Microcontrollers on the Edge – Is ESP32 with Camera Ready for Machine Learning?

Kristian Dokic(✉)

Polytechnic in Pozega, 34000 Pozega, Croatia
kdjokic@vup.hr

Abstract. For most machine learning tasks big computing power is needed, but some tasks can be done with microcontrollers. In this paper well-known SoC ESP32 has been analyzed. It is usually used in IoT devices for data measurement, but some authors started to use simple machine learning algorithms with them. Generally, this paper will analyze the possibility of using ESP32 with a built-in camera for machine learning algorithms. Focus of research will be on durations of photographing and photograph processing, because that can be a bottleneck of a machine learning tasks.

For this purpose, logistic regression has been implemented on ESP32 with camera. It has been used to differentiate two handwritten letters on the greyscale pictures (“o” and “x”). Logistic regression weights have been calculated on the cloud, but then they have been transferred to an ESP32. The output results have been analyzed. The duration of photographing and processing were analyzed as well as the impact of implemented PSRAM memory on performances. It can be concluded that ESP32 with camera can be used for some simple machine learning tasks and for camera picture taking and preparing for other more powerful processors. Arduino IDE still does not provide enough level of optimization for implemented PSRAM memory.

Keywords: ESP32 · Logistic regression · Machine learning · IoT

1 Introduction

A microcontroller is a cheap and programmable system that generally includes memory and I/O interfaces on a single chip. They have been developed for decades but the main paradigm hasn’t changed all that time until the first decade of 21st century. The ubiquity of the internet has resulted in the appearance of services that offer to send, collecting and analyzing data from microcontrollers on the cloud services. In most cases connection between microcontroller and the Internet has been made through Wi-Fi. Cloud services offer lots of advantages like the reliability of cloud services and data visualization but in the last few years new paradigm has arrived – edge computing. Some authors declared that “edge computing refers to the enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of IoT services” [1].

On the other hand, in the last few years, a lot of microcontroller producers have worked on machine learning implementation on microcontrollers. Some of them have developed special libraries with machine learning functions [2, 3] but the others have implemented special hardware with enhanced machine learning capabilities [4, 5].

In this paper, a low-cost Chinese SoC ESP32 with camera has been analyzed. The ability to apply simple machine learning algorithms (Logistic Regression) has been tested as well as the impact of PSRAM memory implementation on performances. The focus of the research has been on durations of photographing and photograph processing. ESP32 has been chosen because it has caused great interest from the start of its production.

In the Sect. 2, few papers with ESP32 used for machine learning algorithms are presented. In Sect. 3, Logistic Regression implementation on an ESP32 with camera is presented. ESP32 board with a camera has been used to differentiate two letters on greyscale photos that have been taken from implemented camera. Logistic regression has been used to solve that problem, and the first part has been released with Google Collaboratory service. After that, final coding and microcontroller programming have been done with Arduino IDE. In Sect. 4, camera and picture processing speed have been analyzed, as well as the impact of PSRAM memory implementation on performances. Finally, in Sects. 5 and 6, discussion and conclusion can be found.

2 Related Work

ESP32 SoC is the second generation of Espressif corporation IoT solution and it includes WiFi and Bluetooth. It is based on the 32-bit RISC Tensilica Xtensa LX106 MCU with included FPU and DSP. Clock speed is 240 MHz, and it has 520 KB SRAM. Ivkovic put it in the MCU IoT Ready group because of FPU, DSP and WiFi components integrated [6]. In the ESP32 datasheets there is the application list where ESP32 can be used. Some of them are speech and picture recognition, internet radio players, an energy monitors and smart lighting, etc. [7].

Only a few authors used ESP32 for some ML tasks. Kokoulin et al. used ESP32 to reduce high network traffic and computing load of the central face recognition server. They implemented system based on a microcontroller that processes video stream from a public place and detects the presence of a face or silhouette fragment. Only pictures with faces or silhouette are sent to the main server. Estimated traffic decrease gains up to 80–90% [8].

