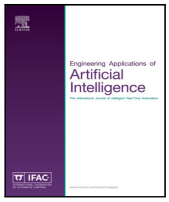
Engineering Applications of Artificial Intelligence 104 (2021) 104384

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



A study on the use of Edge TPUs for eye fundus image segmentation

Javier Civit-Masot [a], Francisco Luna-Perejón [a], José María Rodríguez Corral [c,*], Manuel Domínguez-Morales [a,b], Arturo Morgado-Estévez [c], Antón Civit [a,b]

[a] Architecture and Computer Technology Dept., E.T.S. Ingeniería Informática, Avda. Reina Mercedes, s/n, University of Seville, Seville, Spain
[b] Computer Engineering Research Institute (I3US), University of Seville, Seville, Spain
[c] School of Engineering, Avda. Universidad de Cádiz, 10, University of Cádiz, Puerto Real (Cádiz), Spain

ARTICLE INFO

Keywords:
Deep Learning
Edge TPU
Medical image segmentation
Glaucoma
Single-board computer
U-Net

ABSTRACT

Medical image segmentation can be implemented using Deep Learning methods with fast and efficient segmentation networks. Single-board computers (SBCs) are difficult to use to train deep networks due to their memory and processing limitations. Specific hardware such as Google's Edge TPU makes them suitable for real time predictions using complex pre-trained networks. In this work, we study the performance of two SBCs, with and without hardware acceleration for fundus image segmentation, though the conclusions of this study can be applied to the segmentation by deep neural networks of other types of medical images. To test the benefits of hardware acceleration, we use networks and datasets from a previous published work and generalize them by testing with a dataset with ultrasound thyroid images. We measure prediction times in both SBCs and compare them with a cloud based TPU system. The results show the feasibility of Machine Learning accelerated SBCs for optic disc and cup segmentation obtaining times below 25 ms per image using Edge TPUs.

1. Introduction

In recent years, the use of Deep Learning technologies for medical image analysis has quickly increased (Litjens et al., 2017; Chen et al., 2020; Teikari et al., 2019; Akkara et al., 2019). One of the main applications has been image segmentation, which is the process of detecting automatically or semi-automatically the limits within a two or three-dimensional image.

In medical segmentation problems, many different segmentation networks have been used (Litjens et al., 2017); however, a type of fully convolutional neural network (CNN), known as U-Net (Ronneberger et al., 2015), has become very widely used and shown to be very effective. U-Nets have been used with many types of medical images including X-ray, MRI, CT, Ultrasound and eye Fundus images. The structure of a small three-layer generalized U-Net can be seen in Fig. 1. The network is made up of a set of descending layers, each with a larger number of filters but with the image resolution reduced to a quarter, an intermediate connecting layer (the bottom of the "U") and a set of ascending layers on which the original resolution is recovered.

Eye fundus images are widely used to help in Glaucoma detection. Glaucoma is a retinal decease that can cause blindness in about 2% and sight impairment in over 10% of the cases (Quigley, 1985). Even though vision loss may occur even with optimum treatments, adequate therapy will stabilize the majority of cases.

The key to glaucoma detection is to understand how to examine the optic disc (OD) (Bourne, 2006). The OD is an oval area where the retina connects to the optic nerve. The optic cup (OC) is a white cup-like area in the center of the OD. The zone between the cup and the disc is known as the neuroretinal rim. This region consists mostly of nerve fibers and is usually pink. Most normal discs are mainly vertically oval with their cup horizontally oval. A typical retina fundus image is shown in Fig. 2.

Different indicators are used to help in diagnosing glaucoma from fundus images. The cup to disc ratio (CDR) (MacIver et al., 2017), i.e. the relation between the diameters of the OD and the OC, is the most accepted glaucoma predictor. CDRs for glaucomatous and healthy eyes are about 0.65 ± 0.13 and 0.39 ± 0.15 respectively, thus establishing CDR as a valid diagnostic aid. An alternative detection method is based on the ISTN rule which uses the shape of the neuroretinal rim. On healthy eyes, the thickness of the rim along the vertical and horizontal rim borders decreases in the order inferior (I)>superior (S)>nasal (N)>temporal (T) (Das et al., 2016).

In our previous works (Civit-Masot et al., 2019, 2020), generalized U-Nets were used for eye fundus image segmentation, specifically for optic disc (OD) and optic cup (OC) detection. The U-Net models were implemented on cloud-based GPU and TPU (Google, 2020) architectures.

An accurate OC and OD segmentation is important in order to calculate the CDR that, as already mentioned, is a well-established indicator for the diagnosis of glaucoma (Jonas and Bron, 2015; Patel and Patel, 2018; Barros et al., 2020; Cheng et al., 2013). The ISTN

* Corresponding author.
E-mail address: josemaria.rodriguez@uca.es (J.M.R. Corral).

https://doi.org/10.1016/j.engappai.2021.104384
Received 29 September 2020; Received in revised form 3 June 2021; Accepted 6 July 2021
Available online 27 July 2021

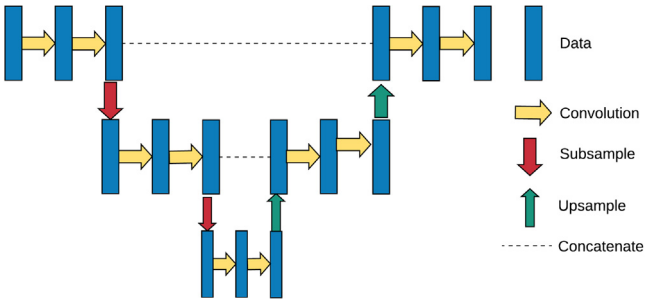


Fig. 1. Basic three layer U-Net.

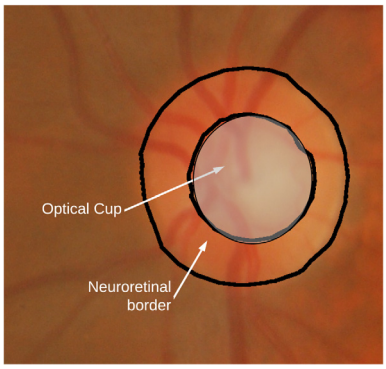


Fig. 2. Optic Disc and Cup.

rule (Nath and Dandapat, 2012) also relies on adequate OC and OD segmentation.

