Febrian Ardiansyah, M., William, T., Ibrahiem Abdullaziz, O., Wang, L.C., Tien, P.L. y Yuang, M.C. (2020). EagleEYE: Aerial Edge-enabled Disaster Relief Response System. In *2020 European Conference on Networks and Communications (EuCNC)*, pp. 321-325.

DOI: https://doi.org/10.1109/EuCNC48522.2020.9200963

EagleEYE: Aerial Edge-enabled Disaster Relief Response System

Muhammad Febrian Ardiansyah[1], Timothy William[2], Osamah Ibrahiem Abdullaziz[3],
Li-Chun Wang[4], Po-Lung Tien[5], Maria C. Yuang[6]
[1345]Department of Electrical and Computer Engineering
[26]Department of Computer Science
National Chiao Tung University, Taiwan
[1]mfardiansyah.eed08g@nctu.edu.tw, [2]timothywilliam.cs06g@g2.nctu.edu.tw,[3]yabolahan.04g@g2.nctu.edu.tw,
[4]lichun@g2.nctu.edu.tw, [5]polungtien@gmail.com, [6]mariayuang@gmail.com

***Abstract*—The fifth generation (5G) mobile network has paved the way for innovations across vertical industries. The integration of distributed intelligent edge into the 5G orchestrated architecture brings the benefits of low-latency and automation. A successful example of this integration is exhibited by the 5G-DIVE project, which aims at proving the technical merits and business value proposition of vertical industries such as autonomous drone surveillance and navigation. In this paper, and as part of 5G-DIVE, we present an aerial disaster relief system, called EagleEYE, which utilizes edge computing and machine learning to detect emergency situations in real-time. EagleEYE reduces training time by devising an object fusion mechanism which enables reusing existing datasets. Furthermore, EagleEYE parallelizes the detection tasks to enable real-time response. Finally, EagleEYE is evaluated in a real-world testbed and the results show that EagleEYE can reduce the inference latency by 90% with a high detection accuracy of 87%.**

***Index Terms*—Low-latency computing, Object detection, Container, Edge computing.**

I. Introduction

Today, the fifth-generation (5G) mobile network has paved the way for innovations across vertical pilots. 5G transforms networks into intelligent orchestrated infrastructures and cements powerful relationships between verticals and operators. This opens a new breed of business value propositions for vertical industries such as autonomous drone surveillance and navigation. Autonomous drones will have a major positive impact in areas including agriculture, communication, and disaster relief [1].

In disaster relief area, drones have been extensively utilized because of their size and flexibility in operating environment [2]. In [3], an aerial disaster relief response architecture is proposed for victim detection. Using sensors such as laser scanner and infrared depth camera, a 3D reconstruction of the area is created to aid in the detection. Similar architecture was developed in [4] where a drone with electromagnetic sensor is utilized to perform victim detection. Another cloud-based multi-drone approach is proposed in [5] for disaster relief response. Although these aforementioned efforts can detect disaster victims, they lack real-time capability and several do not provide latency measurements.

Machine learning has also been used in disaster response relief. In [6], Haar cascade classifier is utilized for real-time detection of people and vehicles. The author reports a detection rate of approximately 70% for people in each image frame. However, the use of Haar cascade classifier for detection and classification has a drawback of being application specific. On the other hand, Convolutional Neural Network (CNN) has been widely used for object detection and classification [7]–[11]. Unlike Haar cascade classifier, CNN feature set is learned automatically during training. In [7], the use of CNN for aiding a SAR team in a snow avalanche victim-detection scenario is proposed. The author reports an accuracy of up to 97.59% and processing latency of $2.8s$ per image frame. In [8] and [9], two-stage object detection and classification is proposed. However, this two-stage approaches produce a high inference latency and are unable to perform the detection in the real-time. Another method for object detection and classification called single-shot detection is developed in [10], [11]. Unlike the two-stage detection, this scheme skips one of the stages in its two-stage counterpart and hence has a lower inference time latency.

Nevertheless, the realization of these disaster relief response systems gives rise to new challenges of meeting latency requirement in addition to efficient deployment, reliability, and scalability. To rise to these challenges, two key technologies, mobile edge computing (MEC) and network function virtualization (NFV), have emerged as solutions where applications and services are brought closer to the edge of the network [12]. While MEC eliminates traffic backhauling and thus end-to-end latency, NFV decouples network functions and applications from the underlying hardware, and facilitates them in software over commodity hardware. Together, MEC and NFV could be complemented by machine learning and cognitive techniques to orchestrate and manage the virtualization environment.

