2022 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW)

Edge Computing versus Cloud Computing: Impact on Retinal Image Pre-processing

Cristiano A. Künas[1], Dayla R. Pinto[1], Lisandro Z. Granville[1], Matheus S. Serpa[1],
Edson L. Padoin[2], Philippe O. A. Navaux[1]
[1]Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS) — Porto Alegre, Brazil
[2]Regional University of Northwestern Rio Grande do Sul (UNIJUÍ) — Ijuí, Brazil

{cakunas, dayla.rpinto, granville, msserpa, navaux}@inf.ufrgs.br, padoin@unijui.edu.br

***Abstract*—With the emergence of new intelligent applications, there has been a revolution in information management, mainly including processing routines, storage, and computing capacity. Cloud computing is one of the most well-known and popular paradigms. However, there are still some barriers to cloud adoption. Edge computing serves as a decentralized extension and provides solutions that facilitate data processing at the generation source. This paper discusses the development of a retinal image pre-processing application for efficient use in screening systems and the impact that pre-processing has on network interconnection. Results show that the parallel version has reduced application execution time by up to $\approx$ 73%, decreasing by $\approx 11.5\times$ the bandwidth used, achieving throughput above 5 images/second with Edge pre-processing, $2.57\times$ higher than Cloud.**

***Index Terms*—high-performance computing; edge; cloud; pre-processing; diabetic retinopathy;**

I. Introduction

With the emergence of new intelligent applications, there has been a revolution in information management, mainly including processing routines, storage, and computing capacity. We have seen a significant evolution of computing paradigms in the last decade. Cloud computing is one of the most well-known and popular paradigms, and its adoption continues to accelerate as companies seek agility, flexibility, and new sources of competitive advantage. According to Gartner[1], about 40% of enterprise workloads will be deployed on cloud infrastructure and platform services by 2023, 20% higher compared to 2020. Avoiding it is almost inconceivable, but there are still some barriers to cloud adoption and the overwhelming spread of smart devices and appliances, as the Internet of Things (IoT) pointed out all the limitations of such a centralized paradigm.

Edge computing serves as a decentralized extension and provides solutions that facilitate data processing at the source of generation or closer to the data creators and users, solving centralization-induced problems such as optimizing latency requirements, bandwidth savings, and autonomy. For inference systems, for example, edge devices capture data and send it to the Cloud for processing, but transferring data from the edge device to the Cloud can take a long time if the volume of data is very large. Therefore, it makes sense to process the captured data locally on the edge node to minimize the transfer time over the network.

Diabetic Retinopathy (DR) is a progressive disease and primary retinal vascular complication of diabetes mellitus. In 2019, 463 million people had diabetes worldwide. This number can reach 693 million by 2045. The global prevalence of diabetes in 2019 is estimated at 9.3% (463 million people), increasing to 10.2% (578 million) in 2030 and 10.9% (700 million) in 2045 [1]. In 2020, a global meta-analysis of 12,620 patients with diabetes showed that the prevalence of DR reached 35.36%, and the vision of 11.72% of patients was seriously affected [2].

Complications of DR are often preventable if detected and treated early. In practice, the clinical features of these complications are evident in the fundus of the eye on [3] ophthalmic examinations. Technological advances have enabled the development of algorithms for automated detection of DR and diabetic macular edema in retinal fundus images. Capturing these images often uses various hardware devices under different environmental conditions, inducing noise in the final image. To reduce this heterogeneity, which affects the diagnosis's performance, as well as highlight some details of the images, the pre-processing of the images becomes a necessary step. From a clinical point of view, about 20% of retinal images are not used due to poor image quality. Therefore, before DR screening, the selection of an effective pre-processing scheme is mandatory [4].

This article aims to show that data pre-processing can be performed at the edge node and accelerated through parallelism techniques. A simple method of pre-processing retinal images (locating the center and resizing) was evaluated and verified its performance in two environments: Cloud and Edge.

The rest of this paper is organized as follows. Section II provides an overview of network analysis, DR, and retinal image pre-processing tools. Section III discusses some related works. We described the implementation methodology and architectures in Section IV. Section V presents the evaluation results covering execution performance and image throughput in sequential and parallel versions. Finally, in Section VI, conclusions and future work are presented.

