AUTOMATION AND COMPUTER-INTEGRATED TECHNOLOGIES

UDC 004.8 (045)
DOI:10.18372/1990-5548.72.16939

[1]V. M. Sineglazov,
[2]V. P. Khotsyanovsky

CAMERA IMAGE PROCESSING ON ESP32 MICROCONTROLLER WITH HELP OF CONVOLUTIONAL NEURAL NETWORK

Faculty of Air Navigation, Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine
E-mails: [1]svm@nau.edu.ua ORCID 0000-0002-3297-9060, [2]sttt912@yahoo.com

Abstract*—This paper analyzes a common ESP32 microcontroller with a built-in camera for image classification tasks using a convolutional neural network. ESP32 is commonly used in IoT devices to read data and control sensors, so its computing power is not significant, which has a positive effect on the cost of the device. The prevalence of ultra-low power embedded devices such as ESP32 will allow the widespread use of artificial intelligence built-in IoT devices. The duration of photographing and photo processing is obtained in the paper, as this can be a bottleneck of the microcontroller, especially together with machine learning algorithms. Deployed convolutional neural network, pre-trained on another device, MobileNet architecture on microcontroller and proved that ESP32 capacity is sufficient for simultaneous operation of both the camera and convolutional neural network.*

Index Terms—Machine learning; transfer learning; microcontrollers; image classification; ESP32**.**

I. INTRODUCTION

The Internet of Things(IoT) has become a reality. Smart home devices range from laptops to TVs, doorway cameras and dishwashers.

Intelligent buildings often include sensors, and electronic devices connected to the network for control, monitoring, and recording. Some sensors can measure vital alarms, location, or user activity. Finally, there are environmental sensors that detect things like temperature, light, or user presence.

This data as well as energy consumption measurement data of some devices can be recorded during the day and then uploaded to a remote server.

Also, it can be used for teaching machine learning models to provide the greatest comfort, economy and safety for the residents of a smart building.

On the other hand, technology, and solutions that are used to create IoT devices have significant limitations.

For example, if place a battery less camera in front of the door, then there are problems with the organization of its power supply. The nearby outlets can not be found, and battery power is not very handy, because exploitation will often have to change the batteries because of the rapid discharge during the streaming video transmission.

Thus, the organization of power supply becomes an important task in the development of IoT devices, which are constantly in an active state. To overcome this barrier, IoT devices must be "intelligent" [1]. It is necessary for them to act as independent processing devices and independently perform data processing, thus reducing the volume of transferred traffic and reducing energy consumption.

One of the ways to solve these problems is to integrate machine learning, namely neural networks into the intelligent block [2].

Neural networks solve tasks that traditional methods cannot compete with, it can successfully solve tasks, focusing on non-conventional, noisy, and spoiled information. A neural network is a system consisting of several simple computing elements (neurons) interconnected in some way. The most widespread are multilayer networks in which neurons are combined into spheres. The sphere, in its turn, is an assemblage of neurons to information from other neurons of the measure, i.e. outputs, is sent in parallel at every stroke of the clock.

However, one of the disadvantages of neural networks is that work requires a significant amount of computing resources that are available only on cost-effective computer systems. With the lapse of time and the development of mobile devices, the launch of neural networks and correct operation in systems with limited resources, such as microcontrollers (microcontrollers are cheap, programmable system, which often includes memory and input-output interfaces on one chip) became relevant.

The solution to this problem was the MobileNet family of neural networks [3].

Neural networks will make the house more useful and responsive to the needs of users by prediction instead of relying entirely on direct commands or programmed procedures manually. Also, neural network integration can potentially make energy management more efficient by limiting the use of the device only when it is needed, without causing inconvenience to the residents.

II. MOBILENET

MobileNet is a family of general-purpose computer vision neural networks designed for mobile devices to support classification, detection, etc. Mobile networks are small, low-latency, low-power models parameterized according to the resource constraints of different use cases. Although the basic MobileNet architecture (Table I) is already tiny and has low latency, sometimes possible to make the model be smaller and faster.