In our H2020 5G-DIVE project [13] (described in Sec. II), we aim to prove the technical merits and business value proposition of vertical industries built around distributed intelligent edge system. In a nutshell, 5G-DIVE architecture consists of three logical components namely, 1) Edge and Fog Computing System (EFS) that is comprised of edge and fog resources, 2) Orchestration and Control System (OCS) - that manages, controls, orchestrates, and federates one or more EFS(s), and

3) 5G-DIVE Elastic Edge Platform (DEEP) - that supports vertical industries operations, management, and automation of businesses processes on-top of OCS and EFS.

In this paper, we propose an **A**eri**a**l Ed**g**e-enab**l**ed Disaster Relief Respons**e** Sy**s**te**m**, referred to as EagleEYE, as one of the major subsystems in the DEEP of 5G-DIVE. It builds on top of the work presented in [11] for its superior performance and simultaneously provide low-latency object detection by parallelizing computing tasks at the edge of the network. Equally important, EagleEYE can seamlessly adjust to various computational loads, and hence it is scalable. The contributions of this work are threefold:

- *Objects fusion* - we devise a mechanism called Merged Object Detection (MOD) to enable reusing existing datasets for the purpose of emergency detection.
- *Low-latency object detection* - we parallelize computing tasks in a virtualized edge environment to enable real-time object detection for emergency response.
- *Experimental evaluation* - we evaluate EagleEYE detection precision and computing latency on a real-world environment.

The rest of the paper is organized as follows. Section II provides an overview of the 5G-DIVE architecture. Section III presents the proposed EagleEYE system and the low-latency object detection scheme. Section IV shows our experimental results while we give our concluding remarks in Section V.

II. 5G-DIVE Overview

The 5G-DIVE project [13] aims to facilitate intelligent infrastructure orchestration and automation for 5G networks. It targets end-to-end 5G vertical pilots and is built around distributed edge and fog system incorporating intelligence located at the proximity of the users. 5G-DIVE architecture is comprised of three logical components as follows (see Figure 1):

- *Edge and Fog Computing System (EFS)* - is a logical system comprised of edge and fog resources belonging to a single administrative domain.
- *Orchestration and Control System (OCS)* - is a logical system that manages, controls, orchestrates, and federates one or more EFS(s). It is designed to address challenges such as heterogeneity, mobility and volatility of resources.
- *5G-DIVE Elastic Edge Platform (DEEP)* - is a logical system that supports vertical industries operations, management, and automation of businesses processes on-top of OCS and EFS.

The DEEP subsystem plays an important role in the overall architecture. It supports vertical industries with features such as day-by-day operations, automation, and management of business processes on-top of an edge and fog infrastructure. DEEP has three main components namely, Data Analytics Support Stratum (DASS), Intelligence Engine Support Stratum (IESS), and Business Automation Support Stratum (BASS). DASS acts as a support for collecting insightful information for vertical industries, while IESS enriches their applications and operations with artificial intelligence. Lastly, BASS enforces policies into the platform by external OSS/BSS systems.

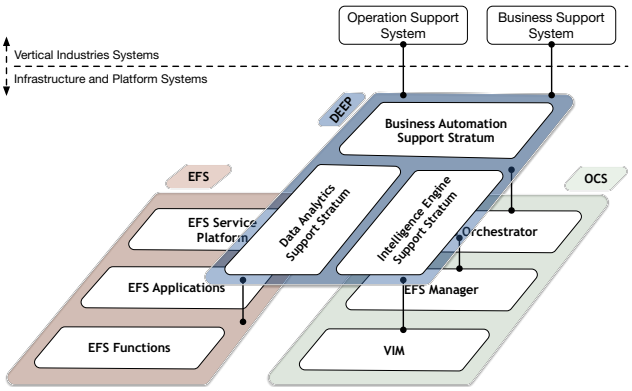

Figure 1: 5G-DIVE reference architecture [13].