[1]https://www.gartner.com/

978-1-6654-5157-4/22/$31.00 ©2022 IEEE
DOI 10.1109/SBAC-PADW56527.2022.00018

II. Background

This Section presents concepts about network traffic analysis tools, Diabetic Retinopathy (DR), and techniques for pre-processing retinal images.

A. Network traffic analysis

Network traffic analysis aims to capture valuable information and material relevant to network management. In general, network traffic analysis can be defined as the inference of information from observing data flow in network traffic. Network analysis is categorized by time (or frequency), criteria and analysis objectives so that it can be classified into real-time analysis, batch analysis, or forensic analysis [5].

One popular tool for capturing network traffic is *sniffer*, a computer program that watches incoming and outgoing traffic on network devices. The *sniffer* captures the packets addressed to other machines and saves them for later analysis (or, more rarely, inline). *Sniffers* are legitimately used by network or system administrators to monitor and support network troubleshooting [6]. Although there are hundreds of *sniffers* on the market, TCPdump, and Wireshark are the most used and appreciated tools [7].

TCPdump is a command-line tool for network monitoring, packet capture, and protocol debugging. TPCdump is free and open source, running only on Linux-based systems [8]. Typically, TCPdump is used to capture data packets transferred over the network in the following cases: designing networks/protocols, checking that network services are working correctly, monitoring and making statistics based on traffic [8]. For this purpose, TCPdump provides many options where the details of the captured packets can be explored and viewed in various formats.

Wireshark, meanwhile, is an open-source network protocol analyzer used to collect, troubleshoot, and assist network administrators by allowing data from a network to be tracked in real-time and stored for later analysis [9]. Wireshark is credited for its simple graphical interface, powerful capture and filtering options, and support for Linux and Windows platforms. It is also capable of scanning networks *Ethernet*, Wi-Fi or even *Bluetooth* [7].

B. Diabetic Retinopathy

Diabets Mellitus is a metabolic disease characterized by an abnormal blood sugar or glucose increase. When not appropriately treated, the patient will be subject to complications such as heart attack, stroke, kidney failure, hard-to-heal injuries, and vision problems [10]. Vision problems occur because diabetes affects the circulatory system, including progressive vascular ruptures caused by chronic hyperglycemia, and can develop regardless of the severity of the patient, causing what is called diabetic retinopathy (DR) [11].

DR is a complication arising from the prolonged diabetic condition, appearing after ten to fifteen years [12]. Hyperglycemia causes damage to the tiny blood vessels within the retina [13]. These blood vessels leak blood and fluid into the retina, causing problems such as microaneurysms, hemorrhages, intraretinal microvascular abnormalities, and exudates [14]. Figure 1 illustrates retinas with DR.

Fig. 1. Retinas with Diabetic Retinopathy.

DR can be clinically divided into two stages: proliferative diabetic retinopathy (PDR) and non-proliferative diabetic retinopathy (NPDR) [15]. PDR is a more advanced form of DR marked by the proliferation of fibrovascular tissue leading to vitreous hemorrhage and retinal detachment [16]. It is also characterized by the proliferation of retinal vessels, the growth of which is variable. They are commonly identified according to their location in the retina, optic disc, or nearby, and are more likely to proliferate on the posterior surface of the vitreous and hemorrhage in the vitreous. Also, over time, new vessels can often contract, resulting in retinal detachment [17]. NPDR is an early stage of DR characterized by retinal hemorrhages, and microaneurysms [16].

Regarding clinical classification protocols for PDR and NPDR, the *Global Diabetic Retinopathy Project Group* [18] proposed a five-level disease severity scale: I) **No apparent retinopathy**: no abnormalities; II) **Mild non-proliferative diabetic retinopathy**: the presence of retinal microaneurysms; III) **Moderate non-proliferative diabetic retinopathy**: more than just microaneurysms, but less than severe non-proliferative diabetic retinopathy; IV) **Serious non-proliferative diabetic retinopathy**: more than 20 intra-retinal hemorrhages in each of the four quadrants, venous pearling in at least two quadrants, and intra-retinal microvascular abnormalities in at least one quadrant in the absence of PDR; and V) **Proliferative diabetic retinopathy**: neovascularization, vitreous/pre-retinal hemorrhage.

