ISSN 2224-087X. Електроніка та інформаційні технології. 2022. Випуск 18. С. 3–11
Electronics and information technologies. 2022. Issue 18. P. 3–11

UDC 004.5 DOI: https://doi.org/10.30970/eli.18.1

DEEP LEARNING MODELS OF THE MANIPULATOR ON THE RASPBERRY PI PLATFORM FOR OBJECTS FINDING

Dmytro Myroniuk**,** Bohdan Blagitko

Ivan Franko National University of Lviv,
107 Tarnavsky St., UA–79017 Lviv, Ukraine
myronyukdmytro@gmail.com , blagitko@gmail.com

This article analyzed basic concepts of convolutional neural network construction. The modern deep learning architectures for solving image classification tasks describes. The prototype manipulator for the object finding realizes by using the Raspberry Pi 4 platform and TFLite framework. The garbage-finding manipulator prototype realizes.

Keywords: convolutional neural networks, deep learning, modeling, image classification.

Used Software

Here all prototype algorithms with Python programming language are realized. It is one of the most used tools for machine learning and data science, which contains a wide range of modules and frameworks for the actual model training process and deployment processes. It ease runs on most modern platforms and supports most of the tools on each of them.

Most modern deep learning frameworks include special modules for graph data type optimization and deploy them to mobile platforms. This work for mobile model construction and converting modules of the deep learning framework TensorFlow are used (TorchScript and TensorFlow Lite).

These frameworks provide functionalities for reducing the model size and RAM usage with minimized influence on result-optimized model accuracy. Result models can run on various mobile device types (e.g., microcontrollers, Mobil microcomputers, smartphones) with good precision and inference time.

TensorFlow Lite

TensorFlow Lite (or TFLite) – open-source deep learning framework for on-device inference is part of the deep learning TensorFlow ecosystem. It includes functionality for deep learning model optimization and a divided Interpreter for running converted models on various types of platforms with a single models format. As input, TFLite can take high-level models from Keras or TensorFlow models to universal ONNX format models. These features (especially ONNX support) make it easier to work with trains using other framework models.

TFLite framework consists of:

- the TensorFlow Lite interpreter runs specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers [3];

- the TensorFlow Lite converters TensorFlow models into an efficient form using an Interpreter and can introduce optimizations to improve binary size and performance [3].

Various types of model optimization are available. Tensorflow Lite quantization techniques describe in Table1.

Table1. TFLite quantization techniques

	Technique	***Benefits***	***Hardware***
1	Dynamic range quantization	4x smaller, 2x-3x speedup	CPU
2	Full integer quantization	4x smaller, 3x+ speedup	CPU, Edge TPU, Microcontrollers
3	Float16 quantization	2x smaller, GPU acceleration	CPU, GPU

For model construction, we use the PyTorch framework. It's one of the most popular deep learning frameworks used by outstanding companies like Facebook.

The main advantage of that library over others is that models in PyTorch have dynamic computational graphs. So, we can change our computational graph without recompilation. Moreover, PyTorch has many optimized pre-trained fine-tuning models, which can use as a base for powerful custom models.

Computational graphs and learning process plots can visualize via a compatible tensor board interface. The data type basic used in computations is tensor optimized for GPU using a multi-dimensional array.

Used Hardware

Training machine

The training and model fine-tuning, notebook on an Intel processor, and Nvidia GTX 1050 GPU used as a base. The base specification of the machine is described in Table2.

Table2. Notebook hardware specification.

	Parameter	***Value***
1	Processor	Intel Core i5 7300 HQ 2.5-3.4 GHz
2	RAM	24 GB
3	ROM	SSHD TOSHIBA MQ02ABD1 1 Tb + SSD Kingston A2000 250 GB
4	GPU	Nvidia GTX 1050 4 GB
5	CUDA	v. 10.2 with CuDNN v.7.2
6	Driver GPU	v. 460.39
7	OS	Ubuntu 20.04
8	Tensorflow	Ver. 2.4
9	Python	Ver. 3.8.5 Anaconds x64

Main Controller

The controller for the prototype Raspberry Pi 4 is used as the main. Raspberry Pi is an outstanding platform for robotics, smart home, IoT, etc. It uses an ARM-architecture-based CPU with four cores and includes all peripherals for using it as a "brain" of robotic systems. The full specification is described in Table3.