III. Aerial Edge-enabled Disaster Relief Response System

In a disaster-impacted area where ground infrastructure is damaged, drones can be deployed to provide disaster relief. Drones can be utilized to provide aerial video surveillance and assessment of disaster impacted areas. This aerial video surveillance can be leveraged further by an intelligent system that is runs on the edge.

The intelligent system is able to provide insightful information to aid for disaster relief response by processing and inferring on the video surveillance data. One such information is the GPS location of Person-in-need-of-Help (PiH) in a disaster-impacted area. To accommodate this scenario, we propose a disaster relied system called EagleEYE which stands for: **A**eri**a**l Ed**g**e-enab**l**ed Disaster Relief Respons**e** Sy**s**te**m**. Deployed at the edge with high-computation capability, EagleEYE is able to process the video surveillance data from aerial drones and provide real-time PiH GPS location information.

A. EagleEYE Architectural Model

The EagleEYE system is built on top of 5G-DIVE architecture to provide a real-time object detection service from the edge to support disaster relief response team. Figure 2 shows the EagleEYE system in detail. The workflow of the EagleEYE system is detailed in the following logical steps.

Step 1: *Stream Video and GPS Data* - aerial video surveillance stream alongside GPS information taken by the drone are transmitted to the edge through the 5G wireless communication network. We assume that the edge will be in the same location as the gNB.

Step 2: *Process Data* - captured video stream will be chopped into image frames, While the received GPS information will be stored into the database.

Step 3: *Allocate Task* - a load balancer will allocate these image frames in a Round Robin fashion to available worker nodes for performing object detection and classification. The number of worker nodes can be

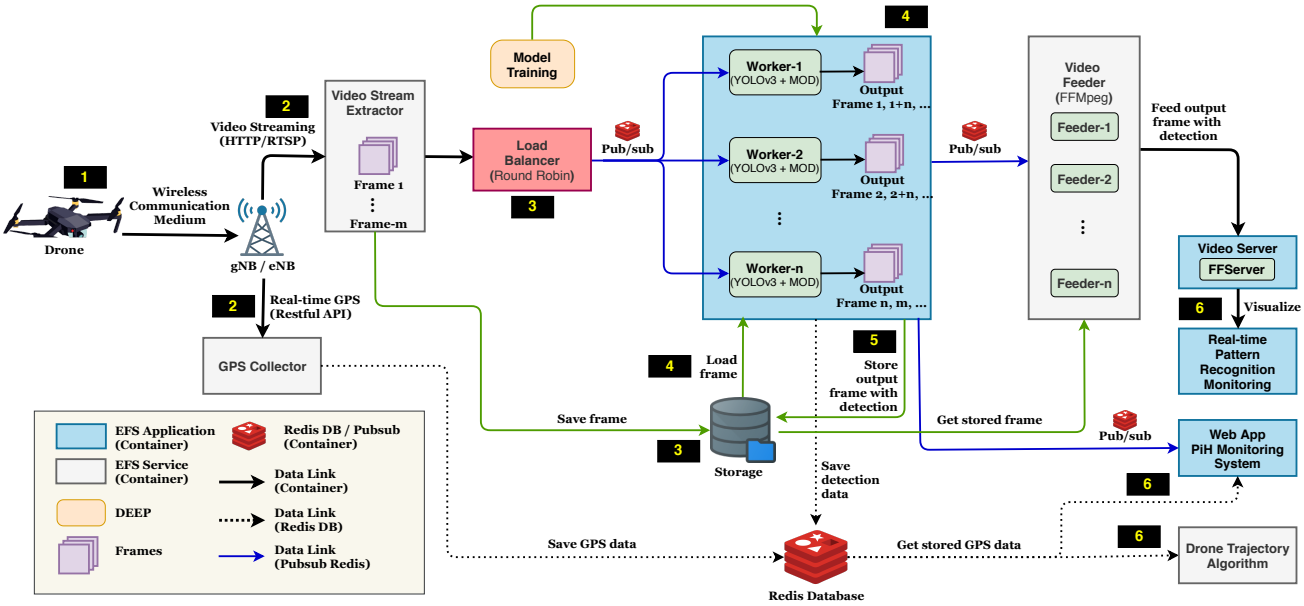

Figure 2: EagleEYE architectural model

scaled up/down according to the number of drones and the required object detection and classification latency.