Currently, single-board computers (SBCs) have become popular as small low-cost computers for development, IoT and educational applications (Hassan et al., 2017; Singh and Kapoor, 2017). The purpose of this work is to evaluate SBCs, with and without specific Machine Learning acceleration hardware, for implementing the generalized U-Net models developed in Civit-Masot et al. (2020) for performing OD and OC segmentation.

In this work, we use very similar generalized U-Net architectures to segment OD and OC from fundus images and train them on Google TPUs. In particular, we use for cup segmentation a 6 level network with 64 channels in the first stage and a layer to layer filter increment ratio (IR) of 1.1. This is a lightweight model (2.5M trainable parameters), but as shown in Civit-Masot et al. (2019) it produces good results for cup segmentation. Although the model has an additional level compared to the original U-Net, and both have 64 channels in the first layer by decreasing the IR to 1.1 instead of the original 2.0, the number of parameters is reduced more than 50 times. For disc segmentation, as this is an easier problem, we use a similar network with only 40 channels in the first stage and less than 1M trainable parameters.

Even though we are using specific U-net examples, our aim is to show that embedded systems with specific Deep Learning acceleration hardware can perform many medical image segmentation problems in very reasonable times. This problem is very interesting in real medical practice as these hardware accelerated embedded processors can be easily included in lightweight portable medical instruments that can perform segmentation on their own without requiring an external PC.

In this sense, segmentation by deep neural networks can also be performed in many other types of medical images (Litjens et al., 2017; Chen et al., 2020), thus allowing this technology to be embedded in a wide range of portable medical diagnosis instruments.

Hence, the conclusions of this work can be applied, for example, to the segmentation performed by Machine Learning accelerated embedded systems for organs and other substructures in cardiac or brain analysis, or for multi-organ segmentation (widely used for abdominal organ segmentation) (Hesamian et al., 2019).

In this work, more specifically, we implement the segmentation subsystem in Civit-Masot et al. (2020) in two embedded devices. The first is a Raspberry Pi with no specific Machine Learning hardware and the second a Coral Dev Board with a Google's Edge TPU. Due to the limited resources of SBCs, even with specific accelerators, in terms of processing power and main memory size, it is technically unfeasible to train relatively large CNNs using them. However, they can be used for prediction purposes, as prediction is computationally a much less demanding process than training.

The Raspberry Pi has become very popular because of its affordable price, its ease of use due to the availability of the Raspbian operating system (a derivative of Debian Linux), and its large development community. Moreover, the new Raspberry Pi 4 includes a more powerful processor and up to 4 GB of RAM. Thus we use this very well known device as one of our reference systems to be able to compare its results with devices that implement hardware acceleration for Machine Learning.

The Coral Dev Board is a SBC specifically designed to perform Machine Learning inferencing in a small-form factor. It includes a simplified tensor processing unit (TPU), the Edge TPU, which is an ASIC developed by Google for providing high performance Machine Learning inferencing with a low power usage. A very similar SBC, also with Edge TPU, is the Tinker Edge T board from Asus.

The purpose of this work consists not only of confirming the feasibility of implementing the U-Nets used in Civit-Masot et al. (2020) to perform OD and OC segmentation in the mentioned SBCs, but also finding if they can make predictions in a reasonable time. We also want to compare these prediction times with those obtained using cloud-based GPU and TPU devices.

As already discussed, it would be interesting and convenient for an ophthalmologist to be able to segment and obtain assistance for his or her diagnostic, directly from the image acquisition medical instrument. This would avoid the need of using a local GPU PC or having to upload the images to the web.

Thus, the importance and usefulness of SBCs with Deep Learning capabilities lies in the fact that these devices can operate autonomously. They do not need to be connected to servers equipped with GPUs or TPUs for performing predictions, since they have Machine Learning hardware built-in.

Moreover, Cloud GPUs and TPUs are usually free for non-commercial uses only. Therefore, the use of SBCs provided with Machine Learning hardware, such as the Coral Dev Board (that includes a Google Edge TPU in its design) or the NVIDIA Jetson boards[1] (that are equipped with GPUs), are interesting options to consider.

Also, the use of a Machine Learning accelerated SBC for performing medical image segmentation ensures the privacy protection of patient's health data, since the information is used locally by the SBC and, thus, it not sent to any cloud server for processing.

The rest of the paper is structured as follows: The background and related works are presented in Section 2. The methodological aspects are explained in Section 3, in order to describe the design of the experimental tests and thus provide a better understanding of the process followed to obtain the results. These results are described in Section 4 and discussed in Section 5. Finally, Section 6 draws the conclusions of this work and proposes future research lines.

2. Background and related works

A study of generalized U-Net architectures was performed in Civit-Masot et al. (2019) as a technique for implementing eye fundus image segmentation in the cloud.

[1] https://developer.nvidia.com/embedded/jetson-developer-kits.

In Civit-Masot et al. (2019) U-Net implementations deeper than the standard 5-layer network and with different layer increment ratios were tested. The use of normalization and drop-out as well as the influence of the initial layer width and the layer to layer width ratio (IR) were studied, since these attributes affect significantly both prediction quality and learning speed, and vary widely among different implementations.

Moreover, in a cloud based scenario, the neural networks must be trained as independently as possible from the acquisition source, since in a cloud-based service images will come from very different sources. Several segmentation researchers have used various datasets, but they always train and test with each of them independently. In our implementation data from several datasets were preprocessed and mixed in order to create independent datasets for training and validation.

Publicly available datasets were used. DRISHTI-GS (Sivaswamy et al., 2014), from Aravind Eye Hospital, Madurai (India), is a set of fundus images labeled by expert ophthalmologists for disc and cup. RIM-ONE-v3 (Fumero et al., 2011), from the MIAG group at the University of La Laguna (Spain), is a set of fundus images also labeled for disc and cup. Finally, DRIONS-DB (Carmona et al., 2008), from Miguel Servet Hospital, Saragossa (Spain), is a set of fundus images where only the optic cup has been labeled.

As a result of that initial study, a set of functions that enable the implementation of generalized U-Nets adapted to TPU execution was developed. These U-Nets are also suitable for developing cloud-based service implementations.

Regarding the use of embedded platforms and single-board computers for medical image segmentation, a low-cost Deep Learning ready GPU embedded platform has been used in Niepceron et al. (2020) for performing segmentation of brain tumors. An Nvidia Jetson AGX Xavier was selected in this study due to its low weight and low power consumption characteristics. Also, this developer kit embeds a modular scalable architecture called Deep Learning Accelerator which includes a support for many widely used CNNs.

