JURNAL INFOTEL
Informatics - Telecommunication - Electronics

Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

All-in-one computation vs. computational-offloading approaches: a performance evaluation of object detection strategies on android mobile devices

Muhammad Abdullah Rasyad[1*], Favian Dewanta[2], Sri Astuti[3]
[1,2,3] Telkom University
[1,2,3]Telekomunikasi Street No.01, Terusan Buahbatu - Bojongsoang, Sukapura, Kec. Dayeuhkolot, Kabupaten Bandung 40257, Indonesia
*Corresponding email: rasyadmar@gmail.com

Received 18 August 2021, Revised 30 September 2021, Accepted 26 October 2021

Abstract — Object detection gives a computer ability to classify objects in an image or video. However, high specified devices are needed to get a good performance. To enable devices with low specifications performs better, one way is offloading the computation process from a device with a low specification to another device with better specifications. This paper investigates the performance of object detection strategies on all-in-one Android mobile phone computation versus Android mobile phone computation with computational offloading on Nvidia Jetson Nano. The experiment carries out the video surveillance from the Android mobile phone with two scenarios, all-in-one object detection computation in a single Android device and decoupled object detection computation between an Android device and an Nvidia Jetson Nano. Android applications send video input for object detection using RTSP/RTMP streaming protocol and received by Nvidia Jetson Nano which acts as an RTSP/RTMP server. Then, the output of object detection is sent back to the Android device for being displayed to the user. The results show that the android device Huawei Y7 Pro with an average FPS performance of 1.82 and an average computing speed of 552 ms significantly improves when working with the Nvidia Jetson Nano, the average FPS becomes ten and the average computing speed becomes 95 ms. It means decoupling object detection computation between an Android device and an Nvidia Jetson Nano using the system provided in this paper successfully improves the detection speed performance.

Keywords – Object Detection, Nvidia Jetson Nano, Android Studio, RTSP, RTMP, TensorRT

I. INTRODUCTION

Object detection usage has been growing rapidly in recent years. Object detection gives computers the ability to recognize many things on input in the form of images, videos, and similar image forms [1]. By working together with augmented reality, it can do object recognition and tracking to aid the augmented reality accuracy. [2].

Regarding object detection in augmented reality, several works combine google glass with a PC server in [3], and other works use smartphones [4]. As for smartphone, there are various types of smartphones with different computational resources (CPU, RAM, and storage). However, those smartphones are mostly not equipped with a graphics processing unit (GPU) inside them, which will perform poorly in object detection [5],[6]. Therefore, those smartphones need to be supported by other devices that can leverage their computational ability in performing object detection tasks.

In recent years Nvidia has developed a microcomputer that affordable and able to operate object detection effectively with its dedicated GPU. It is proven in a paper by Luis et al. [7] who conducted a study on vehicle and pedestrian detection in rural areas using the Nvidia Jetson Nano microcomputer. In [7], the results showed that Nvidia Jetson Nano detection computation speed is around 1 – 10 seconds when processing several combined datasets consisting of 837 pedestrians and 681 vehicles in rural areas. In addition, Martina et al. [8] also conducted a study on deep learning applications in space with Nvidia Jetson Nano. The experiment in [8] showed that even though Nvidia Jetson Nano speed performance still loses to the cutting edge Nvidia GPU, further research on

testing Nvidia Jetson Nano capabilities for this usage is worth considering.

Considering the mobility of an Android device and the computational resource of Nvidia Jetson Nano, this paper discusses the computational performance of object detection conducted by an Android device versus an Android device supported by Nvidia Jetson Nano. Furthermore, the frame per second (FPS) value, the reflection of communication and computation delays between those systems, is compared and analyzed to decide whether offloading the computation process from Android to Nvidia Jetson Nano is efficient considering the delay between transmission.

II. RESEARCH METHODS

The system model depicted in Fig. 1, consists of two devices (an Android device and an Nvidia Jetson Nano) and a live video taken by the Android device camera.