Table3. Raspberry Pi 4 specification.

	Parameter	***Value***
1	Processor	Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
2	RAM	4GB LPDDR4-3200 SDRAM
3	ROM	Kingston MicroSDHC 32Gb Class 10 Canvas Select
4	GPU	Broadcom VideoCore VI
5	CUDA	-
6	Driver GPU	-
7	OS	Raspberry Pi OS
8	Tensorflow	Ver. 2.4
9	Python	Ver. 3.8.5 Anaconds x64
10	GPIO	40-pin GPIO header
11	Bluetooth	Bluetooth 5.0, Bluetooth Low Energy (BLE)
12	Models quantization	Float16 +uint8

Used Models and Convolutional Classifier

Every convolutional classifier consists of two main parts:

- feature extractor (convolutional layers that transform our RGB image array to maps);
- that represent some features that are founding our image.

As a classifier, the network uses fully-connected layers with special architecture: 2-3 layers with classic sigmoid activation functions on the first layer and softmax function on the last layer. The typical convolution classifier architecture is presented in Fig. 1.

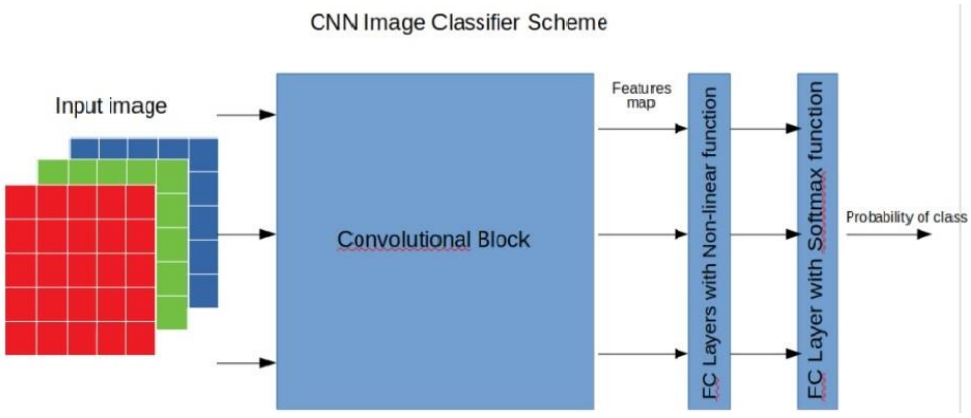

Fig. 1. Typical architecture of convolutional classifier

The benchmarks for popular models are in 2021. It can be including models embedded in libraries or not. Scientists can use it to choose optimal models by many parameters. It's the model accuracy, the computational model complexity, the hardware requirements, etc. For 2021, every used MS COCO (Microsoft Common Objects in Context) [5] and PASCAL VOC 2012 as the top list of used models. Besides, the benchmark [6] is analyzed.

The base architecture uses a simple sequential model type with one main flow and the flops in the convolutional block (like in ResNet architectures). Every CNN architecture can be separated into blocks. Each block consists of 2 types of layers – Convolution operation using a non-linear function and pooling operation.

Convolution operation. A typical convolution block consists of two main parts: entire convolution, where our input image in RGB matrix representation multiplied with a list of filters–matrices, which represent classic weights, and improves during learning. After that, we have an output feature map, that highlights the importance of some pixels of features. It can use as an identifier for certain classes of images. **Convolution operation** realizes three principles different from ANN:

1) Sparse interactions. The special convolution kernels are used in this type of network. It helps to reduce the dimension of weights compared to the input tensor. This feature reduces the computational complexity of a task and the memory volume in RAM or GPU memory.

2) Parameters sharing – every parameter is used only to compute neuron output in classic networks. In convolution, one kernel uses the computing output in every region of an input image. This feature helps to reduce the computational complexity of the network.

3) Eqvariance to broadcast – if the input function transforms, then the output function transforms with the same method.