An existing Deep learning architecture was selected and modified to be usable for both training and inference on the Jetson AGX Xavier platform. More specifically, a MobileNetV2 architecture has been compressed to reduce the number of trainable parameters and increase the training speed. Also, neural network 8-bit fixed-point quantization has been used for performing inferences in addition to the compression of the convolution layers.

Using the Jetson AGX Xavier maximum capacity, the compressed and quantized model was successfully trained, and it was able to segment high and low grade gliomas. The authors compared the model performance with other state of the art approaches, and proved that their method reached comparable results in relation to the reduction of parameters.

More specifically, regarding the use of embedded systems, mobile devices and single-board computers for performing eye fundus segmentation, a HW/SW embedded system that implements a Vertical Cup-to-Disc Ratio (VCDR) evaluation method for the diagnosis of glaucoma was presented in Dantas et al. (2016). This method, which is based mainly on morphological operations, has a reasonably low computational cost, but maintains an accuracy comparable to other related works using the RIM-ONE dataset.

It can be implemented on low power embedded processors with FPGA-based hardware acceleration for the morphological operations, to reduce execution time while maintaining accuracy. The proposed FPGA assisted architecture reduces execution time by at least 30% compared with software-only implementations running on platforms based on low power processors such as Raspberry Pi Model B, BeagleBoard-xM and a system using an Intel Atom processor with 2 GB DRAM.

In Martins et al. (2020), an interpretable computer-aided diagnosis (CAD) pipeline, that runs offline in mobile devices, is used for diagnosing glaucoma using fundus images. Several public datasets were merged and used to train convolutional networks for performing classification and segmentation tasks.

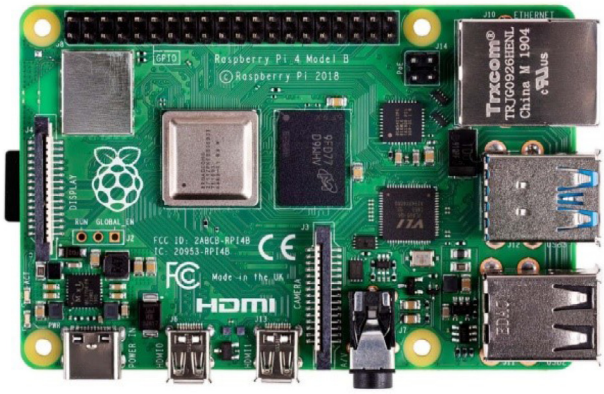


Fig. 3. Raspberry Pi 4 Model B.

These networks were used to build a pipeline that outputs a glaucoma confidence level, and also provides several morphological features and segmentations of relevant structures, resulting in an interpretable diagnosis in a similar fashion to Civit-Masot et al. (2020). This pipeline was integrated into a mobile app that run on a Samsung Galaxy S8 smartphone. Execution times – for CPU and GPU – and memory requirements were assessed.

In a similar way, in Pérez et al. (2020) a Deep Learning method for assessing the eye fundus image quality small enough to be deployed in a smartphone was presented. This method was validated with two different datasets, achieving good accuracy results.

The authors also measured the classification average elapsed time for the binary and three-class models on a smartphone running Android 9.0. The proposed method has a small number of parameters in comparison with other state-of-the-art models, and, thus, it is an attractive alternative for a mobile-based eye fundus quality classification system.

3. Materials and methods

We will start this section with a brief description of the hardware resources used. Then, the specification of the parameters of the generalized U-Nets selected for OC and OD segmentation as well as the datasets used, will complete the design of the experimental tests.

3.1. Hardware

Raspberry Pi[2] can be considered as a general-purpose computing device (Fig. 3), usually with a Linux operating system, that can run multiple programs in a multitasking environment. The Broadcom system-on-chip BCM2711 used by the latest version of the board (Raspberry Pi 4 Model B) (Halfacree, 2018) includes in its design a 64-bit quad-core Cortex-A72 ARM processor @ 1.5 GHz along with the new VideoCore VI 3D unit, and also a natively attached Gigabit Ethernet controller as well as a PCIe link that connects the USB ports.

Raspberry Pi 4 is also capable of addressing 1 GB, 2 GB or 4 GB LPDDR4 RAM depending on the variant of the model. The onboard wireless LAN (dual-band 802.11 b/g/n/ac) and Bluetooth 5.0 low-energy (BLE) connection capabilities make this device useful for the development of IoT applications. This single-board computer can run a variety of operating systems, such as Raspbian, which is the Foundation's officially supported operating system, Ubuntu Mate and Windows 10 IoT Core. We include this device in order to verify the performance of a widely used non hardware accelerated SBC in medical segmentation applications.

The Coral Dev Board (LLC, 2020a) is a single-board computer specifically developed to perform Machine Learning inferencing (Fig. 4). The

[2] https://www.raspberrypi.org/.

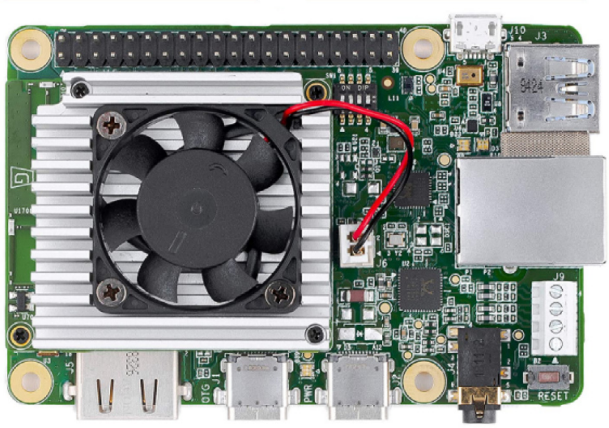


Fig. 4. Coral Dev Board.

Coral System-on-module (SoM), which is part of this prototype board, can be bought separately and used in a custom PCB hardware for production purposes.

The SoM is an integrated system that includes the NXP's IMX 8M system-on-chip (Quad-core Arm Cortex-A53 @ 1.5 GHz and Arm Cortex-M4F processors plus a Vivante GC7000Lite GPU), 8 GB eMMC memory, 1 GB LPDDR4 RAM, Wi-Fi 802.11 b/g/n/ac and Bluetooth 4.2 connection capabilities, and a Google Edge TPU coprocessor (a small ASIC which provides high performance Machine Learning inferencing for TensorFlow Lite models).

The baseboard for the SoM includes the usual connectors needed to perform a prototype project, such as Gigabit Ethernet port, CSI-2 camera interface, DSI display interface, 40 I/O pin header and USB 2.0/3.0 ports. Coral Dev Board uses Mendel operating system, which is a lightweight derivative of Debian Linux that runs on several Coral development boards.