To tracking for the severity level of DR, regular eye examinations are recommended for people with *diabetes mellitus*, as timely diagnosis and subsequent management of the condition are essential to establish early treatment [19], as DR can develop and progress to advanced stages without producing any immediate symptoms to the patient, establishing risks, such as vision loss [15].

C. Pre-processing of retinal images

The image pre-processing step is done to produce an image with better quality. Most retinal images are not uniformly illuminated and sometimes have low visual contrast and noise,

making it difficult to detect lesions [20]. This occurs because of capture device angle error, eye movement, and others, and makes the fundus image inadequate for disease diagnosis. Thus, selecting the appropriate pre-processing method for the retinal fundus image is a required step [4].

One of the main pre-processing steps is the removal of noise, which is categorized as: salt and pepper noise (black and white pixel randomness occurs); Gaussian noise (intensity value variation with normal gaussian distribution occurs); and flicker noise (contains random white pixels) [21]. Other important steps are contrast enhancement, tone correction, and resizing [4]

Filters used in image pre-processing can be done by linear and non-linear methods. In the linear method, the algorithm applies the filter linearly to all pixels without defining which image is corrupted or uncorrupted. The non-linear method algorithm, on the other hand, applies the filter by defining which pixel is corrupted or uncorrupted. The non-linear filter produces better results compared to the linear filter [21].

III. Related Work

Voets et al. [22] replicates the scientific paper by Gulshan et al. [23], in which the source code was not published, only described. To reimplement the original algorithm for RD detection, the authors used similar images from two datasets: EyePACS and Messidor-2. They evaluated the gradability of the images on their own; images considered gradable were preprocessed as described by the original study's protocol for preprocessing and used to train the neural network.

Fatemeh et al [24] propose a new platform called DEFT (Dynamic Edge-Fabric environmenT) that learns where is the best to execute each task based on real-time system status and task requirements. It takes into account behavior from past performance of the available resources. Silva et al [25] propose a methodology to evaluate performance trade-offs through experimental evaluation using two real-life stream processing use-cases executed on fully-Cloud and hybrid Cloud-Edge testbeds. The results were compared with state-of-the-art processing engines for each environment. Pereira [26] increasing the efficiency of fog nodes the Edge Computing through of a priority-based load balancing. Vinay [27] focuses on computational resources of Edge Computing. The work is located in mobile antenna base stations, which offer their computational resources to mobile devices through the adoption of SDN. There was a decrease of more than 50% in response times with the proposed work.

IV. Methodology

The infrastructure of the application testing architecture can be described as a composition of three layers, as depicted in Figure 2: (1) Cloud layer, which runs the DR detection model trained for inference; (2) Storage layer, consisting of a bucket in Google Cloud Storage that is used to store the data sent to the Cloud; and (3) Edge layer, consisting of local machine that performs pre-processing and sending tasks to the Cloud.

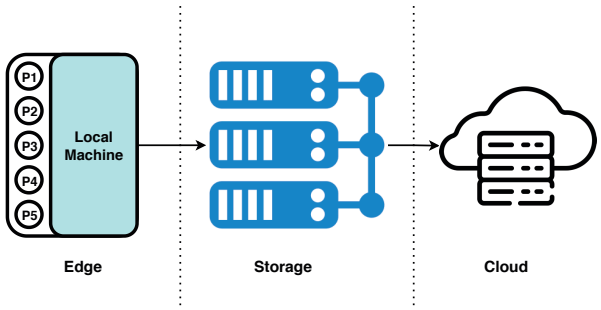


Fig. 2. The architecture comprises three layers: Cloud, Storage, and Edge.

The application is implemented in Python and executed in Shell Script. The application is divided into two modules. The first module performs all pre-processing of the data at the edge layer and then sends it to the storage layer in the Cloud. Differently, the second module does the opposite, first sending the original data – without any pre-processing – to the storage layer, and then the Cloud layer takes responsibility for pre-processing the data. In both modules, we evaluate the pre-processing and the sending of the data sequentially and in parallel through processes using Python's `ProcessPoolExecutor()` class. In the parallel application, each process receives a portion of the data and performs specific functions. For example, in the first module, the main function loads an image from local storage, pre-processes it (as suggested by the original study [23]) by locating the center and radius of the bottom of the eye and resizing it to a height and width of 299 pixels, and then does `upload` to the `bucket` on Google Storage.