Espressif System has been developed ESP-WHO framework for face recognition and detection and it is available on the GitHub [9, 10]. They cited that their framework is based on Multi-task Cascaded Convolutional Networks model and new mobile architecture - MobileNetV2 [11, 12].

3 Materials and Methods

Logistic regression is an algorithm for classification purposes. It is used when a model has to return a limited number of values, and the dependent variable is categorical. With only two possible outcomes Logistic Regression is called Binary Logistic Regression [13].

Logistic Regression is similar to Linear Regression, but the only difference is in the fact that output the weighted sum has to pass through a function that can map any value between zero and one. For that purpose, sigmoid function is used.

In this paper, binary logistic regression has been used, and θT values have been defined in the cloud. These values have been transferred to the ESP32 SoC, and they have been used for deciding about the letter in front of the camera connected with the microcontroller.

3.1 Hardware Part

In this paper, ESP32 with camera module is used. There are lots of vendors that produce ESP32 with a camera with different characteristics. In this paper, ESP32-CAM produced by AI-Thinker has been used as well as M5CAMERA by M5STACK. The main difference between them is that M5CAMERA is in a plastic box and it has 4 MB PSRAM memory. ESP32-CAM hasn't PSRAM memory, and it has only 512 kB RAM on the board. Both boards have the same camera, and it is OV2640.

3.2 Software Development

Logistic regression functions are well known, and it is implemented in lots of machine learning libraries. In this paper, logistic regression weights are calculated in the cloud, and they are transferred on the microcontroller board with a camera. The main idea was to use logistic regression to separate handwritten pictures of letters in two groups: pictures with letter "o" and pictures with letter "x". Described solution can be divided into three parts. First, some handwritten pictures with letters "o" and "x" have to be taken, and then conversion to vectors has to be done. After that, weights for output vector with Logistic Regression have to be calculated, and finally, weights have to be transferred to a microcontroller, including code for decision making.

First, on the twenty little papers, ten letters "o" and ten letters "x" have been written by hand. The pictures of these papers have been taken with the smartphone XIAOMI Redmi 5+ in resolution 3000 × 3000 pixels. These twenty pictures have been converted to 1000 × 1000 resolution with program IrfanView, and then pictures have been renamed. Pictures with letter "o" have been renamed to 0XX.jpg where XX is a number of picture. Pictures with letter "x" have been renamed to 1XX.jpg where XX is a number of picture.

These converted and renamed pictures have been uploaded to Collaboratory environment. Collaboratory is a free Jupyter notebook environment that runs entirely on the cloud. It can be used to write, save and execute code in Python. It provides access to powerful computing resources for free. After pictures uploading Python code converts color pictures to grayscale pictures and then, 1000 × 1000 pixels pictures to 3 × 3 pixel pictures.

After that values of 9 bytes from all twenty pictures have been transferred to an array with nine columns and 20 rows. One row for every picture, and one column for every pixel grey value. In the last tenth column value from the first number for every file, the name has been added (0 or 1). That array has been converted to CSV file named output3.csv. Values are between zero and one. Zero is for black color and one is for white color. It can be seen that the content of the first row of the file are values for the letter “x”. It is easy to prove it because the value for the fifth pixel in the center must be almost zero. It is 0.192156863.

Next step is weights calculating with Logistic Regression. Logistic Regression library from scikit-learn has been used. Code with comments is available on the GitHub. Output is file named weights3.csv with weights. It is nine-dimension vector that has to be transferred to a ESP32 with a camera, and it will be used to decide what picture is in the front of the camera.

Program for ESP32-CAM and M5CAMERA has been developed on Arduino IDE and the vector from the previous step has been imported as an array.

ESP32-CAM and M5CAMERA both have OV-2640 camera implemented, and different resolutions can be used, but the highest is 1600 × 1200. All available resolutions are rectangular and not quadratic. Pictures that have been used for calculating Logistic Regression weights have been quadratic. The solution is to cut parts of the taken picture from the left and the right side. All pictures and code for both SoC boards with comments is available on the address https://github.com/kristian1971/LogRegESP32-CAM/.