3.2. Datasets

Regarding the datasets used in this work, both DRISHTI-GS and RIM-ONE-v3 have been used. Both are publicly available datasets and provide human expert OC and OD segmentation data and additional labels indicating if the images correspond to a patient with glaucoma or not. The labeling process includes the supervised evaluation of each of the dataset samples by a professional in the field.

There are other datasets, such as DRIONS-DB, that was used in previous studies, but it is not used in this work because only the optic cup has been labeled. However, this is not necessarily a drawback for achieving the objective of our study, specified in the introduction section, since relevant results can be obtained using the other two datasets.

DRISTI-GS dataset from Aravind Eye Hospital, Madurai (India), is made up of 101 color fundus images labeled for both disc and cup; and RIM-ONE dataset from the University of La Laguna is composed of 159 images also labeled for disc and cup.

In this work, 75% of the images from each dataset is used for training and the remaining 25% of the images for validating the results. This can be observed in Table 1.

The first column shows the number of images that are provided in those public datasets, the second column indicates the final amount of images used after data augmentation processes and, finally, the other two columns present the number of images used for training and testing purposes respectively.

Finally, in order to have more available data and thus to obtain more experimental results, a new dataset of thyroid gland ultrasound images (Wunderling et al., 2017) has been included. Thus, our study is extended to the segmentation of medical images different from those of eye fundus, that are obtained by other acquisition methods. This new dataset consists of 3665 images with their respective labels.

Table 1
Dataset summary.

Dataset	Images	Images after D.A.	Train (75%)	Test (25%)
DRISHTI-GS	101	2380	1785	595
RIM-ONE-v3	159	6980	5235	1745
TOTAL	260	9360	7020	2340

3.3. System architecture and testing method

After describing the hardware resources and datasets used, we will address the other aspects concerning the system architecture and the design of the experimental tests. In relation to the U-Nets, we have selected from Civit-Masot et al. (2019) a network with 6 levels, 40 filters in the initial layer and a layer-to-layer increment ratio of 1.1 (identified as 6/40/Y/1.1) with 0.9 MTP (millions of trainable parameters) for optic disc segmentation. We have selected a network with 6 levels, 64 filters in the initial layer and a layer-to-layer increment ratio of 1.1 (identified as 6/64/Y/1.1) with 2.4 MTP for optic cup segmentation. The small IR value reduces the number of trainable parameters greatly and is the key to efficient embedded implementation.

The first network is one of the U-Nets that provided better results for the optic disc segmentation. As for the optic cup segmentation, we have selected a U-Net used in Civit-Masot et al. (2020) that, without providing the best results (though they can be considered very good also), allows the generation of a suitable Tensor-Flow Lite model adequate to be processed by the Edge TPU Compiler. Logically, this tool does not admit models whose information does not fit in the memory size of the Edge TPU coprocessor.

A global graphical abstract of the implemented and tested system is shown in Fig. 5.

Additionally, these two U-Nets have been retrained using the thyroid ultrasound dataset in order to obtain two new models for our study. The first model (Thyroid_simple) has been obtained by retraining the U-Net used for OD segmentation, and the second one (Thyroid_complex) has been obtained by retraining the more complex U-Net used for performing OC segmentation (Fig. 6).

For all the experimental tests, prediction times have been obtained calling the `timeit.default_timer()` Python method just before and after making a prediction with the specific model.

Next, we show a general scheme of the developed test programs:

- Image dataset load.
- TensorFlow model definition and compilation, and load of the model weights for performing the experimental tests using the Raspberry Pi board and the iPython notebooks.
- Alternatively, TensorFlow Lite model conversion and load when using the Coral Dev Board for performing the experimental tests.
- Prediction execution with time measurement.

4. Results

The segmentation performance of the proposed system has already been studied in Civit-Masot et al. (2019, 2020). For completeness we include in Table 2 the Dice coefficients (Sørensen, 1948) obtained by our approach compared with other Deep Learning based alternatives that use the same public datasets. As we can see, results are fully comparable with other works even though our networks are trimmed for embedded implementation and we train with a combined dataset while the remaining authors train specifically for each dataset.

In Civit-Masot et al. (2020) we used the same network to segment the disc and cup while in this work we decided to use an smaller

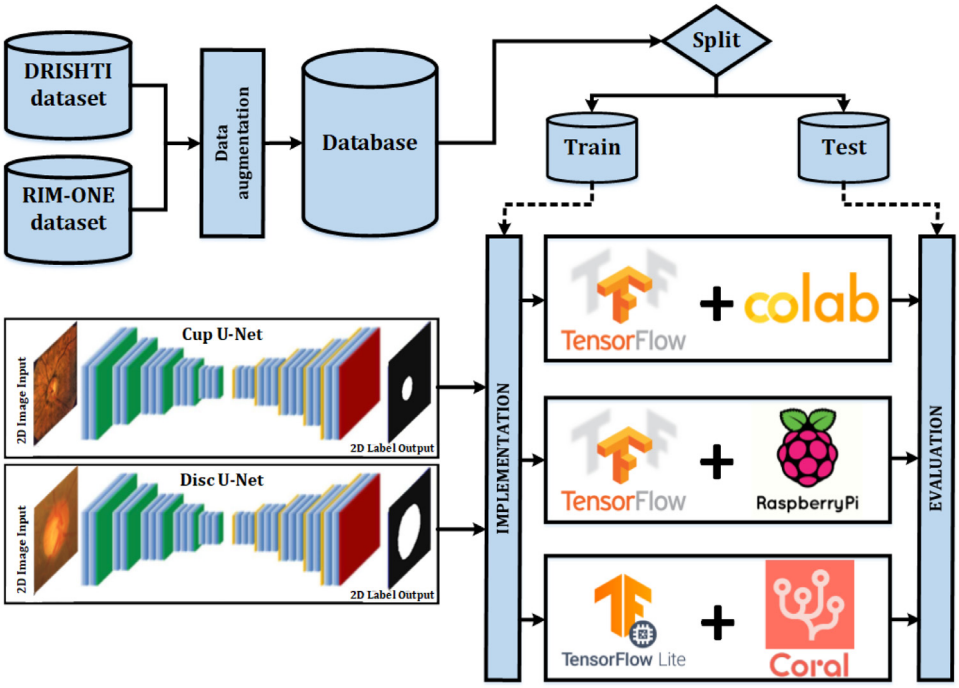


Fig. 5. Full system implementation.