TABLE I. GENERAL ARCHITECTURE MOBILENETV1

Type/Stride	Filter Shape	Input Size
Conv/s2	3 x 3 x 3 x 32	224 x 224 x 3
Conv dw/s1	3 x 3 x 32	112 x 112 x 32
Conv / s1	1 x 1 x 32 x 64	112 x 112 x 32
Conv dw /s2	3 x 3 x 64	112 x 112 x 64
Conv /s1	1 x 1 x 64 x 128	56 x 56 x 64
Conv dw/s1	3 x 3 x 128	56 x 56 x 128
Conv /s1	1 x 1 x 128 x 128	56 x 56 x 128
Conv dw /s2	3 x 3 x 128	56 x 56 x 128
Conv /s1	1 x 1 x 128 x 256	28 x 28 x 128
Conv dw /s1	3 x 3 x 256	28 x 28 x 256
Conv /s1	1 x 1 x 256 x 256	28 x 28 x 256
Conv dw /s2	3 x 3 x 256	28 x 28 x 256
Conv/s1	1 x 1 x 256 x 512	14 x 14 x 256
5x Conv dw/sl	3 x 3 x 512	14 x 14 x 512
5x Conv /s1	1 x 1x 512 x 512	14 x 14 x 512
Conv dw / s2	3 x 3 x 512	14 x 14 x 512
Conv /s1	1 x 1 x 512 x 1024	7 x 7 x 512
Conv dw /s2	3 x 3 x 1024	7 x 7 x 1024
Conv/s1	1 x 1 x 1024 x 1024	7x 7 x 1024
Avg Pool/s1	Pool 7 x 7	7 x 7 x 1024
FC/s1	1024 x 1000	1 x 1 x 1024
Softmax / s1	Classifier	1 x 1x 1000

To address the issue of faster performance, MobileNetV2 was developed based on the ideas of MobileNetV1, using convolution with depth separation as effective building blocks. However, it introduces two new features into the architecture (Fig. 1):

1) linear bottlenecks between layers;
2) short connections between bottlenecks.

The development of the ideas laid down in the first versions of networks was the creation of MobileNetV3. It is tuned to the processors of phones by combining a network architecture, taking into account the hardware, augmented by the NetAdapt algorithm [17] and the new architecture (Fig. 2). To build a less resource-intensive model, MobileNet introduces a parameter α (alpha), which is called the width multiplier.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Fig. 1. General architecture MobileNetV2

The role of the width multiplier α is to liquefy the network evenly on each layer. However, the use of neural networks involves training them based on a training sample.

Input	Operator	exp size	$\#out$
$224^2 \times 3$	conv2d, 3x3	-	16
$112^2 \times 16$	bneck, 3x3	16	16
$56^2 \times 16$	bneck, 3x3	72	24
$28^2 \times 24$	bneck, 3x3	88	24
$28^2 \times 24$	bneck, 5x5	96	40
$14^2 \times 40$	bneck, 5x5	240	40
$14^2 \times 40$	bneck, 5x5	240	40
$14^2 \times 40$	bneck, 5x5	120	48
$14^2 \times 48$	bneck, 5x5	144	48
$14^2 \times 48$	bneck, 5x5	288	96
$7^2 \times 96$	bneck, 5x5	576	96
$7^2 \times 96$	bneck, 5x5	576	96
$7^2 \times 96$	conv2d, 1x1	-	576
$7^2 \times 576$	pool, 7x7	-	-
$1^2 \times 576$	conv2d 1x1, NBN	-	1280
$1^2 \times 1280$	conv2d 1x1, NBN	-	k

Fig. 2. General architecture MobileNetV3

III. THE PROBLEM OF THE LIMITED TRAINING SAMPLE

Due to the active development of neural networks in the last decade, the issues of training data set formation is of particular importance, since in many tasks, deep neural networks demonstrate quality that significantly exceeds other machine

learning algorithms, but to obtain such a gain in the quality, it is necessary to use large training samples (up to several million images).