The program output from the ESP32-CAM and M5CAMERA is a number that can be received over the serial port. Programmed ESP32-CAM and M5CAMERA have been tested with different pictures and results were similar. Boards have sent numbers around 0,25 when a paper with letter “o” has been in front of the camera, as well as the numbers around value 0,70 when a paper with letter “x” has been in front of the camera. The device is sensitive to light, and light intensity change causes slight changes in the output values. Overview of proposed approach can be seen on the Fig. 1.

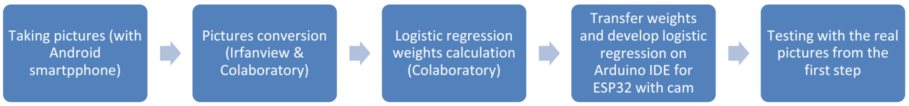
Taking pictures (with Android smartpphone)
Pictures conversion (Irfanview & Colaboratory)
Logistic regression weights calculation (Colaboratory)
Transfer weights and develop logistic regression on Arduino IDE for ESP32 with cam
Testing with the real pictures from the first step

Fig. 1. Overview of proposed approach

4 Testing and Results

After Logistic Regression implementation next goal has been to analyze the speed of the camera and photo processing on an ESP32-CAM and M5CAMERA boards. These boards can be used for taking pictures as well as preparing photos for some more powerful processor that can be connected with ESP32 board with serial, WiFi or Bluetooth connection. This second processor can be used for some more intensive calculations. This part of the research has been separated in the two parts. In the first part, ESP32-CAM and M5CAMERA have been used without 4 MB PSRAM and in the second part PSRAM has been enabled on M5CAMERA board.

The measurements were performed by inserting function micros() in the program code. This function returns a number of microseconds after microcontroller reset. When these values are subtracted, results are the execution times between inserted functions.

Duration of photographing presents time between command for photograph taking and the moment after the command execution. An example is here:

```
time1 = micros();
  //taking photograph
time2 = micros();
Total_time = time2-time1;
```

Photograph processing time presents the time needed to downsample picture and calculate logistic regression value. Downsampling is more time consuming than logistic regression calculating because all used pixels have to be taken into account.

4.1 Processing Speed Without PSRAM

Without 4 MB PSRAM memory, ESP32-CAM and M5CAMERA boards can use only one photo buffer. In Table 1, four different resolutions and duration of photographing and processing can be seen.

Table 1. Processing speed without PSRAM

Resolution	Pixels	Used pixels	Duration of photographing (ms)	Photograph processing (ms)	Overall time (ms)
QVGA	76800	57600	70	9	79
VGA	307200	230400	145	54	199
XGA	786432	589824	463	130	593
SXGA	1310720	1048576	540	254	794

It is obvious that the duration of photographing depends on photo resolution as well as photo processing depends on it. It can be seen in Figs. 2 and 3, too. There are two different charts because the camera takes a photo in full resolution but microcontroller processes only quadratic part of the photo. There is a difference between pixel numbers in taken and processed photos.