The dataset used in the experiments is the *APTOS 2019 Blindness Detection* available in the Kaggle competition for diabetic retinopathy[2], which was chosen because it has a reasonable size compared to the others mentioned above, facilitating our experiments. This database is commonly used for machine learning applications in DR detection and is divided into two subsets. In this work, we selected the training subset containing 3,662 retinal images. The images are in PNG (Portable Network Graphics) format and have varying sizes (Figure 5).

The cloud environment consists of a virtual machine type N1 Standard (16 cores, 60 GB RAM), Linux Ubuntu 20.04 LTS operating system, located in eastern South America. The storage environment uses Google Cloud Storage provided through an instance of Firebase. This storage is an object storage solution with large capacity, high availability, and redundancy. Cloud Storage for Firebase allows you to manipulate files securely and easily via its SDK.

The execution environment on Edge comprises a device with an Intel Core i7-9750 processor with 6 physical cores (2.60 GHz). This equipment has 16 GB of DDR4 RAM

[2]https://www.kaggle.com/c/aptos2019-blindness-detection/data

Memory. We used the operating system Linux Ubuntu 20.04.4 LTS with *kernel* version 5.13.0-41.

V. Results

In this Section, we present the performance evaluation achieved by our application on the experimental platform shown in the previous Section. We present metrics for execution time and image throughput in sequential and parallel versions. The results presented in this Section are the average of at least 10 runs. The relative error was less than 5% using statistical confidence of 95% by distribution t-Student.

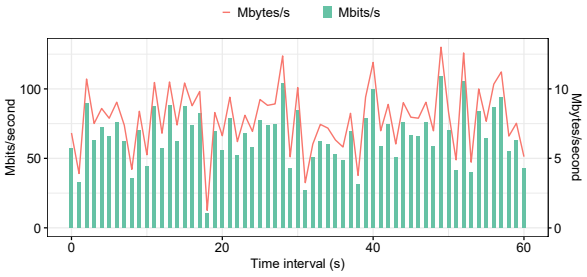
A. Network evaluation

To evaluate the implementation, we first evaluated the network connection. This evaluation measured the maximum amount of data that could be sent from the edge node to the cloud provider. The measurements started with throughput analysis using the Iperf tool [28]. The results obtained from this first step are described in Table I.

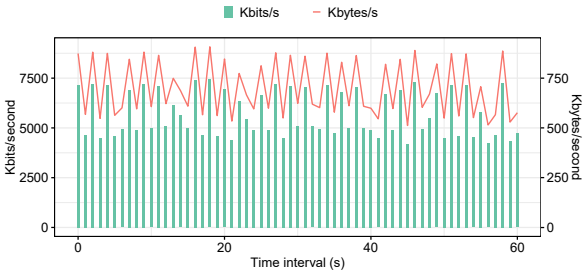
TABLE I
Network measurements with the Iperf tool.

Parameter	Edge to Cloud
Interval	60 seconds
Total transferred	406 MBytes
Bandwidth	56.8 Mbits/second

Figure 3 shows the bandwidth and the amount of bytes transferred every second. Comparing with the measurement performed with the `ipref3` tool, it is noticeable that when sending the original data, we used all the available bandwidth, with an average of $\approx$ 66 Mbps (Figure 3(a)), which can overload the network. The opposite occurs when sending pre-processed data, in which we reduce the bandwidth used by about $\approx 11.5\times$, with an average of ≈ 5.7 Mbps (Figure 3(b)). We also observed a variation in the minimum and maximum transfer peaks for the original data (Figure 3(a)). This is due to size differences between the files.



(a) Without pre-processing.



(b) With pre-processing.

Fig. 3. Bandwidth used and total transferred.

