CONVOLUTIONAL NEURAL NETWORK ON EMBEDDED PLATFORM FOR PEOPLE PRESENCE DETECTION IN LOW RESOLUTION THERMAL IMAGES

Gianmarco Cerutti, Rahul Prasad, Elisabetta Farella

ICT-irst, Fondazione Bruno Kessler, Trento, Italy

ABSTRACT

Detection of human presence is a key feature in Human Computer Interaction. Solutions based on cameras are attractive, but require computer vision techniques to extract meaningful data, which can be expensive from a computational point of view. In this work, we present a new system that merges a low resolution thermal camera with advanced feature extraction techniques such as Convolutional Neural Networks. We demonstrate the possibility to adapt their execution to resource-constrained platform without significant loss of performance, by processing data on a 32-bit low power microcontroller, performing the classification on thermal video stream. It achieve 76.7% of accuracy in the microcontroller, requiring only $16.5\,\mathrm{mW}$ in continuous classification mode and using $6\,\mathrm{kB}$ of RAM.

Index Terms— CMSIS-NN, embedded systems, Convolutional Neural Network, low-resolution thermal camera, thermopile array

1. INTRODUCTION

People detection is a key step for many tasks like people counting [1], surveillance [2], generation of events for human computer interaction (e.g. automatic light switching [3]) and people tracking [4]. These tasks are useful in smart city scenarios, where human presence detection can be very useful in public spaces, which are in many cases outdoor and ample. However, this kind of scenario is more challenging than detection in indoor environments for many reasons, such as the high variability of several parameters, i.e. sudden changes of temperature and light due to weather conditions, or high variability of human and background temperature and wide daytime temperature variation. Therefore, more advanced data processing approaches are required to guarantee meaningful information extraction. At the same time, the typically broader size of outdoor spaces requires distributed sensing and intelligence. Always-on IoT end devices perform data analytic right at the source, reducing latency as well as energy consumption for data communication [5] [6].

Hence, this work targets low-cost, low-power and resource constrained IoT end devices, with the aim of detecting presence in outdoor scenarios while preserving privacy. In particular, we focus on outdoor people detection by means of low-resolution thermal cameras, using an innovative sensor technology, i.e. a thermopile arrays. Specifically, we used the Grid-EYE device, able to sense an 8x8 thermal image every $10\,\mathrm{Hz}$. Furthermore, we implemented a Convolutional Neural Network (CNN) in an ARM-cortex-M4 core based microcontroller, using the output of the sensor. To the best of our knowledge, there is no classifier based on thermopile array output using CNN.

To the purpose, we first trained the system using a custom dataset composed by short thermal videos. Then, we exploited the CMSIS-NN library for neural network developed by ARM in 2018 . Due to its recent release, there are still few works using this framework, which can achieve 4.6X improvement in runtime/throughput and 4.9X improvement in energy efficiency [7] with respect to a basic C implementation. This is enabled by the optimized DSP instructions, such as SIMD and MAC, integrated in M4 and M7 cores to perform workloads of the neural network. Some of these instructions have been used to accelerate low-precision computation in neural networks [8].

In this paper, we show the differences in classification accuracy between a 32 bit floating point representation and the 8 bit fixed point format of the weights, required by CMSIS-NN library. Moreover, we describe the procedure to port the trained classifier from a Tensorflow implementation in a microcontroller (CMSIS-NN).

To demonstrate the concept, we built a prototype composed by an evaluation board and a thermopile-array based sensor. On device measurement shows that it is possible to run a 3-layer convolutional neural network (4,8, and 16 filters respectively) on a microcontroller. The overall execution takes $4.1\,\mathrm{ms}$ to classify one image, thus enabling a real-time classification using $10\,\mathrm{Hz}$ ($100\,\mathrm{ms}$) thermal videos coming from the sensor. With the use of ultra low-power modes available for the chosen microcontroller, we achieve a power consumption of $16.5\,\mathrm{mW}$, including the sensor.