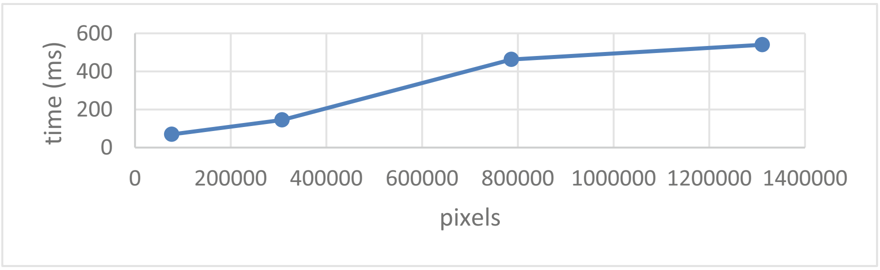

Fig. 2. Duration of photographing without PSRAM

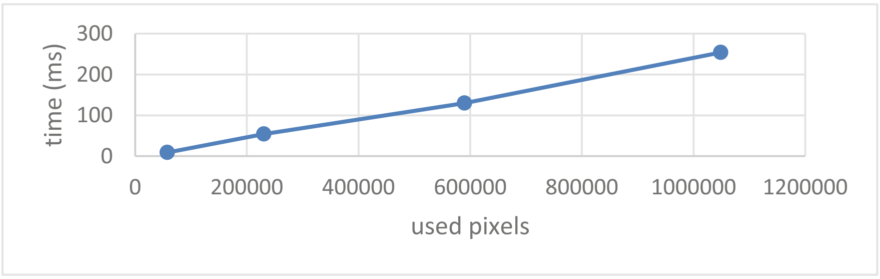

Fig. 3. Photo processing speed without PSRAM

4.2 Processing Speed with PSRAM

With 4 MB PSRAM memory, M5CAMERA board can use more than one photo buffer, so two photo buffers have been enabled in this part of analyze. The results are in Table 2.

Table 2. Processing speed with PSRAM

Resolution	Pixels	Used pixels	Photographing (ms)	Photo processing (ms)	Overall time (ms)
QVGA	76800	57600	<1	12	12
VGA	307200	230400	<1	360	360
XGA	786432	589824	<1	292	292
SXGA	1310720	1048576	<1	1710	1710

It can be seen that the duration of photographing is less than a millisecond despite photo resolution. On the other hand, photo processing times are much higher than in Table 1. Duration of photographing are not presented with the chart, but pho-to processing times are presented on Fig. 4.

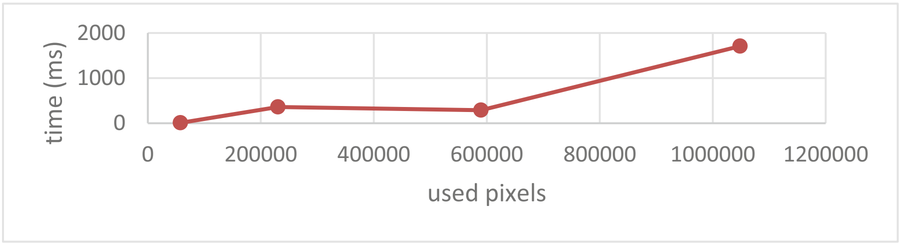

Fig. 4. Photo processing speed with PSRAM

5 Discussion

It is obvious that elapsed time to take a picture depends on picture resolution. It can be concluded that there is a linear connection between number of pixels and elapsed time to take a picture when microcontroller without PSRAM is used. It can be seen in Fig. 2. Values are between 70 ms and 540 ms.

Microcontroller with PSRAM has elapsed time to take a picture less than 1 ms, but as it is mentioned before, a microcontroller with PSRAM use two buffers. While the first picture is being processed, the second is already waiting in the buffer.

Picture processing time, as well as elapsed time to take a picture, depends on picture resolution. Picture processing time consists of downsampling time and logistic regression calculations. Downsampling process consists of two integer multiplications and three integer addition for each pixel in our case. The size of the picture after the downsampling process does not affect the duration of the process because all pixels have to be taken into account.

Microcontroller without PSRAM has downsampling time within the expected range. About 240 ns is needed per pixel for downsampling [14]. Logistic regression calculation time depends on downsampled picture size and includes floating-point multiplication and addition per downsampled picture pixel. Floating-point multiplication and addition take 54 ns with 32bit floating-point numbers [14]. Logistic regression calculation time is very short in our case because our downsampled picture has only nine pixels.

Microcontroller with PSRAM has much higher photo processing times. On producer website can be found that PSRAM uses the same cache as the external flash but when accessing large chunks of data (>32 KB) speeds will fall back to the access speed of the external RAM and it is little over 7 MB/sec [15, 16].

6 Conclusion

In this paper, ESP32 microcontroller has been presented because its quality is proven as well as the boards with cameras have been developed in the last year. Cameras on tested boards (ESP32-CAM and M5CAMERA) have sufficient resolution for most machine learning tasks.

