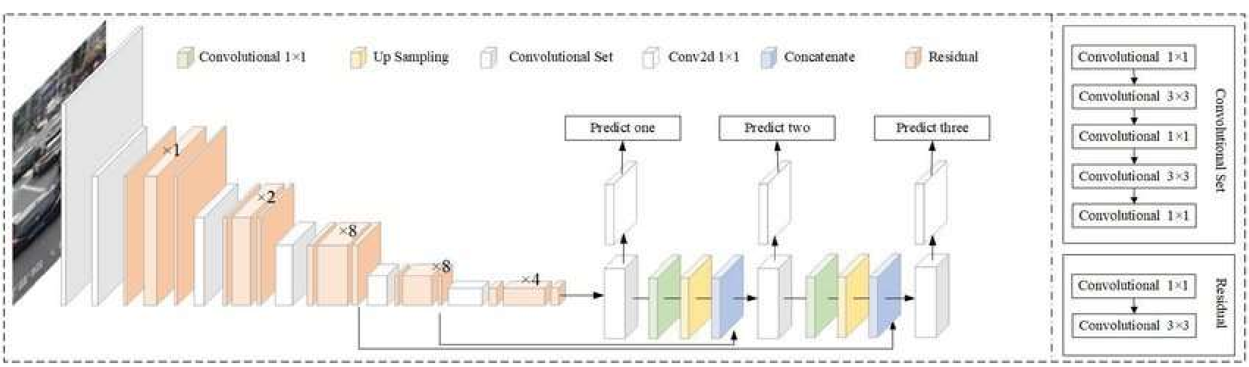
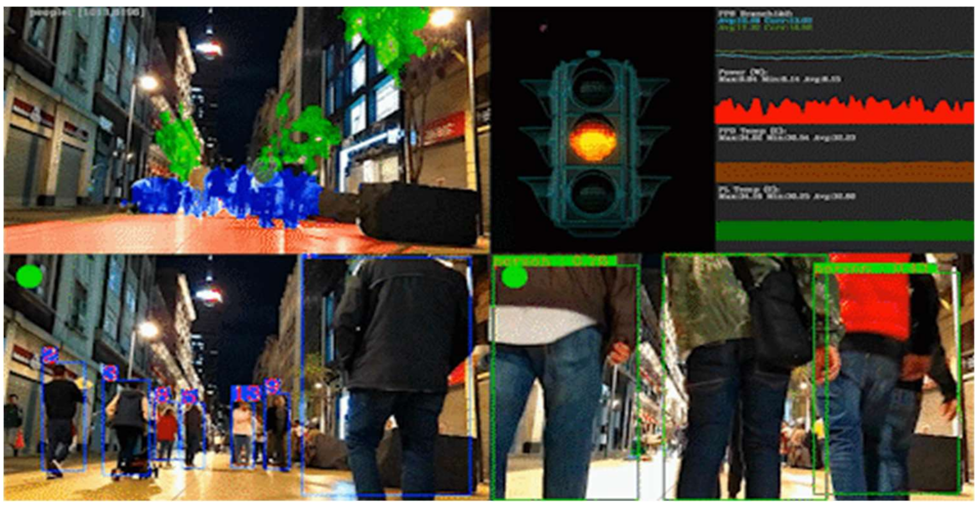


Figure 4 How YOLO works

How does the process works from video till model output?

The Raspberry Pi camera captures video footage, which serves as the input for the human detection process.

1. Preprocessing: The input video frames are preprocessed to enhance the quality and prepare them for analysis. This may involve resizing, normalization, and other image preprocessing techniques.
2. Model Inference: The preprocessed frames are passed through the YOLO and ResNet models for inference. The models analyze the frames and generate predictions regarding the presence and location of humans.
3. Post-processing: The output predictions from the models undergo post-processing steps to refine the results. This may involve filtering, non-maximum suppression, and thresholding to eliminate duplicate or low-confidence predictions.
4. Visualization: The final output is visualized by overlaying bounding boxes around detected humans on the original video frames, providing a clear indication of their presence.



3 Recommendations

1. Explore additional AI models: Consider integrating other AI models, such as object detection or semantic segmentation, to enhance the system's capabilities and support a broader range of applications.

2. Expand hardware compatibility: Investigate the compatibility of the software components with a wider range of hardware platforms to provide flexibility and accommodate diverse hardware configurations.

3. Optimize resource allocation: Further optimize the partitioning of subgraphs between the ARM CPU and the DPU tasks to maximize computational efficiency and improve overall system performance.

4 Limitations

1. Environmental Factors: The accuracy and reliability of the human detection system may be influenced by environmental factors such as lighting conditions, occlusions, or complex backgrounds. Adverse conditions, such as low-light environments or crowded scenes, may present challenges for accurate human detection. The system's performance may vary under different environmental conditions, and additional preprocessing techniques or model adaptations may be required to address such challenges.

2. Hardware constraints: While the Xilinx Kria KV260 Vision AI Starter Kit provides significant capabilities, it may have inherent limitations in terms of memory capacity, computational power, or

scalability. These limitations should be considered when deploying the solution in large-scale smart city deployments.

3. Training data availability: The project assumes the availability of sufficient and representative training data for the image classification model. However, the quality and diversity of the training dataset can significantly impact the model's accuracy and generalization capabilities.

5 Conclusion

The project successfully implemented an all-in-one self-adaptive computing platform for smart cities using the Xilinx Kria KV260 Vision AI Starter Kit, Raspberry Pi camera, PetaLinux, and Vitis Video Analytics SDK [2]. By combining the hardware capabilities, the software frameworks, and the intelligence of the YOLO and ResNet models, the project creates a powerful and adaptable computing platform for smart city applications. The human detection capability enables the platform to identify and track individuals in video footage, facilitating a range of potential use cases such as surveillance, crowd monitoring, traffic analysis, and more, contributing to the development of safer and more efficient smart cities. The platform's real-time processing capabilities ensured efficient analysis of video frames without significant latency, making it suitable for various smart city applications such as surveillance, crowd monitoring, and safety enhancement.

The final output of the system visualized the presence and location of humans by overlaying bounding boxes on the original video frames. This provided clear indications for further analysis and decision-making. The performance of the system was evaluated using metrics such as precision, recall, and accuracy to assess its effectiveness. Comparative analysis with existing methods or benchmarks was conducted to demonstrate the system's superiority in terms of accuracy, speed, and resource utilization.

References

[1] Xilinx, "Vitis™ Video Analytics SDK," AMD, [Online]. Available: https://github.com/Xilinx/VVAS.

[2] Xilinx, AMD, [Online]. Available: https://github.com/Xilinx.