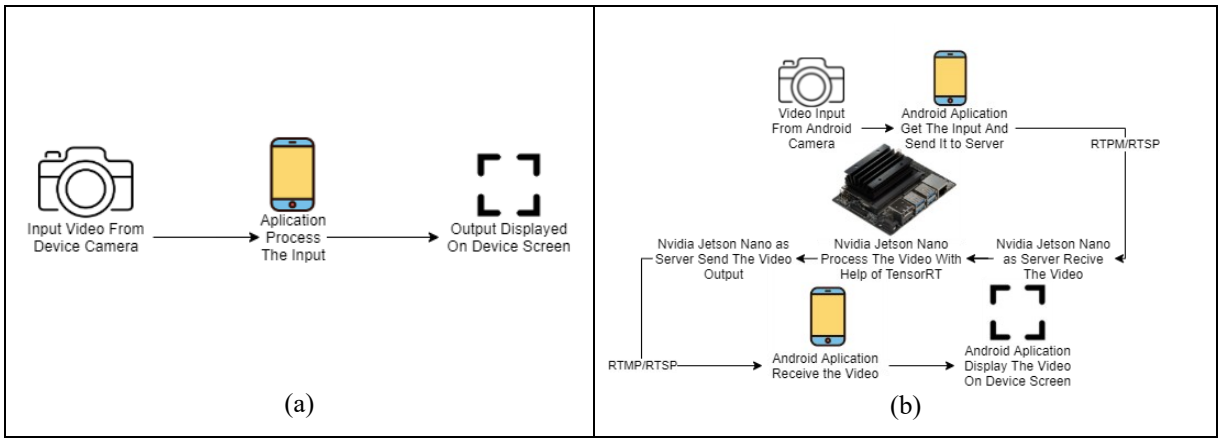

Fig.1. System diagram implemented in this work. (a) An object detection scheme is done on a mobile phone. (b) An object detection scheme using computational offloading into an NVIDIA jetson nano board.

Fig.1. (a) shows the all-in-one object detection computation conducted by an Android device, and Fig.1. (b) shows the computational offloading scheme from an Android device to an Nvidia Jetson Nano.

A. Communication Design Between Android Device and Nvidia Jetson Nano

We use two protocols for sending a live video from the Android device to the Nvidia Jetson nano. Both protocols send the video data via a wireless local area network. The first one is Real-Time Streaming Protocol (RTSP). RTSP is a protocol that controls data transmission in real-time over a computer network [9]. RTSP is used to stream digital media such as audio and video in real-time. RTSP also can be used by streaming receivers to control servers. In other words, RTSP can act as a remote-control network for multimedia servers [9]. And the second protocol is Real-Time Messaging Protocol (RTMP). Adobe use RTMP to stream audio, video, and data over the internet [10]. RTMP is a stateful protocol, meaning that the protocol gives the server the ability to observe the behavior of the stream receiver, such as when playing or pausing the received video [11].

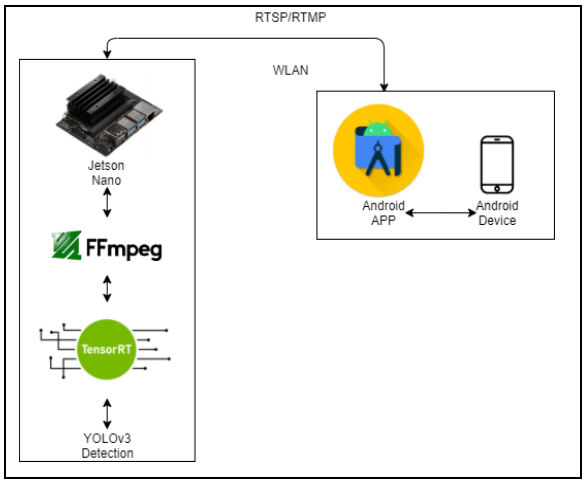

Fig. 2. Android device and nvidia jetson nano communication diagram.

The android device and Nvidia Jetson Nano communicate over an indoor wireless local area network connected to the device, Nvidia Jetson Nano, and a router. To exchange data, Nvidia Jetson Nano acts as a server and provides object detection services to devices.