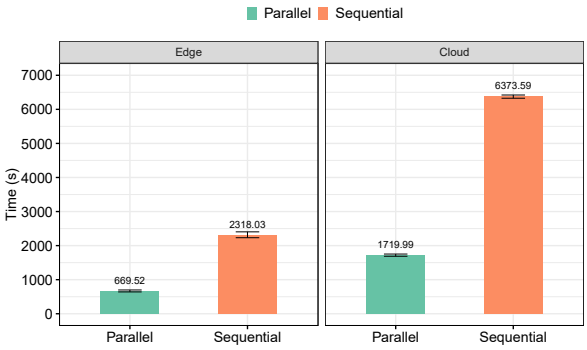


Fig. 4. Sequential and parallel version execution times: comparing execution on the Edge and in the Cloud.

B. Performance Evaluation

The application has two modules: one that performs the pre-processing on the Edge and sends it to the storage in the Cloud, and another that sends the original data to the storage and processes it in the Cloud. Therefore, we compare the **sequential execution** time between the Edge and Cloud environments, and we also compare the **parallel execution** time between the Edge and Cloud environments. As explained in the methodology, the parallel version splits tasks between processes. For both executions, we set 5 processes.

Parallel executions on the Edge and the Cloud show a gain of $\approx 71.12\%$ and $\approx 73.01\%$, respectively, compared to the sequential version. In comparison, the average execution time of the sequential application on the Edge was $\approx 2,318.03$s, a gain of $\approx 63.63\%$ compared to the same version running on the Cloud. In the parallel version, the average execution time on the Edge was ≈ 669.52s, which represents a gain of $\approx 61.07\%$ compared to the same version running on the Cloud. As we can see, executions on the Edge perform better than on the Cloud because they do not overload the network. By pre-processing the data on the Edge, in addition to decreasing the total size of the data by $\approx 44\times$ (decreasing from $\approx 8,204.5$ MB to ≈ 185.7 MB), we also save bandwidth for transferring this data.

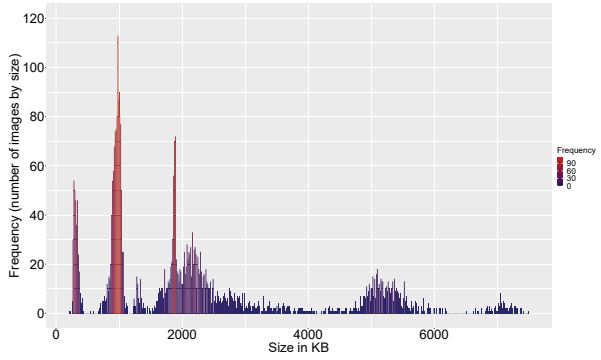
We present the application's throughput in Table II. This throughput rate represents the number of images pre-processed and transferred every second. It can be seen that in the cloud sequential execution model, where the first step is sending the original data to the Cloud, the throughput is too low. This is because the dataset has a large variation in the sizes of the

original images. In Figure 5(a) we demonstrate this variation, with sizes ranging from $\approx$ 200 KB to $\approx$ 7,500 KB. This disparity in values significantly impacts execution time and bandwidth consumption.

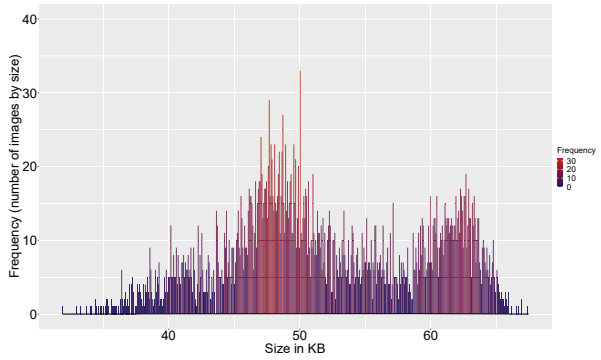
TABLE II
THROUGHPUT IN IMAGES/SECOND, IN SEQUENTIAL AND PARALLEL RUNS, IN EDGE AND CLOUD ENVIRONMENTS.

	Edge	**Cloud**
Parallel	5.47 images/second	2.13 images/second
Sequential	1.58 images/second	0.57 images/second

The best result in terms of throughput is obtained in the parallel version executed on the Edge, reaching $\approx$ 5 images/second. This is because the pre-processing step allows us to normalize the data, reducing the size of each image and the size variation observed in the original data (30 - 70 KB), as shown in Figure 5(b).



(a) Without pre-processing.



(b) With pre-processing.

Fig. 5. Frequency histogram of images by size in KB without and with pre-processing.