The organization of the paper is as follows. Firstly, we analyze common approaches for people detection and advantages of adopting thermopile arrays. Then, we describe how we built both network and automatic tools to export the model in a microcontroller. Finally, we propose the embedded implementation and we discuss power consumption, computation time, memory footprint and classification performance.

2. RELATED WORK

People detection systems usually include camera based solutions, but the output of this kind of sensors is strongly illumination-dependent and requires a high computational load [9]. Furthermore, vision systems often limit privacy.

Detection of people using infrared radiation is a broad research area. In this field, Passive InfraRed sensors (PIR) are often chosen as solution for fire alarm and intruder detection since the eighties [10]. However, there are many disadvantages related to the physics of the measured process, which is in many cases affected by noise. Some examples are background radiation, reflection and change in ambient temperature [11]. The main problem of PIR sensors is the inability to distinguish the human body from a generic source of heat, such as wind, change of light, etc. An alternative solution for detecting people in outdoor environment is using thermal cameras, which provides digital images instead of analog signal as PIR sensors. From these images, it is easier to differentiate humans from other sources of heat. On the other hand, their power consumption and cost are much higher than PIR sensors.

To detect human presence, in this work we use a thermopile array, which senses the temperature in its field of view and output low-resolution thermal images, i.e. 8 by 8 or 4 by 4. [12]

In a previous work, we made experiments using the Grid-EYE sensor for people detection [13]. We implemented 4 classic computer vision techniques and we analyzed the differences between them in terms of performance and computation load. However, the work was focused more on the novelty of the sensor and the image analysis was aimed at demonstrating the feasibility of the detection.

In the present work, we use the same innovative sensor, but we apply more advanced processing such as CNN. Moreover, differently from the previous work, we implement the processing on a microcontroller and characterize the system in terms of memory footprint, power consumption, etc.

A similar work is the one of Gomez et Al. [14], where the authors target edge computing implementing people counting using thermal images. They implemented a CNN on a microcontroller, analyzing the power consumption in a battery-powered prototype. In their case, the environment is indoor, therefore less affected by thermal noise caused by weather changing. They reach a power consumption of $34.4\,\mathrm{mW}$, which is close to our results. However, the execution time is ~$63\,\mathrm{s}$, with a consequent gap between two collected images sufficient to miss detection of people crossing and standing for shorter periods. In our case, the execution time of ~$5\,\mathrm{ms}$ guaranteeing an higher temporal resolution.

3. PROPOSED APPROACH

3.1. Dataset

To train the system we exploited the dataset collected in [13] extended with novel samples collected at different background temperatures. The setup is the same as before: the Grid-EYE camera was placed at $2.85\,\mathrm{m}$ of height, inclined of 60 degree to cover an area of $2\,\mathrm{m}$ x $1.4\,\mathrm{m}$. Labelling is done by means of temporization of crossing during acquisition, tagging the starting and ending instant.

We have collected 5 more acquisitions with respect to the previous one, for a total of 75 independent acquisitions (~21k frames). Moreover, we included more various temperatures.

3.2. Input Data And Preprocessing

The sensor used in this work is the Grid-EYE sensor, with the following features: 64 pixel output (8x8), 60 degree field of view, $10\,\mathrm{Hz}$ frame rate and ~$15\,\mathrm{mW}$ of power consumption. Keeping the image resolution small has two main advantages: it preserves the privacy of the detected person, and the processing required is reasonably light and enable its deployment in embedded platform [15].

We applied background subtraction before feeding the image to the neural network. This is because of the too wide range of possible temperature and to highlight the differences between the background and not only the absolute temperature value. In fact, a static sunspot can create an indistinguishable pattern from a low resolution image.

We implemented a running background average: the main advantage of this method is the adaptive maintenance of the background model while changes occur in the scene [16]. When the sensor is powered on, the background is initialized with the first frame. Afterwards, the algorithm updates each pixel of the background using an exponential filter:

$$BG_{i,j}[n] = (1-\alpha)BG_{i,j}[n-1] + \alpha I_{i,j}[n] \tag{1}$$

$BG[n]$ is the background image at the step n, i and j are the indexes of the image. The idea is to slowly adapt the background model based on the new frames.