On this basis, the problem of limited training samples arises. To mitigate this disadvantage and to reduce the cost of the training sample formation and the acceleration in obtaining a working prototype, should turn to transfer learning.

The first mention of the concept of transfer learning in machine learning dates back to 1993 in [5]. The concept consists in transferring knowledge obtained in one or more original tasks and using it to improve learning in the current task (see Fig. 3).

The techniques that enable knowledge transfer aim to make the machine learning process as effective as human learning. As a result, it was possible to retrain a convolutional neural network trained on a single sample of data to perform tasks on a new set of data, which significantly accelerated the learning process of the network. As a consequence, the material and time costs of forming the training sample and training the neural network were significantly reduced.

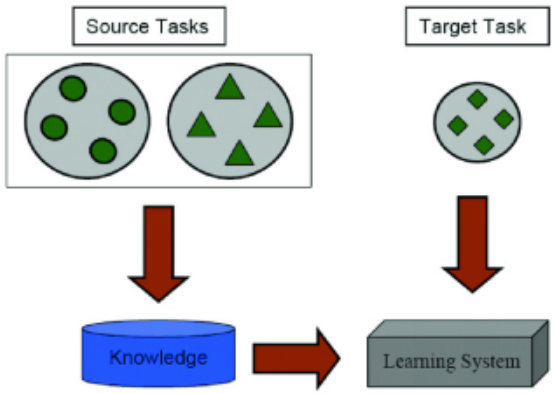

Fig. 3. Learning process of transfer learning

IV. Problem Statement

Given the advances in implementation and operability of the MobileNet architecture and the possibility of transfer learning, it is possible to run neural networks pre-trained on other devices (such as computers) in systems with low processing power, such as microcontrollers.

Over the past few years, many microcontroller manufacturers have worked on implementing machine learning on microcontrollers. Some of them have developed special libraries with machine learning features [6], [7], and others have implemented special hardware with advanced machine learning capabilities [8], [9].

Therefore, the goal of this paper is to organize the input of information from the camera to the ESP32 microcontroller and process it using a convolutional neural network that is deployed on this but has been trained on a different device.

The ESP32 was chosen because it generated a lot of interest from the beginning of production. The M5CAMERA (which is based on the ESP32 microcontroller) from M5STACK was used in this work. The board has 4 MB of PSRAM memory and a camera model OV2640 [10].

Since the camera operation and the image generation for the microcontroller is a complex task, before starting to work with neural networks it is necessary to investigate the image generation time to find the optimal quality without critical problems with the resulting system performance. Another task would be to investigate the effect of PSRAM memory [11] on the performance of such a system since modules with this memory are more expensive.

V. Research of Time of the Creation of a Photo on the ESP32 Microcontroller

To begin with, should study the effect of the PSRAM memory implementation on the performance of the controller. The measurements were made by inserting the function micros() into the program code [12]. This function returns the number of microseconds after the microcontroller starts. If subtracting these values, get the execution time between the inserted functions. The shooting time is the time between the command to shoot and the moment after the execution of the command.

```
time1=micros();
cam.getPhoto();
time2=micros();
total_time=time2-time1;
```

Without PSRAM memory the M5CAMERA board can use only one photo buffer. In Table II can be seen four different resolutions and duration of photo creation and processing.

Table II. Processing Speed Without PSRAM

Resolution	Maximum number of pixels in the image	Used pixels in the image	Shoo-ting time (ms)	Total time (ms)
QVGA	76800	57600	70	79
VGA	307200	230400	145	199
XGA	786432	589824	463	593
SXGA	1310720	1048576	540	794

The duration of the photo depends on the resolution of the photo and also depends on the time

of the photo creation. In Table I there are columns maximum number of pixels in the image and used pixels in the image – the results in them are different because the camera takes a photo in full resolution, but the microcontroller turns only a square part of the photo into an image, so there is a difference between the number of pixels in the taken and received photos at the end.

Now check the running time of the ESP32 microcontroller with PSRAM memory (Table III). It can be noticed that the duration of shooting is less than a millisecond, despite the resolution of the photos. Not counting the processing time of the photos is much higher than in the Table II.

