Working Title - Hardware Acceleration: CNN Inference for an Emergency Vehicle Classifier

Juan Palacios
Electrical and Computer Engineering
Grand Valley State University
Grand Rapids, Michigan
palacioj@mail.gvsu.edu

1. Abstract

This paper demonstrates the deployment of an emergency vehicle classifier on an embedded device. Distracted driving poses a serious risk to the public's safety and can prevent emergency services from reaching those in need. To address this issue, a machine learning model is used to alert the driver of incoming emergency vehicles. Traditional machine learning models are trained with high-performance Graphical Processing Units (GPUs) with a high degree of accuracy. To achieve similar performance on low-power embedded devices, hardware accelerators are used to outsource heavier computational tasks. Field Programmable Gate Arrays (FPGA)-based accelerators offer reconfigurability to deploy a lightweight Convolutional Neural Network (CNN) model. The following objectives are laid out for this paper: the development of a machine learning model for emergency vehicle classification, deployment of this model to run on a low-power embedded device equiped with an FPGA, and assessing the real-time model inference performance. This research showcases the feasibility of inferring a lightweight model on a low-power embedded device with the help of an FPGA-based hardware accelerator.

2. Introduction

The purpose of this project is to explore the field of computer vision to implement object tracking on an FPGA development board. This novel feature has the potential of alerting drivers to moving objects while reserving their vehicle or identifying emergency vehicles in the rearview mirror. This project has three objectives:

1. Develop and train a machine learning model to track objects over time (Python development)
2. Deploy the model onto hardware using a combination of microcontroller and FPGA (HLS/HDL development)
3. Assess the performance on a Gentex FDM® for viability

Computer vision is a powerful tool enabling a computer to mimic the way humans see. the modern implementation of computer vision relies on machine learning. An sensor device captures an image and processing is done on an interpreting

device. A neural network algorithm performs the processing, providing useful information about the image [1]. There are four main categoreis of computer vision applications: image classification, object detection, object tracking, and optical character recognition [1]. For our purposes, we are focused on object tracking. First, an object must be detected in a single image. Next, our object must be followed throughout subsequent images. Finally, our device provides useful information in order to perform specific tasks based on our object tracking data [2]. There are several ways of implementing a model onto an FPGA: using kernel managed by a CPU (HLS), or writing a model in VHDL (HDL). FPGA kernels are not just instructions, but digital circuits (hence why we have pragmas, subset of C/C++).

A. Problem Statement

B. Scope

C. Paper Overview

3. Literature Review

A. Data Labeling

The purpose of data labeling is to provide additional information regarding a machine learning model's area of focus. This means adding contextual information such as a bounding box around a feature in a collection of training data. For example, in a data set of millions of vehicle images, adding labeling information around police vehicles helps a model learn the difference between it and a regular vehicle for better prediction outcomes. Labeled data is used in supervised learning and requires more time to manually label many thousands of images. Additional storage is required for the labeling information such as feature location in an image. There are ways to automate this process by employing a smaller machine learning model for data labeling, however, this process requires a human-in-the-loop for 100% validation. There are a few ways to implement data labeling: internal: using internal databases synthetic: augmenting databases programmatic: script approach for auto-labeling Data labeling is the most involved and important step in the machine learning model development process. If the quality of the training data is low, the model will perform poorly. The labels represent the ground-truth, meaning it is the expected feature for a model to guess correctly. There are many ways to classify objects embedded within an image. The most basic is having a single class per image. In a training set of vehicles, you can label them with a single class label. The next tier of classification involves using more than one label per class. This means that we still have a single class of object but our tags may reflect features regarding this class like a police car with sirens on or off. The most applicable classification schema may be the bounding box approach which just wraps the entirety of an object of interest.

Image annotation is the term for data labeling in computer vision applications. Simple image annotation/tagging is just a single phrase to describe the object of interest. Complex image annotation goes beyond this schema by allowing our label to include information about the number of object instances or areas in an image.

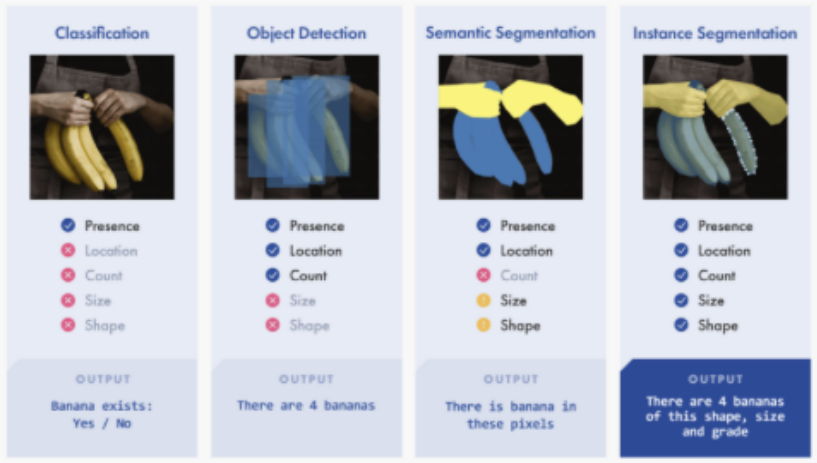

Figure 1: Image Annotation Schemas

4. Methodology/Implementation

5. Results & Discussion

6. Conclusion

7. Appendix

References

[1] Microsoft, “What Is Computer Vision? | Microsoft Azure --- azure.microsoft.com”. 2023.

[2] “What is Computer Vision? | IBM --- ibm.com”. 2023.