Finally the input image is computed as following:

$$INPUT_{i,j}[n] = \mid BG_{i,j}[n] - I_{i,j}[n] \mid \tag{2}$$

The drawback of this method is object vanishing. The exponential filter incorporates any foreground element in the background after a certain time. This duration depends on the learning rate α. Mathematically, the step response of exponential filter reaches around 2/3 after the time constant τ. The relationship between the learning rate and the time constant, given the sampling period ΔT, is:

$$\alpha = e^{-\Delta T/\tau} \quad \longrightarrow \quad \tau = -\frac{\Delta T}{log(\alpha)} \tag{3}$$

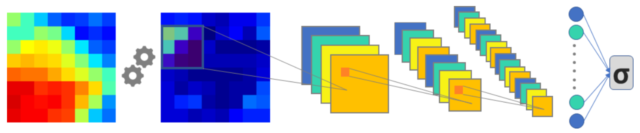

Fig. 1. Flowing diagram depicts the raw image taken from sensor, then performing a running average background and then passed to a CNN

For division implementation in microcontroller, alpha is a power of two ($2^{-7} \approx 0,008$). According to equation 3, it achieve around 10 s of time constant, that is a reasonable duration for people detection in outdoor. This parameter can be tuned according to the specific application requirements.

3.3. Network Description

The problem described in this work is presence detection. Formally, it is a binary classification between the two classes "person" and "no person" in the field of view.

Table 1 describes the network layers. The other common parameters between layers are kernel size 3 and stride 1. No max-pooling and no padding is applied: the number of pixel is already low (8x8) and decreasing it by pooling would harm the classification. We used no padding, adding extra pixels will change the image significantly. The activation function of each convolutional layer is the Rectified Linear Unit (ReLu). Finally, the last dense layer uses sigmoid as activation function, because a binary classification is required. The sigmoid represents a probability, thus we split the possible outputs in two classes with a threshold of 0.5.

Table 1. Layer descriptions and bytes needed in an 8 bit format for weights and intermediate outputs

	Layer Type	Filter Shape	Filter Size	Out Shape	Out Size
#0	Input	-	-	8x8x1	64B
#1	Conv.	3x3x1x4	36B	6x6x4	144B
#2	Conv.	3x3x4x8	288B	4x4x8	128B
#3	Conv.	3x3x8x16	1152B	2x2x16	64B
#4	F. C.	16x2x2	64B	1	1B

3.4. Training Procedure

We used the Python programming language and the framework Tensorflow. The input feature of the network are taken after background subtraction (see section 3.2). Then, we split the Dataset in training, validation and test in 60%-20%-20%. Cross entropy is the cost function and we used the Adam Optimizer to minimize the it, with a starting learning rate of 0.001. We stop the training after 1000 epochs and in each epoch the best model on validation set is saved. Then we used the test set to evaluate the final performance. Table 3 summarize results in the test set using Tensorflow.

3.5. Embedded Programming

To compute the output of the neural network on a microcontroller, we used the CMSIS-NN libraries. This framework achieves performance optimization of neural networks on resource-constrained microcontroller based platforms [7]. The calculation uses low-precision fixed point representation (e.g. 8 or 16 bit), to optimize the memory footprint and avoid the floating point computation. Moreover, the functions take advantage of the SIMD instructions to exploit parallel computing, always present in convolutional operations.

Many compatibility issue should be managed in the conversion from the computer to the embedded programming. Each weight is quantized to 8-bit fixed point format. To decide how many digits should be used for the integer part, we check the value of all the weights. The values are between -1 to 1, so we use all the bits to describe the digits, and 1 for the sign. In this way, the range of possible number is $[-1, 1-2^{-7}]$ with a resolution of 2^{-7}. The intermediate values (the output of each convolutional layer) is hard to predict as it varies with different input. One possible solution is to try the whole space of input, and determine the range of bits to represent decimal value for each intermediate value and preprocessing output.