TABLE III. PROCESSING SPEED WITH PSRAM

Resolution	Maximum number of pixels in the image	Used pixels in the image	Shooting time (ms)	Total time (ms)
QVGA	76800	57600	<1	12
VGA	307200	230400	<1	360
XGA	786432	589824	<1	292
SXGA	1310720	1048576	<1	1710

Taking into account data from Table II and III, means that using PSRAM memory decreases photo capture time, but significantly increases photo processing time. Therefore, the optimal camera mode is QVGA both for boards with PSRAM memory and without it, because machine learning algorithms usually use images with low resolution

VI. PREPARATION OF THE TRAINING SAMPLE AND TRAINING OF THE NEURAL NETWORK

Before training the network, it needs to form a training sample. To investigate transfer learning, the generated sample will consist of 210 initial images (70 images per class).

After the formation of the training sample, it should proceed to its automated augmentation by making small random changes to the training data (cropping or rotating images). For this purpose, the script shown in Fig. 4 has been implemented.

After enlarging the data set, its size is 96x96 pixels. Apart from resizing the images, also need to change the color gradation from RGB to grayscale to keep the actual color depth [13]. Also, working with grayscale helps to reduce the amount of final memory required for logical output.

After conducting operations on the training sample, move on to training the network. It will perform pre-training on a pre-trained model [16] MobileNetV1, which is trained on the ImageNet dataset [14], using Edge Impulse Studio [15].

MobileNetV1 is used because MobileNetV2 will need about 1.3 MB of RAM and 2.6 MB of ROM to run the model, which will cause significant delays in operation. At the same time, using MobileNetV1 and setting $\alpha = 0.10$ will result in less accuracy, but will only need about 53.2 KB of RAM and 101 KB of ROM. The results of the network training are shown in Fig. 5.

```
datagen = ImageDataGenerator(width_shift_range=.2,
                             height_shift_range=.2,
                             fill_mode='nearest')

datagen.fit(x_train)

for X_batch, y_batch in datagen.flow(x_train, y_train, batch_size=9):
    for i in range(0, 9):
        pyplot.subplot(330 + 1 + i)
        pyplot.imshow(X_batch[i].reshape(img_rows, img_cols, 3))
    pyplot.show()
    break
```

Fig. 4. Data augmentation script

Fig. 5. System training report

The model achieved about 77% accuracy, but the amount of RAM to be used in the output is about 60 Kbytes, which is reasonable when using the ESP32 controller along with the camera without significant power issues.

VII. PREPARATION OF THE TRAINING SAMPLE AND TRAINING OF THE NEURAL NETWORK

The learned model can be deployed as a library generated by Edge Impulse Studio, which will store the weights of the network to be connected to the project using code (Fig. 6). Where #include <ESP32-CAM.h> connects the file with the scales.

This code is a template that receives the image unprocessed (stored in the array features) and runs the classifier to output the network.

```
#include <ESP32-CAM.h>

static const float features[] = {


};

int raw_feature_get_data(size_t offset, size_t length, float *out_ptr) {
    memcpy(out_ptr, features + offset, length * sizeof(float));
    return 0;
}
```

Fig. 6. Code template for neural network startup

Primarily need to get the image from the camera, pre-process it by resizing it to 96x96, turning it into grayscale and smoothing it out. This will be the input tensor of the model. The output tensor will be a vector with values showing the probability of each of the classes.

For this, was taken the official code available for testing of the camera https://github.com/edgeimpulse/example-esp32-cam and merged it with the code of the trained neural network.

VIII. CONCLUSIONS

The ESP32 microcontroller was used in this work because its quality is proven. The OV2640 camera has sufficient resolution for most machine learning tasks.

The creation process can take a long time on the ESP32, but machine learning algorithms usually use low resolution images, so it is recommended to set the camera resolution to the lowest level.

It can be concluded that the ESP32 with the OV2640 camera has enough processing power to perform simple machine learning and camera photography tasks.

In a future analysis it will be interesting to test the ESP32 with different neural networks and try to use both Tensilica Xtensa LX106 cores in calculations.