B. Software and Hardware Requirements

To design the system some software and hardware requirements need to be prepared. In this system, two android devices are used for comparison. The hardware requirements are:

1. Nvidia Jetson Nano, used as a server to help object detection computation that has Quad-core ARM A57 1.43 GHz Central Processing Unit (CPU), 128-core Maxwell GPU, and 4 GB 64-bit LPDDR4 25.6 GB/s RAM.
2. Huawei Y7 Pro, a Huawei smartphone that runs on Android OS. This smartphone has an Octa-core 1.8 GHz Cortex-A53 CPU, Adreno 506 GPU, and 3 GB of RAM
3. Huawei Nova 5T, is a Huawei smartphone that runs on Android OS with Octa-core 2.6 GHz Cortex-A76 CPU, Mali-G76 MP10 GPU, and 8 GB RAM.

Huawei Y7 Pro with lower hardware capability is compared with Huawei Nova 5T, which has more hardware capability. Nvidia Jetson Nano helps both android devices to do object detection. The result when both devices work with Nvidia Jetson Nano and process object detection independently are compared.

Nvidia Jetson Nano is one of the microcomputers produced by NVIDIA to run Artificial Intelligence applications that are affordable by the broader community at affordable prices.

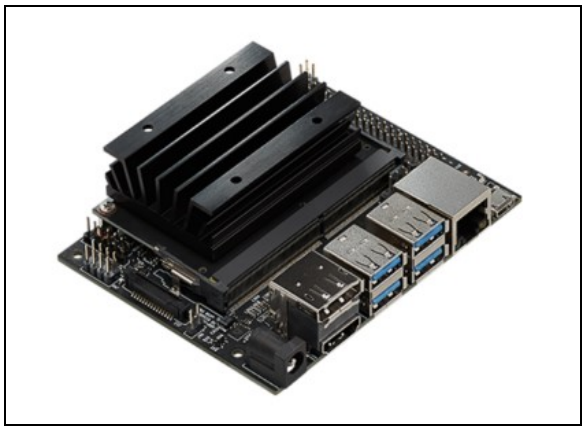

Fig. 3. Nvidia jetson nano [7].

The Jetson Nano is equipped with JetPack Software Development Kit (SDK) to aid performance [7]. JetPack SDK has several components in it, namely:

1. TensorRT, an SDK to aid deep learning performance, including Object Detection by maximizing GPU performance [12].
2. Compute Unified Device Architecture (CUDA) toolkit provides an environment for c/c++ developers to build applications with accelerated GPU [7].
3. CUDA Deep Neural Network (cuDNN), a library to accelerate GPU for deep neural networks [7].

Next is the software requirements. The software requirements are:

1. Android Studio, used to develop the android application. Applications made with Android Studio are native applications from Android [13]. It means that the initial developers of the Android operating system have designed Android Studio as software to create applications on it.
2. Python 3.6 is a programming language used in programs running on Nvidia Jetson Nano. Python programming language can do a variety of jobs. Some of which are numerical computing, web development, databases, network programming, and Image Processing [14].
3. OpenCV, one of the libraries in python. OpenCV is a library of programming functions that are widely used for image processing [15].
4. FFmpeg, an open-source multimedia framework for the transmission of multimedia such as audio and video. FFmpeg helps to encode and decoding video and audio data [16]. FFmpeg implemented in Nvidia Jetson Nano.
5. YoloV3-tiny is an object detection model that used to be the detector in android applications and Nvidia Jetson nano. Yolov3-tiny is one of the types of YoloV3 detection models. YoloV3 is the third generation of the Yolo detection model that provides a pre-trained model so we can use object detection easily and effectively [17]. When tested with Pascal Titan X GPU YoloV3 got 57.9% mean average precision (mAP) accuracy and 20 FPS of detection speed, while YoloV3-tiny got 33% mAP accuracy and 220 FPS of detection speed [18]. Therefore, we chose YoloV3-tiny because its accuracy is acceptable while having fast detection speed.

C. Measured Parameters

The test is carried out in both scenarios. The test is repeated 30 times and every test has a duration of 2 minutes. The test is carried out in a laboratory room with a size of 5x3 m.