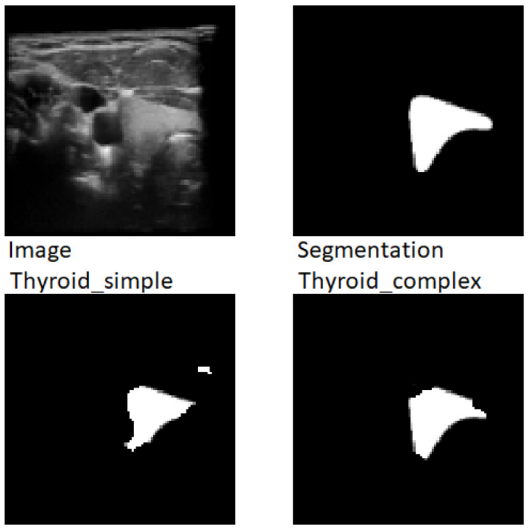


Fig. 6. Images from thyroid, segmentation, and predictions obtained with Thyroid_simple and Thyroid_complex models.

network for cup segmentation. This reduces the number of trainable parameters by a factor of almost 3, but clearly has some impact in the segmentation performance. If we used the same network for both OC and OD segmentation the Dice coefficient would be the same as those in Civit-Masot et al. (2020).

Regarding the segmentation performance, another aspect that needs to be mentioned is that on edge TPUs the models have to be quantized to 8 bit fixed precision numbers. We can see that this has a small impact on the segmentation performance. The obtained Dice coefficients when running the quantized models on Edge TPUs are shown on the last row of Table 2.

In the rest of the paper we will include results only related to prediction times, as this is the main objective of this work and the segmentation performance is almost identical in the different proposed implementations.

First, we have obtained a set of prediction times with the two selected U-Nets using Google Colaboratory notebooks.[3] These times have been obtained for GPU, CPU and TPU to be used as reference values (Tables 3 and 4), so that we can compare them with those prediction times obtained with Raspberry Pi and Coral Dev Board SBCs.

As in Civit-Masot et al. (2019), we have used the Google Colaboratory iPython notebook development environment. This environment supports TensorFlow and Keras (Chollet, 2018), and allows the implementation and training of networks using GPUs and TPUs (Google, 2020) in Google Cloud. In order to obtain prediction times in Google Colaboratory environment, we have used 2.4.1 and 2.4.0 versions of TensorFlow and Keras respectively.

Predictions using Colab notebooks have been made on an Intel(R) Xeon(R) CPU @2.20 GHz using a single core with two threads (Google, 2019). An Nvidia Tesla T4 has been used for making predictions using GPUs. Also, for performing predictions using Google Cloud TPUs, v2 TPU Pods have been used. A TPU v2 has 8 GiB of high-bandwidth memory and one matrix unit (MXU) for each TPU core. A v2 TPU Pod is a cluster consisting of up to 512 TPU cores and 4 TB of total memory (Google, 2020).

The first part of Tables 3 and 4 shows the results obtained using Google Colaboratory environment for optic disc and cup, and thyroid segmentation. These times have been calculated from the next ten predictions on a dataset after performing the initial prediction. Since TensorFlow performs a prediction over an entire dataset, the prediction time for a single element can be calculated as the total prediction time for a dataset divided by its number of elements. Thus, once the ten image prediction times have been obtained, the mean prediction time per image along with its standard deviation is shown.

The first prediction on a dataset using CPU and GPU takes more time than the next ones due to the necessity of performing a set of memory allocations and initializations. In the case of prediction times using Cloud TPU, the first prediction also involves sending the model information through the network and copying it into the TPU memory.

For implementing the U-Nets in Raspberry Pi 4 Model B and performing the experimental tests, 2.1.0 and 2.2.4-tf versions of TensorFlow and Keras (Q-engineering, 2020) have been used respectively. As

[3] https://colab.research.google.com.

Table 2
Dice coefficients for cup and disc segmentation.

Author	Cup RIM-ONE	Disc RIM-ONE	Cup DRISHTI	Disc DRISHTI
Zilly et al. (2017)	-	–	0.87	0.97
Sevastopolsky (2017)	0.82	0.94	–	–
Shankaranarayana et al. (2017)	0.94	0.98	–	–
Al-Bander et al. (2018)	0.69	0.90	0.83	0.95
Civit-Masot et al. (2020)	0.84	0.92	0.89	0.93
CPU/GPU/TPU	0.84	0.86	0.89	0.91
Edge TPU[a]	0.84	0.85	0.88	0.90

[a]Results obtained with quantized models.

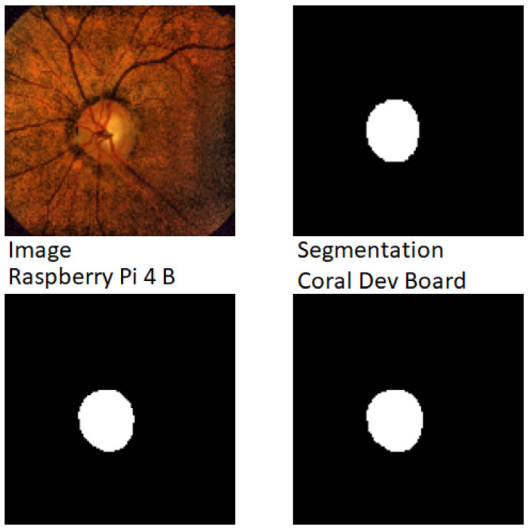


Fig. 7. Images from optic disc, segmentation, and predictions obtained with Raspberry Pi and Coral Dev Board.

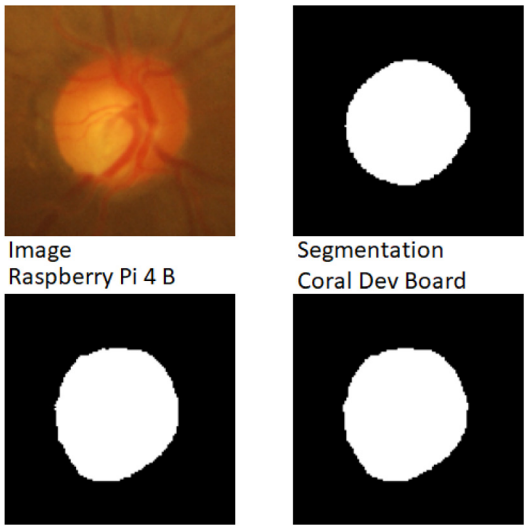


Fig. 8. Images from optic cup, segmentation, and predictions obtained with Raspberry Pi and Coral Dev Board.

with previous results and due to the same reason stated before, the corresponding times for Raspberry Pi using CPU have also been calculated using the next ten predictions on a dataset after the first inference. Again, once the ten image prediction times have been obtained dividing each total prediction time for a dataset by its number of elements, the mean prediction time per image along with their standard deviation are also shown.