REFERENCES

[1] D. Schweizer, M. Zehnder, H. Wache, H. Witschel, D. Zanatta and M. Rodriguez. *Using Consumer Behavior Data to Reduce Energy Consumption in Smart Homes: Applying Machine Learning to Save Energy without Lowering Comfort of Inhabitants*, 2015. https://doi.org/10.1109/ICMLA.2015.62

[2] Liciotti, Daniele & Bernardini, Michele & Romeo, Luca & Frontoni, Emanuele. *A Sequential Deep Learning Application for Recognising Human Activities in Smart Homes,* Neurocomputing, 2019, pp. 396. https://doi.org/10.1016/j.neucom.2018.10.104

[3] Keras documentation*: MobileNet, MobileNetV2, and MobileNetV3*. Keras: the Python deep learning API. https://keras.io/api/applications/mobilenet/

[4] Jian Mao, Qixiao Lin, Jingdong Bian. *Application of learning algorithms in smart home IoT system security*, 2018. https://doi.org/10.3934/mfc.2018004

[5] L. Y. Pratt, *Discriminability-based transfer between neural networks*. NIPS Conference: Advances in Neural Information Processing Systems, 1992.

[6] STM32Cube.AI: *Convert neural networks into optimized code for STM32*. ST life.augmented Blog. https://blog.st.com/stm32cubeai-neural-networks/

[7] L. Lai,, N. Suda, I. V. Chandra, *CMSIS-NN: efficient neural network kernels for arm cortex-M CPUs. Comput*. https://arxiv.org/abs/1801.06601, 2018

[8] General vision, Presentation of the Curie Neurons on Arduino/Genuino101,https://www.general-vision.com/publications/PR_CurieNeuronsPresentation.pdf

[9] Allan, A. *Getting started with the NVIDIA jetson nano developer kit*, 2019

[10] M5-docs. https://docs.m5stack.com/en/unit/m5camera

[11] Pseudostatic (random-access) memory (PSRAM) | JEDEC. https://www.jedec.org/standards-documents/dictionary/terms/pseudostatic-random-access-memory-psram, 2019

[12] Micros() – arduino reference. Arduino - Home. https://www.arduino.cc/reference/en/language/functions/time/micros/

[13] Stephen Johnson. *Stephen Johnson on Digital Photography*. O'Reilly. ISBN 0-596-52370-X , 2006

[14] Contributors to Wikimedia projects. ImageNet - Wikipedia. Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/ImageNet

[15] Intro to machine learning with edge impulse - silicon labs. (2020). Silicon Labs. https://www.silabs.com/support/training/intro-machine-learning-with-edge-impulse/intro-machine-learning-with-edge-impulse-presentation

[16] Tensorflow.(2022). models/research/slim/nets/mobilenet at master tensorflow/models. GitHub.(2022),https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet

[17] NetAdapt: *Platform-aware neural network adaptation for mobile applications*. (2018). arXiv.org. https://arxiv.org/abs/1804.03230

Received January 09, 2022

Sineglazov Victor. ORCID 0000-0002-3297-9060. Doctor of Engineering Science. Professor. Head of the Department. Aviation Computer-Integrated Complexes Department, Faculty of Air Navigation Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine.
Education: Kyiv Polytechnic Institute, Kyiv, Ukraine, (1973).
Research area: Air Navigation, Air Traffic Control, Identification of Complex Systems, Wind/Solar power plant, artificial intelligence.
Publications: more than 670 papers.
E-mail: svm@nau.edu.ua

Khotsyanovsky Volodymyr. Post-graduate Student.
Faculty of Air Navigation, Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine.
Education: National Aviation University, Kyiv (2020).
Research area: artificial intelligence.
Publications: 10.
E-mail: sttt912@yahoo.com

В. М. Синєглазов, В. П. Хоцянівський. Обробка зображень з камери на мікроконтролері ESP32 за допомогою згорткової нейронної мережі