Step 4: *Perform Object Detection & Classification* - worker nodes will work in parallel to process the input image frames. The output of these worker nodes will be an image frame that is marked with Bounding Box (BBox) information of *'Person'* and *'Flag'* object.

Step 5: *Store Output Frame* - BBox marked image frame are stored in the storage for later use.

Step 6: *Monitoring* - marked image frames are displayed to a dashboard in real-time with the respective PiH GPS information. Drone GPS waypoint are also provided to update the drones trajectory.

B. Object Detection using CNN

EagleEYE adopts YOLOv3 object detection model [11]. This algorithm uses CNN as its backbone for object detection and classification. For our use case, we train YOLOv3 model to detect two object classes namely, *Person* and *Flag*. We chose those two object classes, specifically as a way to simulate the PiH mentioned in the use case scenario. Our rationale is that if the model was able to detect and classify both '*Person*' and '*Flag*' object, then we will be able to devise an algorithm that can correlate the objects together as a PiH. We have implemented this correlation algorithm, and it will be explained in further detail in the next sub-section.

The training for the model was done using a mix of publicly available datasets, (1) COCO Dataset [14] and (2) Google Open Image Dataset [15]. As to our knowledge, there is no single dataset that has both the '*Person*' and '*Flag*' objects. We train our model using the pre-trained weight provided by the YOLOv3 original author [16].

C. Merged Object Detection

We build the MOD algorithm on top of the object detection pipeline (See figure 2). It brings the benefit of using public dataset and avoids the extra efforts of creating a custom training dataset. To successfully pair the objects in real-time, MOD algorithm requires to collect at least one *Person* object and one *Flag* objects from YOLOv3 [11]. If this condition is fulfilled, then the algorithm checks whether these objects exist in the output frame that has a valid intersection. An illustration of a valid intersection can be seen in Figure 3. In a successful merge, a bounding box will be drawn in the final output frame. Otherwise, no bounding box will be drawn. The pseudo-code of the MOD can be found in Algorithm 1.

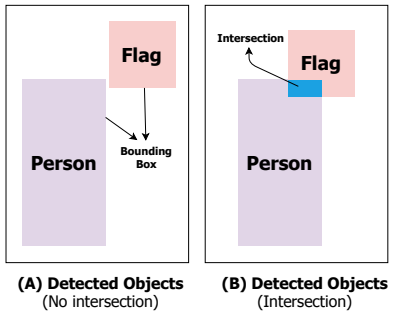

Figure 3: Person and flag bounding boxes intersection

Algorithm 1 Merged Object Detection

Require:
Frame F; $F = \{f_1, f_{m+1}, ..., f_M\}$
Person Object $\sum_{k=1}^{K} O^P$; $O^P = \{O_1^P, O_{k+1}^P, ..., O_K^P\}$
Flag Object $\sum_{l=1}^{L} O^F$; $O^F = \{O_1^F, O_{l+1}^F, ..., O_L^P\}$
Each object O_k^P and O_l^F consists of (x_1, y_1, x_2, y_2)
Intersection between two objects $I = \{0\ or\ 1\}$
Number of neighbours $k = 1$

Ensure:
Collection of Person and Flag objects pairs.

for each frame f_m **do** ▷ Object Detection outputs
  if $K \geq 1\ and\ L \geq 1$ **then**
    for $l = 1$ to L **do**
      Set an empty valid intersection $I^V = \{\}$;
      for $k = 1$ to K **do**
        if $I(O_k^P, O_l^F) = 1$ **then** ▷ Intersection
          Add into I^V;
        end if
      end for
      Perform kNN between O_l^F and $\forall I^V$;
      Pair O_l^F with the nearest O_l^P;
      Label as a 'person waving flag' object;
      Merge O_k^P and O_l^F;
      Draw the merged bounding box;
      Delete key k from O^P;
    end for
  end if
end for

IV. Experimental Results

A. Experimental Setup

EagleEYE system is deployed on top of OPTUNS edge data center [17]. OPTUNS provides crucial features, such as high scalability, high bandwidth capability, ultra-low latency communication between servers, and fault tolerance. In our experiment, we make an assumption that there is no communication latency between servers. In our setup, we run the object detection and classification model in a server equipped with 48 cores CPU running at 3GHz frequency, 128GB of RAM and a NVIDIA Tesla V100 GPU with 32GB of VRAM. As for the drone video surveillance, we simulate it with a pre-recorded FullHD video clip[1] and is streamed from another machine in the local network. FFmpeg and FFserver [18] software are used to stream and capture the video clip respectively.