For performing the experimental tests with Coral Dev Board, we have used TensorFlow Lite 2.5.0, which is the framework used to perform Deep Learning inferences on the Edge TPU (LLC, 2020c). First, predictions on DRISHTI-GS and RIM-ONE datasets with the TensorFlow Lite models for OD and OC segmentation, and predictions on THYROID dataset with the TensorFlow Lite models for segmenting the thyroid (Thyroid_simple and Thyroid_complex), have been obtained using the Coral Dev Board CPU.

In order to adapt the original TensorFlow models for the two U-Nets to the format used by TensorFlow Lite, the post-training quantization technique (Lite, 2020; LLC, 2019) has been used. The next step is to obtain suitable versions of the TensorFlow Lite models to be executed on the Edge TPU coprocessor (LLC, 2020e). The Edge TPU Compiler (LLC, 2020b) processes the information of a TensorFlow Lite model and generates an Edge TPU compatible model.

At the end of the process, this tool also generates a ".log" file indicating which operations of the original TensorFlow Lite model have been mapped to the Edge TPU coprocessor, and which operations will continue running on the Coral Dev Board CPU.

Moreover, the inference program that includes the model compiled for the Edge TPU must use a TensorFlow Lite delegate (LLC, 2020d), so that whenever the interpreter finds a graph node compiled for the Edge TPU, it sends that operation to the coprocessor executing the rest of the program on the ARM CPU. Thus, the file name of the Edge TPU runtime library must be passed to the `load_delegate()` method when initializing the interpreter.

The last part of Tables 3 and 4 shows the results for Coral Dev Board. Inference programs using TensorFlow Lite models do not perform predictions on an entire dataset as TensorFlow programs do (by calling the `predict()` method). Instead, TensorFlow Lite programs perform their predictions on the individual elements of a dataset. Thus, a loop must be used for iterating over each element.

Moreover, the first prediction time is not considered, since the first inference on the Edge TPU coprocessor is slow as it includes the load of the TensorFlow Lite model into the memory of the device (LLC, 2020c). Therefore, after performing the first inference, the prediction loop starts iterating over the first image of the dataset so that the first inference is performed again.

Finally, Figs. 7 and 8 show images from the optic disc and cup respectively along with the corresponding segmentation made by ophthalmologists, as well as the predictions performed with Raspberry Pi 4 Model B and Coral Dev Board SBCs.

5. Discussion

Once we know that the two SBCs selected for our study – Raspberry Pi and Coral Dev Board – are valid for performing segmentation of eye fundus and thyroid ultrasound images, and that predictions performed by these devices are basically equal to predictions made by Cloud CPUs, GPUs and TPUs when using Google Colaboratory notebooks, it is necessary to evaluate the performance of such devices – in particular the Coral Dev Board as it is equipped with specific Machine Learning hardware – and thus verify if they are capable of making predictions in reasonable times.

Table 3
Image prediction times for optic disc and Thyroid_simple (in milliseconds).

Dataset (shape)	Google Colaboratory			Raspberry Pi 4B	Coral Dev Board	
	CPU	GPU	TPU	CPU	CPU	TPU
DRISHTI (595, 128, 128, 3)	73.11 ± 0.44	2.16 ± 0.05	17.24 ± 1.91	259.52 ± 0.52	576.45 ± 0.19	8.55 ± 0.90
RIM-ONE (1745, 128, 128, 3)	71.84 ± 0.12	1.59 ± 0.04	7.71 ± 1.19	256.15 ± 0.59	576.37 ± 0.89	8.73 ± 1.16
THYROID (3665, 128, 128, 3)	76.22 ± 0.09	1.42 ± 0.01	4.88 ± 0.37	255.35 ± 0.26	575.71 ± 0.73	8.31 ± 1.06

Table 4
Image prediction times for optic cup and Thyroid_complex (in milliseconds).

Dataset (shape)	Google Colaboratory			Raspberry Pi 4B	Coral Dev Board	
	CPU	GPU	TPU	CPU	CPU	TPU
DRISHTI (595, 128, 128, 3)	163.72 ± 1.41	4.20 ± 0.18	38.49 ± 1.43	591.11 ± 1.21	1148.76 ± 0.22	21.64 ± 0.95
RIM-ONE (1745, 128, 128, 3)	160.61 ± 0.72	2.92 ± 0.21	15.81 ± 2.05	581.56 ± 1.01	1149.64 ± 1.65	21.47 ± 1.14
THYROID (3665, 128, 128, 3)	155.07 ± 2.96	2.29 ± 0.01	4.81 ± 0.15	576.83 ± 0.97	1145.45 ± 0.54	21.87 ± 0.81

Prediction times for CPU, GPU and TPU using Google Colab notebooks are clearly smaller than those obtained for Raspberry Pi 4 Model B, as expected. This result can be easily explained, since the performance of the CPU used in Colab notebooks (Intel(R) Xeon(R) CPU @ 2.20 GHz) is greater than that of Raspberry Pi 4 Model B CPU (Cortex-A72 ARM processor @ 1.5 GHz). Moreover, prediction times obtained when using Colab notebooks are obviously smaller for GPU and TPU than the ones obtained for CPU.

However, prediction times for TPU are greater, to some extent, than prediction times for GPU. This result can be explained by the delays due to data transmission over a network (Díaz del Río et al., 2016), since the CPU and the GPU are in the same node (and the communication between them is local) but the TPU pod (Google, 2020) is in another node. This makes cloud based TPUs much more useful for network training than for predictions on small data samples.

Regarding the results for Coral Dev Board, although our interest focuses on prediction times for the Edge TPU coprocessor, we also show prediction times for CPU in Tables 3 and 4. Thus, we can use them as reference values for highlighting the performance improvement when making predictions using Edge TPUs.

