Real-time Object Detection on a Custom Built Edge Device Using TensorFlow Lite

Md Javed Ahmed Shanto[§], Md Raihan Subhan[§], Dong-Seong Kim[§], and Taesoo Jun[§]
[§]*Department of IT Convergence Engineering, Kumoh National Institute of Technology, Gumi 39177, Korea*
(shantoa729, raihan, dskim, taesoo.jun)@kumoh.ac.kr

***Abstract*—In recent times, there has been a notable increase in interest in on-device object detection. This technology allows for real-time processing of visual data without relying on a connection to a distant server. Additionally, it opens up several possibilities for application in other fields. Deploying these models on edge devices, particularly custom-made ones, presents several obstacles, mostly due to the limited processing power, memory, and storage capacity of edge devices, which are insufficient for AI training that demands significant computational power and storage. In addition, the development of ARM-based devices requires tailored model training and implementation that is specifically designed for the edge device. This article presents the implementation of an object detection model using a TensorFlow Lite model with a 94% accuracy. The model is designed to operate in real time on a custom-built edge device, making this work novel. Furthermore, it yields favorable outcomes in comparison to its stated criteria.**

***Index Terms*—Custom Built Edge Device, Object Detection, TensorFlow Lite.**

I. Introduction

Object detection is a crucial domain within computer vision. It finds applications in several domains, such as autonomous vehicles, cybersecurity, and robotics. It aids in the detection and recognition of objects in photos or movies. An essential aspect of object detection is the capability to do it locally on the device without requiring an internet connection. On-device object detection enables immediate processing and eliminates the delays and privacy concerns associated with transmitting data over the Internet.

We use TensorFlow for the purposes of machine learning and deep learning [1], but it necessitates a substantial amount of computational power and resources. TensorFlow Lite is particularly well-suited for deployment on edge devices. TensorFlow Lite is a costless deep-learning framework that enables developers to construct and use machine-learning models on edge devices [2]. Its architecture prioritizes compactness and speed, making it highly suitable for on-device activities such as object identification. To transform a TensorFlow model into a TensorFlow Lite model, one might use the TensorFlow Lite Converter. The TensorFlow Lite converter utilizes optimizations and quantization methods to minimize the dimensions and delay of the model. This assignment is completed with little to insignificant effect on detection or model accuracy. The TensorFlow Lite converter generates an optimized FlatBuffer format, indicated by the .tflite file extension, by using the original Tensorflow model. The TensorFlow Lite Converter landing page offers a Python API for the conversion of models (TensorFlow Lite) . This paper's contributions are as follows:

1) This study has successfully deployed an object detection model on a bespoke ARM-based device using the TensorFlow Lite model, which is a novel work.
2) This research provides a comprehensive performance analysis and demonstrates that the findings of this article are superior to those of any other current article.

The subsequent portion of the report delineates the methodologies used to construct and authenticate the suggested model. The outcomes of the suggested experiments are reported in Section III. These findings, together with the technique used, have successfully contributed to achieving the aim. Section IV discusses potential improvements alongside the conclusion.

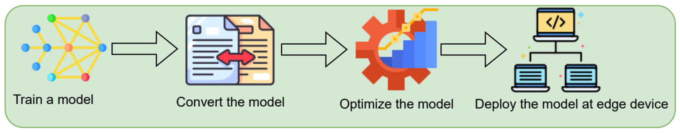

Fig. 1. Proposed Methodology for Detecting an Object using the Edge Device

II. Proposed Methodology

This article utilizes a specially constructed edge device. The architecture of the device is based on AArch64 and supports both 32-bit and 64-bit CPU modes. There are a total of 6 CPUs. The system has a configuration of one thread allocated per core, with a total of three cores assigned per socket. It has two sockets. The gadget in question is based on the ARM architecture and utilizes the Cortex-A53 processor. The CPU has a maximum clock speed of 2208 MHz and a minimum clock speed of 500 MHz. The product has a single USB-C connector for power, an HDMI port for connecting to a display, three USB ports, and one Ethernet port for internet connectivity. Additionally, this gadget is equipped with a Wi-Fi adapter, allowing us to use wireless internet connectivity.

We use the USB burning program to transfer the vim3-ubuntu-20.04-gnome-linux-4.9-fenix-1.3-221118-emmc operating system, which is a variation of Ubuntu developed by Khadas, to our device. The first step involves establishing a connection between the edge device and the host PC by using an A-type port and a C-type port. Once the operating system has been successfully installed on the device, we proceed to detach the power cord and operate the gadget using an adapter. It will start the Ubuntu operating system. Upon starting Ubuntu, it is necessary to acquire some libraries in order to execute the TensorFlow lite model on the edge device. The suggested technique is presented in a straightforward manner in reference 2.