To measure the success of both scenarios some parameters is retrieved on every repetition. The parameters to measure the object detection computation performance are detected video output frame per second (FPS) and detection computation speed per frame.

Detection video input FPS for both scenarios is set to be 15 FPS. When the output FPS is closer to input FPS it means the detection speed is close to real-time [19].

The parameters to measure Nvidia Jetson Nano and android device resource usage are CPU usage and RAM usage. And to measure the quality of transmission, we use throughput and delay between Nvidia Jetson Nano and android devices as parameters.

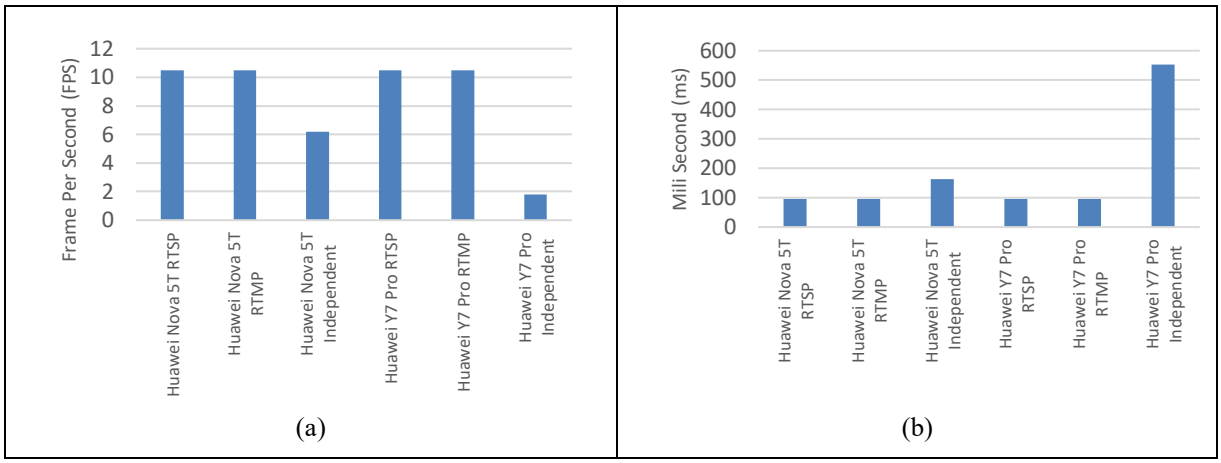

Fig. 4. Object detection computation performance comparison. (a) FPS comparison. (b) computation speed comparison.

III. RESULTS

A. Object Detection Computation Performance

Figure 4. (a) and Fig.4. (b) show the average values of all 30 tests. It shows the amount of FPS and computation speed comparison between android devices when offloading computation on Nvidia Jetson Nano with both protocols and when doing it independently.

From Fig. 4. (a), we can see that both devices have nearly similar FPS performance values when offloading computation on Nvidia Jetson Nano with both protocols. Huawei Nova 5T gets an average FPS value of 10.469 FPS with RTSP and 10.469 FPS with RTMP. Furthermore, Huawei Y7 Pro gets an average FPS value of 10.469 FPS with RTSP and 10.468 FPS with RTMP. The value is nearly the same as it indicates that differences in protocols and devices are not impacting the performance of Nvidia Jetson Nano on offloading the object detection computation. However, when both devices are doing object detection computation independently, the performance becomes poor. Huawei Nova 5T has better hardware performance gets an average FPS value of 6.18 FPS. Moreover Huawei Y7 Pro get an average FPS value of 1.82 FPS.

The average value of computation speed is nearly similar on both devices and protocols. Huawei Nova 5T gets an average computation speed value of 95.51 ms with RTSP and 95.51 ms with RTMP. Furthermore, Huawei Y7 Pro gets an average computation speed value of 95.52 ms with RTSP and 95.52 ms with RTMP. It means that the computing speed of the Nvidia Jetson Nano is not affected by device and protocol differences. When doing computation independently, FPS and computation performance also becomes poor. Huawei Nova 5T has better hardware performance gets an average computation speed value of 162.78 ms. And Huawei Y7 Pro gets an average computation speed value of 552.29 ms.