The corresponding results for optic disc and cup and thyroid (Tables 3 and 4) show that prediction times for the TensorFlow Lite models are appreciably greater than those for the TensorFlow models when using the Raspberry Pi 4 Model B board. This can be explained by the higher performance of Raspberry Pi 4 Model B CPU (Cortex-A72 @ 1.5 GHz) in relation to that of Coral Dev Board CPU (Cortex-A53 @ 1.5 GHz). Cortex-A72 microprocessor supports out-of-order execution, has a 15-stage pipeline (against the 8-stage pipeline of Cortex-A53), a more sophisticated branch predictor and greater L1 and L2 caches.

Regarding prediction times for Edge TPU, which are our main objective, we can see a significant performance improvement, as expected, in comparison with prediction times for CPU in Google Colab notebooks, Raspberry Pi and Coral Dev Board itself.

Finally, when comparing TPU prediction times, it can be observed that with a sufficiently large dataset (as RIM-ONE-v3 and THYROID in this case), prediction times in Tables 3 and 4 are smaller for Colab notebooks compared with prediction times for Coral Dev Board. The opposite is the case when the number of elements in the dataset is relatively small (as with DRISHTI-GS).

Since the CPU for a Colab notebook and the cluster of TPUs (TPU Pod) (Google, 2020) are in different nodes of a network, there is a data transmission time (Díaz del Río et al., 2016) which can be considered to be bounded except for a technical incidence that may arise in the system.

Thus, when the number of dataset images on which the predictions are performed is relatively large, the data transmission time ceases to be significant in relation to the total time for the set of predictions performed by Cloud TPUs, which have a much greater performance than the Edge TPU coprocessor. This last device is primarily intended for model inferencing, but not for training large and complex Machine Learning models[4] which is one of the main objectives of cloud based TPUs.

Therefore, for a sufficiently large number of predictions (n) on a dataset, the total time using Cloud TPUs along with the data transmission time over the network ends up being smaller than the total time used for predictions on the dataset using the Edge TPU.

$$SUCTET = \lim_{n\to\infty} \frac{n \times ETPT}{NDTT + n \times CTPT} \tag{1}$$

The speed up (SUCTET) in the performance of Cloud TPUs for Colab notebooks versus Edge TPU for Coral Dev Board can be expressed with Eq. (1). When the number of images (n) in the dataset on which predictions are made is relatively large, the network data transmission time (NDTT) ceases to be significant in relation to the total time for the set of inferences performed by Cloud TPUs. The term CTPT indicates the prediction time on a dataset element for the Cloud TPU pod, whereas ETPT indicates the prediction time for the Edge TPU on an element of the same dataset.

$$n \times ETPT < NDTT + n \times CTPT \tag{2}$$

$$n < \frac{NDTT}{ETPT - CTPT} \tag{3}$$

In order to know the maximum value for the number of images (n) of a dataset for which the performance of the Coral Dev Board Edge TPU is better than the performance of the Cloud TPU, the total prediction time on this dataset must be smaller when using the Edge TPU (Eq. (2)). Thus, for values of n that are lower than a maximum limit (Eq. (3)), the total prediction time using the Edge TPU will be smaller than the total prediction time using the Cloud TPU.

6. Conclusions and future works

The feasibility of using single-board computers for segmenting eye fundus images with a Deep Learning model in a reasonable time has been demonstrated experimentally: Less than 1.2 s per image for the worst case (using Coral Dev Board CPU) and less than 9 ms per image for the best case (using Coral Dev Board Edge TPU). It is clear that including specific Machine Learning hardware accelerators provides an speedup of over 130 times and thus allows many sophisticated segmentation problems to be performed in real time on embedded devices, such as many medical image acquisition instruments.

As future work, we plan to extend our study by including not only segmentation but also direct Glaucoma classification subsystems to be able to build explainable glaucoma diagnosis aids directly on the acquisition instrument. We also plan to use Coral accelerator devices[5] for performing alternative experimental tests on the proposed implementations. These devices incorporate an Edge TPU for performing Machine Learning inferencing in existing systems.

[4] https://coral.ai/docs/edgetpu/faq/.
[5] https://coral.ai/products/.

Coral USB Accelerator works with Debian Linux, macOS and Windows 10. It supports TensorFlow Lite framework and is compatible with Raspberry Pi boards.

Mini PCIe, M.2 A+E key and M.2 B+M key Accelerators are PCIe devices that also enable the integration of the Edge TPU coprocessor into existing systems. These three devices support Debian Linux operating system and TensorFlow Lite framework.

The results obtained from experimental tests on systems equipped with these TPU-based devices will allow us to extend this study and quantify the performance improvement when using high speed serial interfaces between Edge TPUs and CPUs. This case will most likely provide an additional delay when loading data into the Edge TPU and result in similar effects to those described in Eq. (1).

CRediT authorship contribution statement

Javier Civit-Masot: Conceptualization, Methodology, Software, Writing - original draft. **Francisco Luna-Perejón:** Software, Data curation, Writing - original draft. **José María Rodríguez Corral:** Conceptualization, Investigation, Methodology, Software, Tests, Writing - original draft. **Manuel Domínguez-Morales:** Conceptualization, Formal analysis, Methodology, Supervision, Validation, Writing - reviewing. **Arturo Morgado-Estévez:** Conceptualization, Formal analysis, Methodology, Supervision, Validation. **Antón Civit:** Conceptualization, Formal analysis, Methodology, Supervision, Validation, Writing - reviewing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was developed in the framework of the AUROVI Project, supported by the Ministry of Science, Innovation and Universities under Grant EQC2018-005190-P.

Figure 3 is provided by courtesy of Raspberry Pi Foundation. Figure 4 is provided by courtesy of Coral.

References

Akkara, J.D., Kuriakose, A., et al., 2019. Role of artificial intelligence and machine learning in ophthalmology. Kerala J. Ophthalmol. 31 (2), 150.

Al-Bander, B., et al., 2018. Dense fully convolutional segmentation of the optic disc and cup in colour fundus for glaucoma diagnosis. Symmetry 10 (4), 87.

Barros, D.M., et al., 2020. Machine learning applied to retinal image processing for glaucoma detection: review and perspective. Biomed. Eng. OnLine 19, 1–21.

Bourne, R.R., 2006. The optic nerve head in glaucoma. Community Eye Health 19 (59), 44.

Carmona, E.J., et al., 2008. Identification of the optic nerve head with genetic algorithms. Artif. Intell. Med. 43 (3), 243–259.

Chen, C., et al., 2020. Deep learning for cardiac image segmentation: A review. Front. Cardiovasc. Med. 7, 25.