TABLE I
RESULT COMPARISON OF THE PROPOSED AND EXISTING MODELS

Researches	Accuracy	Average FPS	Latency (ms)	Detected object	Edge Device
[2]	N/A	N/A	N/A	Any Objects	N/A
[3]	N/A	N/A	N/A	Only Dog	Raspberry Pi 4
[4]	91.76	N/A	358.95 (inference time)	Cardboard, Paper, Metal, Plastic, Glass	Arduino
[5]	N/A	4.5	N/A	Object detection for blind people	Raspberry Pi 4
This article's	**94.17%**	**15**	**37**	**Any Objects**	**Custom Build Device**

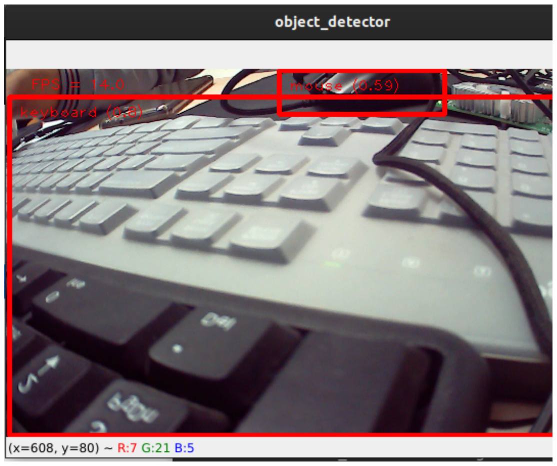

Fig. 2. The edge device is perceiving the items inside its field of vision

III. Experiment and Performance Analysis

This article employs Google's edge device's Convolutional Neural Network (CNN) model to validate the compatibility of the TensorFlow Lite (tf-lite) model with the device. This model has undergone training using the COCO dataset. We used the efficientdet-lite0 model, which has a default configuration of 50 epochs and a batch size of 64. It has a single step per execution. Although the option to use TPU is available, it is not currently being used. Additionally, the default verbosity level is set to 0. After performing the upgrade and update command on Ubuntu, we proceeded to clone one of the official Google scripts and then made unique modifications to it. In order to do this, we replicated the code from the tensorflow lite examples. Subsequently, we installed python3 to execute or modify the code as needed. In order to execute our code, it is important to set up a virtual environment (venv) to isolate it from the root libraries and ensure that all the required libraries are included inside it.

Upon the completion of the venv installation, it is necessary to activate the venv. By using the command "sh setup.sh", one may activate their virtual environment (venv). Next, go to the downloaded file by using the "cd" command. Then, execute the sh file by using the "sh setup.sh" command. There is a single defect in the code. One may resolve this issue by using a file with a lower version. In order to do this, we will use the following command: "python -m pip install –upgrade tflite-support==0.4.3". Next, we will proceed with the installation of the guvcview program specifically designed for the camera. Next, we will execute the detect.py script using our chosen model, which, in the context of this post, is "–model efficientdet_lite0.tflite". The camera will now automatically open and begin detecting the objects. Upon analyzing the results, it is evident that the work presented in this paper is very noteworthy and surpasses that of others. The accuracy of the system is 94.17%, with an average frame per second (FPS) of 15. Additionally, the latency of the system is just 37 milliseconds.

IV. Conclusion

This work used tensorflow lite to identify objects in a real-time scenario on a specifically constructed edge device. It attains a 94% level of accuracy while maintaining a minimal latency and delivering satisfactory frames per second (FPS). Our future endeavors will focus on enhancing precision by optimizing the frames per second (FPS) and integrating it into a practical scenario, such as a factory execution system.

Acknowledgment

This work was partly supported by the Innovative Human Resource Development for Local Intellectualization program through the Institute of IITP grant funded by the Korean government (MSIT) (IITP-2024-2020-0-01612, 33.3%) and by Priority Research Centers Program through the NRF funded by the MEST (2018R1A6A1A03024003, 33.3%) and This was also supported by the results of a study on the "Leaders in INdustry-university Cooperation 3.0" Project, supported by the Ministry of Education and the National Research Foundation of Korea(33.3%).

References

[1] M. J. A. Shanto, L. A. C. Ahakonye, A. Zainudin, J.-M. Lee, D.-S. Kim, and T. Jun, "An exploratory deep learning-based inventory management solution in the manufacturing execution system," , pp. 656–657, 2022.

[2] R. S. Praneeth, K. C. S. Akash, B. K. Sree, P. I. Rani, and A. Bhola, "Scaling object detection to the edge with yolov4, tensorflow lite," in *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, 2023, pp. 1547–1552.

[3] S. Swain, A. Deepak, A. K. Pradhan, S. K. Urma, S. P. Jena, and S. Chakravarty, "Real-time dog detection and alert system using tensorflow lite embedded on edge device," in *2022 1st IEEE International Conference on Industrial Electronics: Developments & Applications (ICIDeA)*. IEEE, 2022, pp. 238–241.

[4] N. C. A. Sallang, M. T. Islam, M. S. Islam, and H. Arshad, "A cnn-based smart waste management system using tensorflow lite and lora-gps shield in internet of things environment," *IEEE Access*, vol. 9, pp. 153 560–153 574, 2021.

[5] M. Konaite, P. A. Owolawi, T. Mapayi, V. Malele, K. Odeyemi, G. Aiyetoro, and J. S. Ojo, "Smart hat for the blind with real-time object detection using raspberry pi and tensorflow lite," in *Proceedings of the International Conference on Artificial Intelligence and its Applications*, 2021, pp. 1–6.