This approach is not optimal for our system, because background subtraction output hardly reach the ideal maximum value (an item of 80° appears in a background model of 0°). Thus, we decided to use the training set to estimate the range of the value in practice. We evaluated how many numbers goes above a certain power of two. If the percentage of this numbers is negligible (1%), we use the respective number of digits for the integer, clipping the value above. This is a middle ground between loosing resolution (decimals) and clipping the unlikely integer.

Finally a program export all these information. Firstly, the weights are quantized according to this formula

$$W_q = round(W * 2^n) \tag{4}$$

where W_q is the quantized weight, W is the floating point weight in Tensorflow and n is the number of digits used to describe the decimal part. If the output number is bigger than 2^7 or smaller than $-(2^7 - 1)$, it is clipped to these values.

A program saves the weights in a header file, in a flattened way ([out channel, filter size, filter size,in channel]) according to the one expected from the CMSIS functions. The script automatically saves the number of integer digits for intermediates output in a header files, together with others hyperparameters like number of filters, output-image dimensions, padding and kernel size.

3.6. Hardware Platform

In order to evaluate the power consumption and the real-time performance of classification, we choose a generic low power hardware.

The board is the STM NUCLEO-L476RG. The main feature of its microcontroller is a Cortex-M4 core, up to 80 MHz clock and the possibility to reach very low consumption (down to 8 µA) in stop modes. The sensor outputs data every 100 ms. The forward propagation in the neural network takes less than 5% of this time in maximum frequency mode. As a result, the microcontroller has an idle period and it lets the system go in low power mode to reduce the overall power consumption. To wake up the system every 100 ms, the microcontroller has an integrated low power timer that enable periodic interrupt also during the sleeping mode, with a low-frequency clock of 32 kHz. Then when the system is woken up, the clock is set to 80 MHz and acquisition restarts.

4. RESULTS

In this section, we describe the hardware performance in terms of execution time, power consumption and memory footprint. Then, a different firmware enables the simulation of test images on the microcontroller and let us calculate the loss in classification performance due to the quantization process.

Table 2. Overall Results

Power Consumption [mW]	16.5
Power Consumption μc [mW]	2.3
Execution Time [ms]	4.01
Text [B]	19688
BSS [B]	2712
Data [B]	2680

Power Consumption: As explained in section 3.6, the algorithm goes in low power when it processes the image and it wakes up when the image from the sensor is ready. So the current absorbed is considerable during acquisition and processing, but it is negligible during stop mode. The device used for current measuring is the Real-Time Current Monitor EE203. STM board let the costumers to measure the consumption of microcontroller, without considering the current flowing in regulators, debugger and other devices on the board. Anyhow, the sensor is part of the system itself, so we have measured current absorbed from the voltage supply, with and without the sensor connected.

Computation Time: Part of the time is needed to load the image from the sensor via I2C bus, then it computes the background subtraction and lastly it performs the classification. Therefore, computation time is the time elapsed between starting reading and classification output. Total time is 4.01 ms; figure 4 shows the different parts in the active phase. Reading the data takes most of the time, background subtraction is negligible and classification takes around 2 ms.

Memory Footprint The compiler returns Text, BSS and Data values for the implemented firmware. Biggest size is text, that fits into ~20 kB Flash. The other two values in table

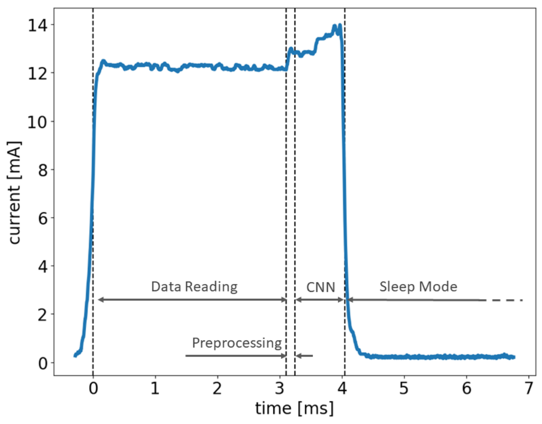

Fig. 2. Current consumption during the active phase.