VI. CONCLUSION AND FUTURE WORK

This paper presents an application for pre-processing retinal images for efficient use in tracking systems and the impact that pre-processing causes on the network. We evaluate a simple method for pre-processing retinal images and verify its performance in sequential and parallel versions running in Cloud and Edge environments.

With pre-processing on the Edge, we reduced $\approx 72\%$ the execution time and $\approx 11.7\times$ the bandwidth used, achieving throughput over 5 images/second in the best case, $2.52\times$ higher than in the Cloud. This is due to the reduction in total data size, $\approx 44\times$ smaller than the original data.

Our experiments have shown that the data size can negatively influence the application's execution time. In this context, parallelism strategies become essential. Network interconnection can also be challenging since the original data often have large sizes, which congests the connection. Thus, pre-processing data at the Edge can contribute to bandwidth savings.

Future work will extend the performance evaluation with more sophisticated pre-processing techniques. In addition, we plan to expand the study by evaluating larger and higher-resolution datasets.

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. This work has been partially supported by Green Cloud project (2016/2551-0000 488-9), from FAPERGS and CNPq Brazil, program PRONEX 12/2014, by CNPq/MCTI/FNDCT - Universal 18/2021 under grants 406182/2021-3, MCTIC/CNPq - Universal 28/2018 under grants 436339/2018-8 and by CIARS RITEs/FAPERGS project.

REFERENCES

[1] X.-N. Wang, L. Dai, S.-T. Li, H.-Y. Kong, B. Sheng, and Q. Wu, "Automatic grading system for diabetic retinopathy diagnosis using deep learning artificial intelligence software," *Current Eye Research*, vol. 45, no. 12, pp. 1550–1555, 2020.

[2] P. Saeedi, I. Petersohn, P. Salpea, B. Malanda, S. Karuranga, N. Unwin, S. Colagiuri, L. Guariguata, A. A. Motala, K. Ogurtsova *et al.*, "Global and regional diabetes prevalence estimates for 2019 and projections for 2030 and 2045: Results from the international diabetes federation diabetes atlas," *Diabetes research and clinical practice*, vol. 157, p. 107843, 2019.

[3] J. Lechner, O. E. O'Leary, and A. W. Stitt, "The pathology associated with diabetic retinopathy," *Vision research*, vol. 139, pp. 7–14, 2017.

[4] A. Chatterjee, N. S. Datta, H. S. Dutta, K. Majumder, and S. Chatterjee, "A study on retinal image preprocessing methods for the automated diabetic retinopathy screening operation," *Applications of Artificial Intelligence and Machine Learning*, pp. 375–384, 2021.

[5] P. Asrodia and H. Patel, "Network traffic analysis using packet sniffer," *International journal of engineering research and applications*, vol. 2, no. 3, pp. 854–856, 2012.

[6] L. Ying-hua, Y. Bing-Ru, C. Dan-yang, and M. Nan, "State-of-the-art in distributed privacy preserving data mining," in *2011 IEEE 3rd International Conference on Communication Software and Networks*. IEEE, 2011, pp. 545–549.

[7] P. Goyal and A. Goyal, "Comparative study of two most popular packet sniffing tools-tcpdump and wireshark," in *2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 2017, pp. 77–81.

[8] T. Solomon, A. M. Zungeru, and R. Selvaraj, "Network traffic monitoring in an industrial environment," in *2016 Third International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA)*. IEEE, 2016, pp. 133–139.

[9] F. Luo, L. Dong, and F. Jia, "Method and implementation of building forces protocol dissector based on wireshark," in *2010 2nd IEEE International Conference on Information Management and Engineering*. IEEE, 2010, pp. 291–294.

[10] R. Pires and A. Rocha, "Combinação de classificadores para um sistema automático de triagem de retinopatia diabética," in *Proceedings of the SPS 2011*, 2011. [Online]. Available: https://www.sps.fee.unicamp.br/sps2011/proceedings_sps2011/Ramon_Diabetica_SPS2011.pdf

[11] M. Janghorbani, R. B. Jones, and S. P. Allison, "Incidence of and risk factors for proliferative retinopathy and its association with blindness among diabetes clinic attenders," *Ophthalmic Epidemiology*, vol. 7, no. 4, pp. 225–241, 2000.