As shown in Fig. 4 both devices' performances are increased when offloading computation on Nvidia Jetson Nano. It means the computational offloading scheme from an Android device to an Nvidia Jetson Nano successfully improve both devices performance.

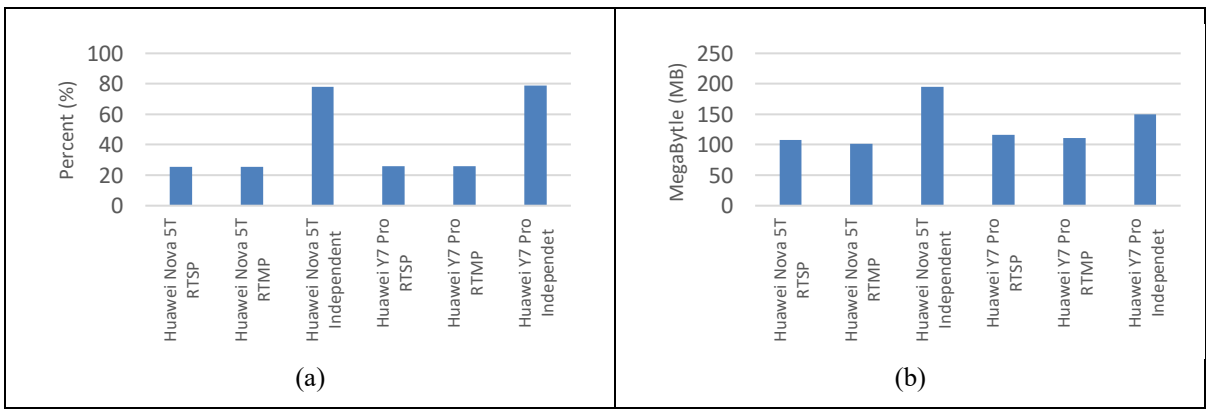

Fig. 5. Android devices resource usage (a) CPU usage comparison. (b) RAM usage comparison.

B. Android Devices Resource Usage

Figure 5 shows the average values of all 30 tests. Fig.5 (a) shows the comparison of CPU usage, and Fig.5 (b) shows the comparison of RAM usage when offloading computation on Nvidia Jetson Nano with both protocols and when doing it independently.

Jetson Nano. When computing independently, Huawei Nova 5T gets an average value of 78.09 percent. And Huawei Y7 Pro gets an average CPU usage value of 25.81 percent with RTSP and 25.82 percent with RTMP when offloading computation on Nvidia Jetson Nano. When computing independently, Huawei Y7 Pro gets an average value of 78.88 percent. As we can see, the value of both android devices' CPU usage is nearly similar, and CPU usage when both android devices are computing independently is vastly higher. It means the computational offloading scheme from an Android device to an Nvidia Jetson Nano successfully reduces CPU usage of both android devices, and the differences in android devices and protocols are not impacting android devices CPU usage when offloading computation on Nvidia Jetson Nano.

Figure 5 (b) shows that Huawei Nova 5T gets an average RAM usage value of 108.35 MB with RTSP and 102.07 MB with RTMP when offloading computation on Nvidia Jetson Nano. When computing independently, Huawei Nova 5T gets an average value of 195.38 MB. Moreover, Huawei Y7 Pro gets an average RAM usage value of 115.76 MB with RTSP and 111.13 MB with RTMP when offloading computation on Nvidia Jetson Nano. When computing independently, Huawei Y7 Pro gets an average value of 149.89 MB. Same as before, both android devices have a nearly similar RAM usage value, and RAM usage when both android devices are computing independently is vastly higher. It means the computational offloading scheme from an Android device to an Nvidia Jetson Nano successfully reduces RAM usage of both android devices, and the differences in android devices and protocols are not impacting android devices RAM usage when offloading computation on Nvidia Jetson Nano.

C. Nvidia Jetson Nano Resource Usage

The data in this section are retrieved when the android device works with the help of Nvidia Jetson Nano. Therefore, the retrieved data are Nvidia Jetson Nano CPU usage and Nvidia Jetson Nano RAM usage.