Cheng, J., Liu, J., Xu, Y., Yin, F., Wong, D.W.K., Tan, N.-M., Tao, D., Cheng, C.-Y., et al., 2013. Superpixel classification based optic disc and optic cup segmentation for glaucoma screening. IEEE Trans. Med. Imaging 32 (6), 1019–1032.

Chollet, F., 2018. Deep Learning Mit Python Und Keras: Das Praxis-Handbuch Vom Entwickler Der Keras-Bibliothek. MITP-Verlags GmbH & Co. KG.

Civit-Masot, J., et al., 2019. TPU cloud-based generalized U-Net for eye fundus image segmentation. IEEE Access 7, 142379–142387.

Civit-Masot, J., et al., 2020. Dual machine-learning system to aid glaucoma diagnosis using disc and cup feature extraction. IEEE Access 8, 127519–127529.

Dantas, P.C., Sarmento, A., Sarmento, A., 2016. A HW/SW embedded system for accelerating diagnosis of glaucoma from eye fundus images. In: 2016 International Symposium on Rapid System Prototyping. RSP. IEEE, pp. 1–7.

Das, P., Nirmala, S., Medhi, J.P., 2016. Diagnosis of glaucoma using CDR and NRR area in retina images. Netw. Model. Anal. Health Inform. Bioinform. 5 (1), 3.

Díaz del Río, F., et al., 2016. Extending Amdahl's Law for the cloud computing era. Computer 49 (2), 14–22.

Fumero, F., et al., 2011. RIM-ONE: An open retinal image database for optic nerve evaluation. In: 2011 24th International Symposium on Computer-Based Medical Systems. CBMS. IEEE, pp. 1–6.

Google, 2019. Colab system specs. URL https://bit.ly/35G6LQZ.

Google, 2020. Cloud TPU system architecture. URL https://cloud.google.com/tpu/docs/system-architecture.

Halfacree, G., 2018. The Official Raspberry Pi Beginner's Guide: How To Use Your New Computer. Raspberry Pi PRESS.

Hassan, Q.F., Madani, S.A., et al., 2017. Internet of Things: Challenges, Advances, and Applications. CRC Press.

Hesamian, M.H., et al., 2019. Deep learning techniques for medical image segmentation: Achievements and challenges. J. Digit. Imaging 32 (4), 582–596.

Jonas, J.B., Bron, A.M., 2015. Optic Disc Photography in the Diagnosis of Glaucoma. Elsevier.

Lite, T.F., 2020. Post-training quantization. URL https://bit.ly/33C3rDI.

Litjens, G., et al., 2017. A survey on deep learning in medical image analysis. Med. Image Anal. 42, 60–88.

LLC, G., 2019. Retrain a classification model for edge TPU (with TF 2.0) - google colab. URL https://bit.ly/35Hgzuf.

LLC, G., 2020a. Coral dev board datasheet. URL https://bit.ly/2EbuwFq.

LLC, G., 2020b. Edge-TPU compiler. URL https://coral.ai/docs/edgetpu/compiler/.

LLC, G., 2020c. Get started with the dev board. URL https://coral.ai/docs/dev-board/get-started/.

LLC, G., 2020d. Run inference with tensorflow lite in python. URL https://coral.ai/docs/edgetpu/tflite-python/.

LLC, G., 2020e. Tensorflow models on the edge TPU. URL https://coral.ai/docs/edgetpu/models-intro/.

MacIver, S., MacDonald, D., Prokopich, C.L., 2017. Screening, diagnosis, and management of open angle glaucoma. Canad. J. Optom. 79 (1), 5–71.

Martins, J., Cardoso, J.S., Soares, F., 2020. Offline computer-aided diagnosis for glaucoma detection using fundus images targeted at mobile devices. Comput. Methods Programs Biomed. 192, 105341.

Nath, M.K., Dandapat, S., 2012. Techniques of glaucoma detection from color fundus images: A review. Int. J. Image Graph. Signal Process. 4 (9).

Niepceron, B., Nait-Sidi-Moh, A., Grassia, F., 2020. Moving medical image analysis to GPU embedded systems: Application to brain tumor segmentation. Appl. Artif. Intell. 34 (12), 866–879.

Patel, S.C., Patel, M.I., 2018. Analysis of CDR of fundus images for glaucoma detection. In: 2018 2nd International Conference on Trends in Electronics and Informatics. ICOEI. IEEE, pp. 1071–1074.

Pérez, A.D., Perdomo, O., González, F.A., 2020. A lightweight deep learning model for mobile eye fundus image quality assessment. In: 15th International Symposium on Medical Information Processing and Analysis, vol. 11330. International Society for Optics and Photonics, p. 113300K.

Q-engineering, 2020. Install tensorflow 2.1.0 on raspberry pi 4. URL https://bit.ly/3mr1Idd.

Quigley, H.A., 1985. Better methods in glaucoma diagnosis. Arch. Ophthalmol. 103 (2), 186–189.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, pp. 234–241.

Sevastopolsky, A., 2017. Optic disc and cup segmentation methods for glaucoma detection with modification of U-net convolutional neural network. Pattern Recognit. Image Anal. 27 (3), 618–624.

Shankaranarayana, S.M., et al., 2017. Joint optic disc and cup segmentation using fully convolutional and adversarial networks. In: OMIA 2017. In: Fetal, Infant and Ophthalmic Medical Image Analysis, Springer International Publishing, pp. 168–176.

Singh, K.J., Kapoor, D.S., 2017. Create your own internet of things: A survey of IoT platforms. IEEE Consum. Electron. Mag. 6 (2), 57–68.

Sivaswamy, J., et al., 2014. Drishti-gs: Retinal image dataset for optic nerve head (onh) segmentation. In: 2014 IEEE 11th International Symposium on Biomedical Imaging. ISBI. IEEE, pp. 53–56.

Sørensen, T.J., 1948. A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application To Analyses of the Vegetation on Danish Commons. I kommission hos E. Munksgaard.

Teikari, P., et al., 2019. Embedded deep learning in ophthalmology: making ophthalmic imaging smarter. Therap. Adv. Ophthalmol. 11, 2515841419827172.

Wunderling, T., et al., 2017. Comparison of thyroid segmentation techniques for 3D ultrasound. In: Medical Imaging 2017: Image Processing, vol. 10133. International Society for Optics and Photonics, pp. 346–352.

Zilly, J., Buhmann, J.M., Mahapatra, D., 2017. Glaucoma detection using entropy sampling and ensemble learning for automatic optic cup and disc segmentation. Comput. Med. Imaging Graph. 55, 28–41.