[12] S. J. Lee, C. A. McCarty, H. R. Taylor, and J. E. Keeffe, "Costs of mobile screening for diabetic retinopathy: a practical framework for rural populations," *Australian Journal of Rural Health*, vol. 9, no. 4, pp. 186–192, 2001.

[13] T. Vandarkuzhali, C. Ravichandran, and D. Preethi, "Detection of exudates caused by diabetic retinopathy in fundus retinal image using fuzzy k means and neural network," *IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE) e-ISSN*, pp. 2278–1676, 2013.

[14] F. Alzami, R. A. Megantara, A. Z. Fanani *et al.*, "Diabetic retinopathy grade classification based on fractal analysis and random forest," in *2019 International Seminar on Application for Technology of Information and Communication (iSemantic)*. IEEE, 2019, pp. 272–276.

[15] A. W. Stitt, T. M. Curtis, M. Chen, R. J. Medina, G. J. McKay, A. Jenkins, T. A. Gardiner, T. J. Lyons, H.-P. Hammes, R. Simo *et al.*, "The progress in understanding and treatment of diabetic retinopathy," *Progress in retinal and eye research*, vol. 51, pp. 156–186, 2016.

[16] T. Y. Wong, R. Klein, F. A. Islam, M. F. Cotch, A. R. Folsom, B. E. Klein, A. R. Sharrett, S. Shea, M.-E. S. of Atherosclerosis (MESA *et al.*, "Diabetic retinopathy in a multi-ethnic cohort in the united states," *American journal of ophthalmology*, vol. 141, no. 3, pp. 446–455, 2006.

[17] N. K. Waheed, "Proliferative diabetic retinopathy," in *Atlas of Retinal OCT: Optical Coherence Tomography*. Elsevier, 2018, pp. 88–89.

[18] C. Wilkinson, F. L. Ferris III, R. E. Klein, P. P. Lee, C. D. Agardh, M. Davis, D. Dills, A. Kampik, R. Pararajasegaram, J. T. Verdaguer *et al.*, "Proposed international clinical diabetic retinopathy and diabetic macular edema disease severity scales," *Ophthalmology*, vol. 110, no. 9, pp. 1677–1682, 2003.

[19] S. I. G. Network, "Management of obesity: a national clinical guideline," *Scottish Intercollegiate Guidelines Network: Edinburgh*, vol. 20, 2010.

[20] H. F. Jaafar, A. K. Nandi, and W. Al-Nuaimy, "Detection of exudates from digital fundus images using a region-based segmentation technique," in *2011 19th European signal processing conference*. IEEE, 2011, pp. 1020–1024.

[21] C. Swathi, B. Anoop, D. A. S. Dhas, and S. P. Sanker, "Comparison of different image preprocessing methods used for retinal fundus images," in *2017 Conference on Emerging Devices and Smart Systems (ICEDSS)*. IEEE, 2017, pp. 175–179.

[22] M. Voets, K. Møllersen, and L. A. Bongo, "Reproduction study using public data of: Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *PloS one*, vol. 14, no. 6, p. e0217541, 2019.

[23] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *Jama*, vol. 316, no. 22, pp. 2402–2410, 2016.

[24] F. Jalali, T. Lynar, O. J. Smith, R. R. Kolluri, C. V. Hardgrove, N. Waywood, and F. Suits, "Dynamic edge fabric environment: Seamless and automatic switching among resources at the edge of iot network and cloud," in *2019 IEEE International Conference on Edge Computing (EDGE)*, 2019, pp. 77–86.

[25] P. Silva, A. Costan, and G. Antoniu, "Investigating edge vs. cloud computing trade-offs for stream processing," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 469–474.

[26] E. P. Pereira, E. L. Padoin, R. D. Medina, and J.-F. Méhaut, "Increasing the efficiency of fog nodes through of priority-based load balancing," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–6.

[27] V. Chamola, C.-K. Tham, and G. S. Chalapathi, "Latency aware mobile task assignment and load balancing for edge cloudlets," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2017, pp. 587–592. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7917628

[28] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: the tcp/udp bandwidth measurement tool (2005)," *URL: http://iperf. sourceforge. net*, 2005.