The convolution operation computes by the following formula

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n) K(m, n) , \quad (1)$$

where: I - matrix of the Input image (channel), K - convolution Filter kernel, m, n– dimensions of kernel matrix, S – matrix of Output map.

The typical visualization of convolution is presenting Fig. 2.

Pooling operation. The operation **Pooling** applies a function of maximum function (Max Pooling), average (average pooling), l2 –norm, or weighted the average on feature maps. Any of these layer functions are:

1) reduce size of input tensor (list of feature maps after convolution can use large volumes of memory);

2) release independence from minor changes of the input tensor;

3) distinguish the entire entity rather than the location of the entity;

4) help to disassociate local signs in favor of global signs.

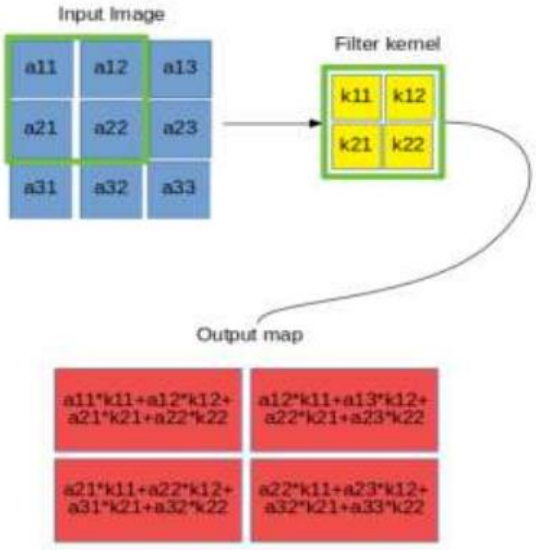

Fig. 2. Visualization of convolution

The convolutional neural classifier learning process. The special backpropagation method modifications allow learning kernel matrices to highlight important features there are to learn CNN. On classifier stage one uses a standard backpropagation function for weights correction:

$$w \Rightarrow w + \Delta w, \quad \Delta w = -\zeta \cdot \frac{\partial E}{\partial w} \tag{2}$$

where ζ – learning rate; $\frac{\partial E}{\partial w}$ – gradient of error.

For convolutional layers, one used kernel correction based on partial gradients of errors in feature map cells. Visual representation of the process is presenting in Fig. 2.

Fig. 3. The backpropagation function for the convolutional layer.

There are two main techniques based on convolutional neural networks. They used to learn deep learning system:

- **Training from scratch.** Model weights with random values and train models initialized. This method helps to train the best and most specific model for the concrete task. It should use an extremely large dataset of about 1000 images per class with powerful hardware in training. It can be expensive.

- **Transfer learning**. One can use the pre-trained model with weights that are oriented on the same feature recognition. The method can be used for the adjacent task. The model proposes using and retraining only classifier layers without training convolution kernels from scratch. This feature uses to improve the general accuracy of this model. It reduces the solution's computational complexity and reduces the training number of labeled images.

There are three techniques of transfer learning methods:

1) Using completed models from the library. We use a model from the pre-trained model list and replace the standard classifier with a custom (based on our number of output classes of images) just.

2) Design your model based on described architecture. One can design a model with specialized architecture and use pre-trained weights from popular competitions like MS COCO, ImageNet, Pascal VOC, etc. These can be found on the official competition site.

3) Combining several models and using several archives of weights.

Prototype specification

As a base prototype of the arm platform, variations of Uarm from Ufactory are used [3]. It is one of the best platforms for prototyping with open-source codes and SDK at and high price. In this article, variations of Uarm from Custom Electronics are used for prototype construction [4].

The arm is based on five servomotors (3x MG 996R and 2x MG 90 S). The parallel scheme with 2x3 serial-connected 18650 batteries (3.6V, 2200mAh) uses a power supply. The Raspberry Pi Camera ver.1.3 uses as the camera prototype.

The parameters of the lifting nodes and turning mechanism, the gripper, and the main properties of cameras, are given in Table 4.