CPU and RAM usage of Nvidia Jetson Nano is retrieved using operating system default task manager. The average value of all 30 tests is shown in this graph.

When using RTMP, the Nvidia Jetson Nano CPU usage is greater than when using RTSP to communicate with Huawei Nova 5T or Huawei Y7 Pro. CPU usage when communicating using the RTSP protocol with the Huawei Nova 5T has an average of 26.25 percent, and when communicating with the Huawei Y7 Pro, the average value is 26.24 percent. Then, CPU usage when communicating using the RTMP protocol with the Huawei Nova 5T has an average of 26.74 percent, and when communicating with the Huawei Y7 Pro, the average value is 26.77 percent.

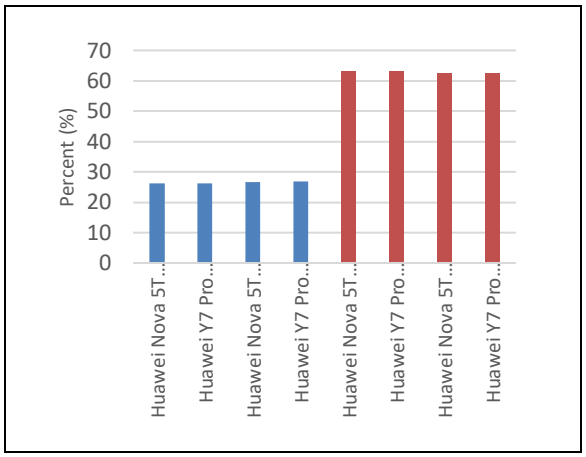

Fig. 6. CPU usage of Nvidia jetson nano when working with android device.

When the Nvidia Jetson Nano communicates with RTMP with the Huawei Nova 5T, its CPU usage values have a similar overall average as when communicating with the Huawei Y7 Pro using the same protocol. Furthermore, when using RTSP, the Nvidia Jetson Nano CPU value also has a similar overall average on both devices. It means that the difference in devices does not affect the CPU computing load of the Nvidia Jetson Nano, but the use of the type of protocol does.

Based on Fig.6, the RAM used by the Nvidia Jetson Nano when using RTSP is greater than the RTMP on both devices. When communicating using the RTSP protocol with the Huawei Nova 5T, the RAM usage is an average of 62.99 percent. When communicating with the Huawei Y7 Pro, the average value is 63.01 percent. Then, RAM usage when communicating using the RTMP protocol with the Huawei Nova 5T has an average of 62.49 percent. When communicating with the Huawei Y7 Pro, the average value is 62.51 percent. Furthermore, just like CPU usage, RAM usage has an average similarity when both devices use the same protocol. It means that the difference in devices does not affect the RAM load of the Nvidia Jetson Nano, but the use of the type of protocol does.

D. Communication Performance

Delay and throughput during communication are indicators of the successful delivery of data. Input data sent from the device to the Nvidia Jetson Nano is called Upload. Moreover, the output sent from the Nvidia Jetson Nano to the device is called Download. Delay and throughput are retrieved using Wireshark when Nvidia Jetson Nano and android devices communicate with each other. The average delay value of all 30 tests is shown in this graph.

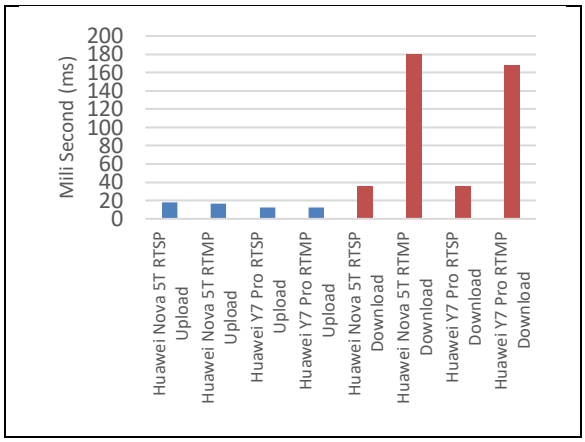

Fig. 7. Delay comparison when devices are working with Nvidia Jetson Nano.