We perform the weight training for our object detection model by using a GitHub fork from [19]. In total, we provide four self-trained weights and named them as Trained Model (TM-01 - TM04). We use weight TM-04 in our testing. As for the object detection model itself, we adopt the work from [20]. The GitHub Project that we used from [20] is a YOLOv3 object detection that is based on [11] work but re-written in PyTorch. Our MOD algorithm is added on top of [20].

[1]https://youtu.be/YrVeDM0kafg

For the latency testing, we measure the latency that the EagleEYE system needs to perform the whole image processing pipeline, from capturing the video stream until finishing the object detection pipeline. It also includes the time that the EagleEYE system needs to perform merged object detection.

As for the deployment, we logically map each component into 5G-DIVE architecture [13] by denote the blocks into three logical components: EFS application, EFS service and EFS function. We deploy six type of micro-services described as follows:

- *Video Server*. A virtualized video server on top of FF-Server to host HTTP video stream. It is categorized as an EFS application.
- *Load Balancer*. A system to balance each frame collected from HTTP stream into each corresponding YOLOv3 workers. It is categorized as an EFS function.
- *YOLOv3 Worker*. Virtualized YOLOv3 object detection worked in parallel. We are deploying up to 6 workers in this experiment. They are categorized as EFS applications.
- *Web App PiH Monitoring System*. A web-based dashboard to monitor the status of located *Person waving flag*. It is categorized as an EFS application.
- *Redis Database*. A key-value memory-based database. We utilize this tool as a publish/subscribe message protocol as well. It is categorized as an EFS service.
- *Real-time Object Detection Monitoring*. An application to display detection output. It is categorized as an EFS application.

B. Training Result of Object Detection Model

The results for testing and the training dataset used can be seen in Table II and Table I. The metrics used to assess the performance of TM are mean Average Precision (mAP) [21]. The mAP results were obtained by running the Object Detection model on the pre-recorded FullHD video clip. We can see that the TM performance for *Person* object is comparable with the original YOLOv3 mAP result for *Person* object.

Trained Model	*COCO [14] (Person)*	*OID [15] (Person)*	*OID [15] (Flag)*	*Total Training Images*
TM-01	-	991	968	1999
TM-02	1663	-	968	2631
TM-03	1663	991	968	3622
TM-04	3000	3000	6000	12000

Table I: Training Dataset Used

Trained Model	*AP Person (%)*	*AP Flag (%)*	*mAP Value (%)*
TM-01	90.45	71.69	81.07
TM-02	94.91	*79.41*	*87.16*
TM-03	90.27	72.92	81.94
TM-04	*97.92*	75.53	86.72
YOLOv3 [11]	*98.43*	N/A	N/A

Table II: Trained Model Results

C. Latency Analysis

To evaluate the detection latency performed by our proposed EagleEYE system, we do experiments by scaling up YOLOv3 workers. This evaluation aims to analyze: 1) the impact of having the objection detection scaled up, and 2) the resource allocation due to satisfying the increasing number of multiple workers. To make a fair comparison, we calculate total recognition latency for every 6 frames, called $batch$. The recognition latency consists of the following measurements:

1) *Video to frames Extraction*. The latency to extract a HTTP Stream into frames where each frame will be sent to each YOLOv3 worker.
2) *YOLOv3 Inference*. The latency to perform object detection and classification of Person objects and Flag objects.
3) *MOD Algorithm*. The latency of the proposed algorithm to generate pairs of *PiH* object.