TableI4. Prototype Specification

Parameter	**Value**
Main controller	Raspberry Pi 4 4 GB
Max weight, g	200
Angles of rotation, degrees	Forearm 1 (bottom) – 0-100 Forearm 2(top – wrist) – 0-90 Wrist: 0-180 Grabber (fingers): 180-0
Used models	MobileNet v2/YOLO v5
Count of classes to train	5-10
Models quantization	Float16 +uint8
Training strategy	Transfer Learning (ImageNet/COCO based)
Camera Module PCB dimensions	25mm x 24mm (9mm thickness)
Image resolution	5MP Max 2592 x 1944
Lens	f=3.6 mm, f/2.9
Viewing Angle	54° x 41°
Sensor size	3.67mm x 2.74mm (1/4" format)
Interface	Ribbon Cable

The main models for the prototype [8, 10] use fine-tuned models YOLO V5 small and MobileNet V3. Model MobileNet V3 Specifications describe in Table5. Their SE denotes whether Squeeze-And-Excite is in that block. NL denotes the nonlinearity type used. There HS denotes h-swish, and RE denotes ReLU. NBN denotes no batch normalization. *S* denotes stride.

Table5. Specifications for Model MobileNet V3 Small

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

These models demonstrate good recognition results with Raspberry Pi 4 and image size 224x224 px. It's enough to recognize correctly and classify objects on images. The strategies with quantization to float16 and uint8 are used to reduce the time recognition learning.

The manipulator prototype "Garbage classifier" is used as the main idea. The manipulator detects objects of irregular limited shape with deep learning models by using image processing algorithms. The maximum object dimensions are 200x90x90 mm. The ultrasonic sensor HC-SR04 measures the distance up the object to catch it and moves it to a special garbage box. The maximum mass of the object finding is 200g.

For model construction and training a Python programming language is used. It's used to deploy trained models for the mobile device Raspberry Pi 4 Model B 4 GB and run inference on it. Python is one of the most used programming languages for implementation and using machine learning algorithms and deep learning models. It includes a wide tool range, modules, and frameworks for effective training models and inference implementation.

A photo of the manipulator prototype “Garbage classifier” is considered in Fig.4.

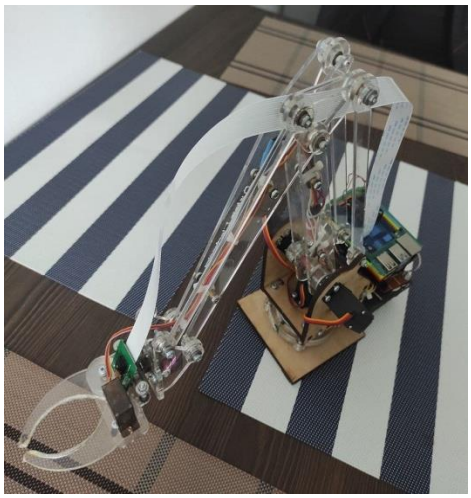

Fig. 4. Photo of the manipulator prototype "Garbage classifier".

Conclusion

The manipulator finding garbage prototype realized. The manipulator detects objects of irregular limited shape with deep learning models using image processing algorithms. It demonstrated an object recognition project example with high accuracy on a well-known Linux-based mobile platform Raspberry Pi 4. When tested, it was possible to correctly recognize and classify objects up to images with a size of 224x224 px.

References

[1] *Ian Goodfellow, Yoshua Bengio, Aaron Courville*. Deep Learning. An MIT Press Book //Ian Goodfellow, Yoshua Bengio, Aaron Courville. - MIT Press, 2016. - 716 p.

[2] Benchmark Analysis of Representative Deep Neural Network Architectures. Retrieved from:https://www.groundai.com/project/benchmark-analysis-of-representative-deep-neural-network-architectures/1

[3] TensorFlow Lite guide. Retrieved from: https://www.tensorflow.org/lite/guide

[4] Post-training quantization. Retrieved from: https://www.tensorflow.org/lite/performance/post_training_quantization

[5] Ufactory Offical Website. Retrieved from: https://www.ufactory.cc/#/en/support/download/products.