Figure 7 shows upload and download delays. When uploading, the delay from the Huawei Nova 5T RTSP has an average of 18.22 ms, and when using RTMP, the average delay is 16.48 ms. For Huawei Y7 Pro, the average delay when using RTSP is 12.22 ms, and when using RTMP, the average is 12.56 ms. The average value of the upload delay in Fig. 8 differs on the order of 1 ms. The possibility of this difference occurs, is due to the instability of the wireless connection.

When downloading, the delay from RTMP far exceeds RTSP on either Huawei Nova 5T or Huawei Y7 Pro, as shown in Fig.7. For example, the average delay of Huawei Nova 5T when using RTSP is 35.95 ms and when using RTMP is 180.05 ms. Furthermore, for Huawei Y7 Pro, the average when using RTSP is 36.08 ms, and when using RTMP, the average value is 167.92 ms.

It means that object detection displayed on the device through RTMP has a higher delay than RTSP because upload and download delays contribute to delays in device object detection display.

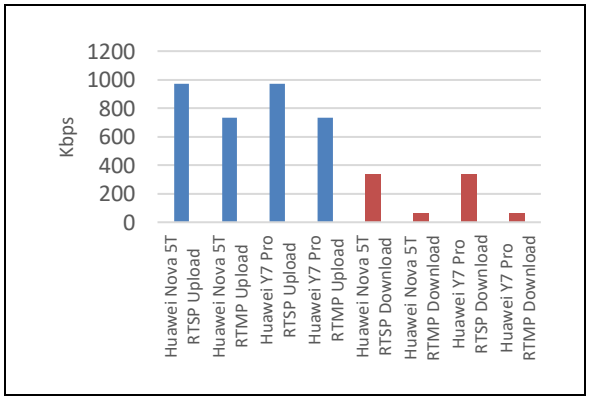

Fig. 8. Throughput comparison when devices are working with Nvidia jetson nano.

In Fig. 8, Huawei Nova 5T has an average upload throughput of 969 Kbps when using RTSP and 735 Kbps when using RTMP. While the Huawei Y7 Pro has an average of 971 Kbps when using RTSP and 735 Kbps when using RTMP. In both android devices, RTSP upload throughput is higher than RTMP. Same as when uploading, the throughput during the download process on the Huawei Nova 5T and Huawei Y7 Pro using RTSP are much higher than RTMP. When Huawei Nova 5T uses RTSP, the average throughput is 337 Kbps, and when using RTMP, the average is 65 Kbps. Meanwhile, when Huawei Y7 Pro uses RTSP, the average value is 338 Kbps, and when using RTMP, the average value is 64 Kbps.

It means that the object detection results when using RTMP are experiencing greater lag than RTSP on both devices. Because large throughput indicates smooth communication between the device and the Nvidia Jetson Nano.

IV. DISCUSSION

As expected, the Nvidia Jetson Nano successfully helps the android device to do object detection if the transmission delay and throughput are not concerned. However, bad delay and throughput of the transmission causes some lag and delay when transmitted detections are displayed in the android device screen. RTSP got a better result. It is tolerable, as we can see in Fig. 7 and 8. However, RTMP not perform so well that it causes 3-6 second delay and lag when displayed on the device screen.

V. CONCLUSION

Decoupling computation of object detection between an Android device and an Nvidia Jetson Nano using the system provided in this paper successfully improves the detection speed performance. The results show that the Huawei Y7 Pro android device, which has an average FPS performance of 1.82 and an average computing speed of 552 ms, gets better when working with Nvidia Jetson Nano. The average FPS becomes 10, and the average computing speed average becomes 95 ms. For further work, other video transfer protocol such as WebRTC, FTL, and SRT needs to be tested. Hopefully, it can improve the transmission delay and throughput. For further improvement in aiding detection of the android device, the Nvidia Jetson Nano that acts as a server needs to be upgraded. The suggested upgrade are Jetson TX2Series, Jetson Xavier NX, dan Jetson AGX XavierSeries.