У роботі проаналізовано поширений мікроконтролер ESP32 з вбудованою камерою для завдань класифікації зображень з використанням згорткової нейронної мережі. Зазвичай ESP32 використовується в пристроях IoT для зчитування даних та управління сенсорами тому його обчислювальна потужність не є значною, що позитивно впливає на вартість пристрою. Поширеність вбудованих пристроїв з наднизьким енергоспоживанням, таких як ESP32 дозволить масове поширення вбудованих пристроїв IoT із штучним інтелектом. В роботі одержано тривалість фотографування та обробки фотографій, оскільки це може бути вузьким місцем мікроконтролера, особливо разом з алгоритмами машинного навчання. Розгорнуто згорткову нейронну мережу, попередньо навчену на іншому пристрої, архітектури MobileNet на мікроконтролері та доведено, що потужностей ESP32 достатньо для одночасної роботи як камери так і згорткової нейронної мережі.
Ключові слова: машинне навчання; трансферне навчання; мікроконтролери; класифікація зображень; ESP32.

Синєглазов Віктор Михайлович. ORCID 0000-0002-3297-9060.
Доктор технічних наук. Професор. Завідувач кафедрою.
Кафедра авіаційних комп'ютерно-інтегрованих комплексів, Факультет аеронавігації, електроніки і телекомунікацій, Національний авіаційний університет, Київ, Україна.
Освіта: Київський політехнічний інститут, Київ, Україна, (1973).
Напрям наукової діяльності: аеронавігація, управління повітряним рухом, ідентифікація складних систем, вітроенергетичні установки, штучний інтелект.
Кількість публікацій: більше 670 наукових робіт.
E-mail: svm@nau.edu.ua

Хоцянівський Володимир Петрович. Аспірант.
Факультет аеронавігації, електроніки та телекомунікацій, Національний авіаційний університет, Київ, Україна.
Освіта: Національний авіаційний університет , Київ (2020).
Напрям наукової діяльності: штучний інтелект.
Кількість публікацій :10.
E-mail: sttt912@yahoo.com

В. М. Синєглазов, В. П. Хоцяновский. Обработка изображений с камеры на микроконтроллере ESP32 с помощью сверточной нейронной сети

В работе проанализирован распространенный микроконтроллер ESP32 со встроенной камерой для задач классификации изображений с использованием сверточной нейронной сети. Обычно ESP32 используется в устройствах IoT для считывания данных и управления сенсорами, поэтому его вычислительная мощность не является значительной, что положительно влияет на стоимость устройства. Распространенность встроенных устройств со сверхнизким энергопотреблением, таких как ESP32, позволит массовое распространение встроенных устройств IoT с искусственным интеллектом. В работе получена продолжительность фотографирования и обработки фотографий, поскольку это может быть узким местом микроконтроллера, особенно вместе с алгоритмами машинного обучения. Развернута сверточная нейронная сеть, предварительно обученная на другом устройстве, архитектура MobileNet на микроконтроллере и доказана, что мощностей ESP32 достаточно для одновременной работы как камеры так и сверточной нейронной сети.
Ключевые слова: машинное обучение; трансферное обучение; микроконтроллеры; классификация изображений; ESP32.

Синеглазов Виктор Михайлович. ORCID 0000-0002-3297-9060.
Доктор технических наук. Профессор. Заведующий кафедрой.
Кафедра авиационных компьютерно-интегрованных комплексов, Факультет аэронавигации, электроники и телекоммуникаций, Национальный авиационный университет, Киев, Украина.
Образование: Киевский политехнический институт, Киев, Украина, (1973).
Направление научной деятельности: аэронавигация, управление воздушным движением, идентификация сложных систем, ветроэнергетические установки, искусственный интеллект.
Количество публикаций: более 670 научных работ.
E-mail: svm@nau.edu.ua

Хоцяновский Владимир Петрович. Аспирант.
Факультет аэронавигации, электроники и телекоммуникаций, Национальный авиационный университет, Киев, Украина.
Образование: Национальный авиационный университет, Киев (2020).
Направление научной деятельности: искусственный интеллект.
Количество публикаций: 10.
E-mail: sttt912@yahoo.com