[6] Custom Electronics Robo-arm. Retrieved from: http://www.customelectronics.ru/robo-ruka-chast-1-opisanye/.

[7] COCO – Common Objects in Context. Retrieved from: http://cocodataset.org/#home

[8] Simply explain YOLOv5. Retrieved from: https://www.programmersought.com/article/20545272716/

[9] Raspberry Pi 4 Model B. Retrieved from https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[10] MobileNet Review. Retrieved from:https://lixinso.medium.com/mobilenet-c08928f2dba7

ISSN 2224-087X. Electronics and information technologies. 2022. Issue 18

МОДЕЛІ ГЛИБИННОГО НАВЧАННЯ МАНІПУЛЯТОРА НА ПЛАТФОРМІ RASPBERRY PI ДЛЯ ПОШУКУ ОБ'ЄКТІВ

Миронюк Дмитро, Благітко Богдан

Львівський національний університет імені Івана Франка,
вул. Ген. Тарнавського, 107, 79017 м. Львів, Україна
myronyukdmytro@gmail.com , blagitko@gmail.com

У цій статті аналізуються основні концепції побудови згорткових нейронних мереж. Сучасні системи, засновані на концепціях глибинного навчання і нейронних мережах різних типів, можуть вирішувати широкий спектр складних узагальнюючих завдань, особливо в області обробки зображень. Багато сучасних архітектур, які використовуються для вирішення цих завдань, є дуже дорогими з точки зору обчислень. Однією з популярних проблем для вирішення є навчання оптимізації моделі для використання мобільних платформ, таких як смартфони або Raspberry Pi. Більшість сучасних фреймворків глибинного навчання включають спеціальні модулі для оптимізації типів графових даних і розгортають їх на мобільних платформах. У цій роботі для побудови мобільних моделей і конвертації використовуються модулі глибинного навчання фреймворку TensorFlow (TorchScript і TensorFlow Lite). Ці структури забезпечують функціональні можливості для зменшення розміру моделі та використання оперативної пам'яті з мінімізованим впливом на точність оптимізованої моделі. Результативні моделі можуть працювати на різних типах мобільних пристроїв (наприклад, мікроконтролерах, мобільних мікрокомп'ютерах, смартфонах) із хорошою точністю та часом виводу. Доступні різні типи оптимізації моделі. Описується техніка квантування Tensorflow Lite. Для побудови моделі використовується фреймворк PyTorch. Це один із найпопулярніших фреймворків глибинного навчання, який використовують такі видатні компанії, як Facebook. Головна перевага цієї бібліотеки перед іншими полягає в тому, що моделі в PyTorch мають динамічні обчислювальні графи. Отже, ми можемо змінити наш обчислювальний граф без повторної компіляції. Крім того, PyTorch має багато оптимізованих попередньо навчених моделей тонкого налаштування, які можна використовувати як основу для потужних користувацьких моделей. За основу взято платформу для навчання та доопрацювання моделі, ноутбук на процесорі Intel і графічний процесор Nvidia GTX 1050. Як класифікатор мережа використовує повнозв'язані рівні зі спеціальною архітектурою: 2-3 рівні з класичними функціями сигмоїдної активації на першому рівні та функцією softmax на останньому рівні. Представлено типову архітектуру згорткового класифікатора. Спеціальні модифікації методу зворотного поширення дозволяють вивчати матриці ядра, щоб висвітлити важливі функції, які існують для вивчення CNN. Реалізовано прототип маніпулятора, який знаходить сміття. Маніпулятор виявляє об'єкти неправильної обмеженої форми за допомогою моделей глибокого навчання і алгоритмів обробки зображень. Маніпулятор продемонстрував приклад проекту розпізнавання об'єктів з високою точністю на відомій мобільній платформі Raspberry Pi 4 на базі Linux. Під час тестування можна було правильно розпізнати та класифікувати об'єкти аж до зображень розміром 224x224 пкс.

Ключові слова: згорткові нейронні мережі, глибинне навчання, моделювання, класифікація зображень.

Стаття надійшла до редакції 06.06.2022.
Прийнята до друку 15.09.2022.