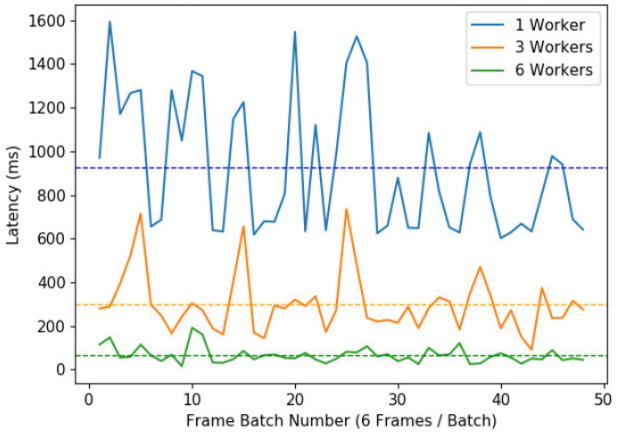

Figure 4: EagleEYE detection latency

Figure 4 depicts the variation tendency of the recognition latency captured different numbers of workers. The result shows that by having the number of workers scaled up and process frames in parallel, we manage to reduce the recognition latency up to 90%. As for the resource utilization, our experiments show that each worker consumes 4.7% GPU Memory, which is around $1.515\ GB$ out of $32\ GB$.

V. Conclusion

In this paper, we propose an aerial edge-enabled disaster relief response system called EagleEYE. We make three contributions to achieve real-time capable object detection for disaster relief. First, we developed an object fusion mechanism, called Merged Object Detection, to reduce training dataset preparation effort by reusing existing datasets for new application. Second, we parallelized the computation of the detection tasks to reduce the inference latency. Finally, we evaluated EagleEYE in a real testbed. Our experimental results show that EagleEYE can not only reduce the inference latency by 90% but also has high detection accuracy of 87%.

VI. Acknowledgement

This work has been partially funded by the H2020 EU/TW joint action 5G-DIVE (Grant #859881).

References

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.
[2] P. Petrides, P. Kolios, C. Kyrkou, T. Theocharides, and C. Panayiotou, "Disaster prevention and emergency response using unmanned aerial systems," in *Smart Cities in the Mediterranean*. Springer, 2017, pp. 379–403.
[3] S. Lee, D. Har, and D. Kum, "Drone-assisted disaster management: Finding victims via infrared camera and lidar sensor fusion," in *2016 3rd Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*. IEEE, 2016, pp. 84–89.
[4] T. J. Tanzi, M. Chandra, J. Isnard, D. Câmara, O. Sébastien, and F. Harivelo, "Towards "drone-borne" disaster management: future application scenarios," 2016.
[5] C. Alex and A. Vijaychandra, "Autonomous cloud based drone system for disaster response and mitigation," in *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*. IEEE, 2016, pp. 1–4.
[6] A. Gaszczak, T. P. Breckon, and J. Han, "Real-time people and vehicle detection from uav imagery," in *Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*, vol. 7878. International Society for Optics and Photonics, 2011, p. 78780B.
[7] M. B. Bejiga, A. Zeggada, and F. Melgani, "Convolutional neural networks for near real-time object detection from uav imagery in avalanche search and rescue operations," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2016, pp. 693–696.
[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.
[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.
[11] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
[12] O. Ibrahiem, L.-C. Wang, S. B. Chundrigar, and K.-L. Huang, "Enabling mobile service continuity across orchestrated edge networks," *IEEE Transactions on Network Science and Engineering*, 2019.
[13] C. Guimarães, A. de la Oliva, and A. Azcorra, "5G-DIVE: edge intelligence for vertical experimentation," 2019.
[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.
[15] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv preprint arXiv:1811.00982*, 2018.
[16] J. Redmon, "Darknet: Open source neural networks in c," 2013.
[17] M. Yuang, P.-L. Tien, W.-Z. Ruan, T.-C. Lin, S.-C. Wen, P.-J. Tseng, C.-C. Lin, C.-N. Chen, C.-T. Chen, Y.-A. Luo *et al.*, "Optuns: Optical intra-data center network architecture and prototype testbed for a 5g edge cloud," *Journal of Optical Communications and Networking*, vol. 12, no. 1, pp. A28–A37, 2020.
[18] F. team, "Ffmpeg," 2019. [Online]. Available: https://ffmpeg.org/
[19] I. GitHub, "Open source survey," https://github.com/AlexeyAB/darknet, 2020.
[20] G. Jocher, guigarfr, perry0418, Ttayu, J. Veitch-Michaelis, G. Bianconi, F. Baltacı, D. Suess, WannaSeaU, and IlyaOvodov, "ultralytics/yolov3: Rectangular Inference, Conv2d + Batchnorm2d Layer Fusion," Apr. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2672652
[21] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.