Downsampling process can be time-consuming on ESP32, but machine learning algorithms usually use low-resolution pictures, so it is recommended to set camera capture resolution to the lowest levels. Logistic regression calculation speed depends on number of inputs, but with single-precision floating-point numbers, it lasts about 100 ns per input node. In our case with nine inputs it is negligible.

From our experience, PSRAM usage is not recommended with Arduino core for ESP32 WiFi chip, version 1.0.2. It looks like that Arduino IDE still does not provide enough level of optimization for all boards type. Espressit Systems has developed Espressif IoT Development Framework, and it is the official development framework for the ESP32 chip, and it probably provides the highest optimization level.

We can conclude that ESP32 with camera has enough computing power for simple machine learning tasks and for camera picture taking and preparing for other more powerful processors. In the future analysis it will be interesting to test ESP32 with different neural networks and to try to use both Tensilica Xtensa LX106 cores in calculations because ESP32 has two cores.

References

1. Shi, W., et al.: Edge computing: vision and challenges. IEEE. Int. Things J. **3**(5), 637–646 (2016)
2. ST Microelectronics, STM32Cube.AI: Convert Neural Networks into Optimized Code for STM32, 9 January 2019. https://blog.st.com/stm32cubeai-neural-networks/ Accessed 13 June 2019
3. Lai, L., Suda, N., Chandra, I.V.: CMSIS-NN: efficient neural network kernels for arm cortex-M CPUs. Comput. Res. Repository.svez. abs/1801.06601 (2018)
4. General vision, Presentation of the CurieNeurons on Arduino/Genuino101, 6 June 2016. https://www.general-vision.com/publications/PR_CurieNeuronsPresentation.pdf Accessed 16 June 2019
5. Allan, A.: Getting Started with the NVIDIA Jetson Nano Developer Kit, 15 April 2019. https://blog.hackster.io/getting-started-with-the-nvidia-jetson-nano-developer-kit-43aa7c298797 Accessed 18 June 2019
6. Ivkovic, I.J., Ivkovic, L.: Analysis of the performance of the new generation of 32-bit microcontrollers for IoT and big data application. In: Proceedings of the International Conference on Information Society and Technology (ICIST 2017), Kopaonik (2017)
7. Espressif Systems, ESP32 Datasheet, 10 April 2019. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf Accessed 11 June 2019
8. Kokoulin, A.N., Tur, A.I., Yuzhakov, A.A., Knyazev, I.A.I.: Hierarchical convolutional neural network architecture in distributed facial recognition system. In: 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Saint Petersburg and Moscow (2019)
9. Allan, A.: Face Detection and Recognition on the ESP32, 10 December 2018. https://blog.hackster.io/face-detection-and-recognition-on-the-esp32-3b4b9a35c765 Accessed 7 June 2019
10. Espressif System, ESP-WHO. https://github.com/espressif/esp-who Accessed 19 June 2019
11. Sandler, M., Howard, A., Zhu, M., Zhmoginov, I.L.A., Chen, C.: MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on computer vision and Pattern Recognition, Salt Lake City (2018)
12. Zhang, K., Zhang, Z., Lii, Z., Qiao, Y.: Joint face detection and alignment using multi-task cascaded convolutional networks. IEEE. Sig. Proc. Lett. **23**(10), 1499–1503 (2016)
13. Cox, D.: The regression analysis of binary sequences. J. Roy. Stat. Soc. Ser. B (Methodol.) **20**(2), 215–242 (1958)
14. Erich11, GitHub, 8 May 2019. https://github.com/espressif/arduino-esp32/issues/2538. Accessed 14 June 2019
15. ESP_Angus, ESP32.com, 13 September 2018. https://esp32.com/viewtopic.php?t=7158. Accessed 14 June 2019
16. Espressif, Support for external RAM, Espressif (2016). https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/external-ram.html. Accessed 15 June 2019