ACKNOWLEDGMENT

The authors sincerely thank Telkom University for allowing this research and The APTRG Laboratory of Telkom University, who provide tools for this research..

REFERENCES

[1] A. Kumar, J. Zhang, and H. Lyu, “Object detection in real time based on improved single shot multi-box detector algorithm,” *EURASIP J. Wirel. Commun.*

Netw., vol. 2020, Oct. 2020, doi: 10.1186/s13638-020-01826-x.

[2] H. Le, M. Nguyen, W. Q. Yan, and H. Nguyen, "Augmented reality and machine learning incorporation using yolov3 and arkit," *Appl. Sci.*, vol. 11, no. 13, pp. 1–19, 2021, doi: 10.3390/app11136006.

[3] Z. He *et al.*, "Design and implementation of augmented reality cloud platform system for 3D entity objects," *Procedia Comput. Sci.*, vol. 131, pp. 108–115, 2018, doi: 10.1016/j.procs.2018.04.192.

[4] D. Ambarwulan and D. Muliyati, "The Design of Augmented Reality Application as Learning Media Marker-Based for Android Smartphone," *J. Penelit. Pengemb. Pendidik. Fis. p-ISSN 2461-0933 e-ISSN2461-1433*, vol. 2, no. 1, pp. 73–80, 2016.

[5] E. Buber and B. Diri, *Performance Analysis and CPU vs GPU Comparison for Deep Learning*. 2018.

[6] P. Sowa and J. Izydorczyk, *Darknet on OpenCL: A Multi-platform Tool for Object Detection and Classification*. 2020.

[7] L. Barba-Guaman, J. E. Naranjo, and A. Ortiz, "Deep learning framework for vehicle and pedestrian detection in rural roads on an embedded GPU," *Electron.*, vol. 9, no. 4, pp. 1–17, 2020, doi: 10.3390/electronics9040589.

[8] M. Lofqvist and J. Cano, "Accelerating Deep Learning Applications in Space," vol. 44, no. 0, 2020, [Online]. Available: http://arxiv.org/abs/2007.11089.

[9] Y. Syaifudin, I. Rozi, R. Ariyanto, R. Erfan, and S. Adhisuwignjo, "Study of Performance of Real Time Streaming Protocol (RTSP) in Learning Systems," *Int. J. Eng. Technol.*, vol. 7, pp. 216–221, Dec. 2018, doi: 10.14419/ijet.v7i4.44.26994.

[10] X. Li, M. Darwich, M. A. Salehi, and M. Bayoumi, "A survey on cloud-based video streaming services," in *Advances in Computers*, vol. 123, 2021, pp. 193–244.

[11] T. Ruether, "Streaming Protocols: Everything You Need to Know (Update)," 2021. https://www.wowza.com/blog/streaming-protocols (accessed Aug. 09, 2021).

[12] B. Patel, V. Sanchez, and J. Anderson, "Deep Learning Inference on PowerEdge R7425," *Dell EMC Tech. White Pap.*, 2019.

[13] N. Verma, S. Kansal, and H. Malvi, "Development of Native Mobile Application Using Android Studio for Cabs and Some Glimpse of Cross Platform Apps," *Int. J. Appl. Eng. Res.*, vol. 13, no. 16, pp. 12527–12530, 2018, [Online]. Available: http://www.ripublication.com.

[14] K. R. Srinath, "Python – The Fastest Growing Programming Language," *Int. Res. J. Eng. Technol.*, pp. 354–357, 2017, [Online]. Available: https://www.irjet.net/archives/V4/i12/IRJET-V4I1266.pdf.

[15] A. Brdjanin, A. Akagic, D. Džigal, and N. Dardagan, *Single Object Trackers in OpenCV: A Benchmark*. 2020.

[16] X. Wu, P. Qu, S. Wang, L. Xie, and J. Dong, *Extend the FFmpeg Framework to Analyze Media Content*. 2021.

[17] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv*, 2018.

[18] A. Redmon, Joseph and Farhadi, "YOLO: Real-Time Object Detection," 2018. https://pjreddie.com/darknet/yolo/ (accessed Aug. 13, 2021).

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.