2, BSS and Data, are respectively initialized and uninitialized variables. The sum of the two shows a total of ~6 kB in the RAM.

Classification Performance To evaluate the differences, we computed the accuracy of the test Dataset with Tensoflow on the PC. Then we send the Dataset via UART to the board, already preprocessed on the computer. This is to focus on the loss due to the classification, by not considering the contribution of preprocessing. We sent all the three sets (training, validation and test) to have a more generic description of the losses.

Table 3. Loss in classification between 32 bit floating point and 8 bit fixed point representation

Accuracy	Tensorflow [%]	CMSIS-NN [%]	Loss[%]
Train	81.1	80.9	0.2
Validation	77.4	76.4	1.0
Test	76.9	76.7	0.2

5. CONCLUSIONS

In this work, we implemented a Convolutional Neural Network on a microcontroller for people detection by means of low resolution thermal images. We demonstrated that is possible to achieve the full processing of the image within 4 ms and consuming only 2.3 mW without considering the sensor. Then we demonstrated that low precision format (8 bit fixed-point) does not affect significantly the performance, and it makes the system lose at maximum 1% of accuracy. Finally, it lets the network fits in just ~20 kB of Flash and ~6 kB of RAM.

6. ACKNOWLEDGMENT

This work was partially funded by project PAT L.P. 6/1999 Make it PAYT! coordinated by Gruppo Sartori Ambiente - Altares, protocol number 837.

References

[1] Ya-Li Hou and Grantham KH Pang, "People counting and human detection in a challenging situation," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 41, pp. 24–33, 2011.

[2] Alex Leykin and Riad Hammoud, "Robust multi-pedestrian tracking in thermal-visible surveillance videos," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*. IEEE, 2006, pp. 136–136.

[3] Gierad Laput, Yang Zhang, and Chris Harrison, "Synthetic sensors: Towards general-purpose sensing," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017, pp. 3986–3999.

[4] Piero Zappi, Elisabetta Farella, and Luca Benini, "Tracking motion direction and distance with pyroelectric ir sensors," *IEEE Sensors Journal*, vol. 10, no. 9, pp. 1486–1494, 2010.

[5] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] Manuele Rusci, Davide Rossi, Michela Lecca, Massimo Gottardi, Elisabetta Farella, and Luca Benini, "An event-driven ultra-low-power smart visual sensor," *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5344–5353, 2016.

[7] Liangzhen Lai, Naveen Suda, and Vikas Chandra, "Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus," *arXiv preprint arXiv:1801.06601*, 2018.

[8] Chao Li, Yi Yang, Min Feng, Srimat Chakradhar, and Huiyang Zhou, "Optimizing memory efficiency for deep convolutional neural networks on gpus," in *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for*. IEEE, 2016, pp. 633–644.

[9] Chakravartula Raghavachari, V Aparna, S Chithira, and Vidhya Balasubramanian, "A comparative study of vision based human detection techniques in people counting applications," *Procedia Computer Science*, vol. 58, pp. 461–469, 2015.

[10] RW Whatmore, "Pyroelectric devices and materials," *Reports on progress in physics*, vol. 49, no. 12, pp. 1335, 1986.

[11] Jurgen Kemper and Holger Linde, "Challenges of passive infrared indoor localization," in *Positioning, Navigation and Communication, 2008. WPNC 2008. 5th Workshop on*. IEEE, 2008, pp. 63–70.

[12] Masafumi Kimata, "Trends in small-format infrared array sensors," in *SENSORS, 2013 IEEE*. IEEE, 2013, pp. 1–4.

[13] Gianmarco Cerutti, Bojan Milosevic, and Elisabetta Farella, "Outdoor people detection in low resolution thermal images," in *2018 3rd International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE, 2018, pp. 1–6.

[14] Andres Gomez, Francesco Conti, and Luca Benini, "Thermal image-based cnn's for ultra-low power people recognition," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*. ACM, 2018, pp. 326–331.

[15] "Grid-eye state of the art thermal imaging solution," Tech. Rep., 03 2016.

[16] Andrews Sobral and Antoine Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4–21, 2014.