Nghiên cứu nhận dạng vật sử dụng mạng nơ-ron Inception-v3 hoạt động trên Raspberry Pi 3B+

Nguyễn An Toàn*, Nguyễn Ngọc Thiện, Nguyễn Thanh Trực

Khoa Kỹ thuật và Công nghệ, Trường Đại học Quy Nhơn, Việt Nam

**Corresponding author. Email: nguyenantoan@qnu.edu.vn*

ABSTRACT

Phân loại ảnh (Image Classification) là bài toán quan trọng bậc nhất trong lĩnh vực thị giác máy tính (Computer Vision). Mặc dù nó đơn giản nhưng lại có rất nhiều ứng dụng thực tế, phân loại hình ảnh có nhiệm vụ gán một nhãn cho ảnh đầu vào từ một nhóm danh mục cố định. Bài báo này đã ứng dụng việc phân loại hình ảnh để nhận dạng vật bằng cách đưa hình ảnh của vật cần nhận dạng sau đó tiến hành gán nhãn cho hình ảnh rồi thông báo tên nhãn (tên vật) qua kênh âm thanh. Việc phân loại dựa trên mô hình mạng nơ-ron Inception-v3 đã được đào tạo sẵn trên Tensorflow và sử dụng hệ điều hành Raspberian chạy trên Raspberry Pi 3B+ tạo ra một thiết bị có khả năng nhận dạng vật nhỏ gọn và tiện lợi có thể ứng dụng trong nhiều lĩnh vực.

Từ khóa: *Nhận dạng vật, Raspberry, mạng nơ-ron tích chập, mạng nơ-ron Inception-v3, Tensorflow.*

Research of object recognition using neural network Inception-v3 model operating on Raspberry Pi B3+

Nguyen An Toan*, Nguyen Ngoc Thien, Nguyen Thanh Truc

Faculty of Engineering and Technology, Quy Nhon University, Vietnam

** Corresponding author. Email: nguyenantoan@qnu.edu.vn*

TÓM TẮT

Image Classification is the most important problem in the field of computer vision. It is very simple and has many practical applications, the image classifier is responsible for assigning a label to the input image from a fixed category group. This article has applied image classification to identify objects by giving the image of the object to be identified, then labeling the image and announcing the label name (object name) through the audio channel. The classification is based on the neural network Inception-v3 model that has been trained on Tensorflow and used Raspberian operating system running on the Raspberry Pi 3 B+ to create a device capable of recognizing objects which compact size and convenient to apply in many fields.

Keywords: *Object recognition, Raspberry, convolutional neural network, neural network Inception-v3, Tensorflow.*

1. INTRODUCTION

In the Industry 4.0, the application of recognition technology in the working environment and industry has posed the problem that the need for devices capable of image classification to recognize objects completely automatically without need human support. With the above requirement, many methods to assist and replace human vision have also been developed to recognize surrounding objects such as detecting and vibrating devices when there are obstacles by Sahoo et al.[1] Currently, there is OrCam MyEye 2 device[2] that supports object recognition but it is a foreign product, English language and high price. These devices have partly assisted in identifying objects but are not really reasonable and can meet the needs of object identification for Vietnamese users. Therefore, it is necessary to have a compact, reasonable cost device that supports object recognition to replace the recognition of objects by human vision.

Many computer vision technologies have been developed today, and one of the technologies that can help with object recognition is image classification.[3] Image classification process can be based on many different convolutional neural network models such as LeNet-5, AlexNet, VGG-16, Inception-v3, ResNet-50... In this study, the authors propose a device that uses a Raspberry Pi 3B+ to recognize images of objects captured from a camera. This study applies the image classification problem using the Inception-v3 model.[4] With this technology, the identification of objects will become easy, accurately respond to information of objects with many modes of operation and flexible support for users.

2. CONTENT

2.1. Convolutional neural network

Convolutional neural network (CNN)[5,6] is network architecture for deep learning.[7] CNN are deep artificial neural networks that are used primarily to classify images cluster them by similarity and perform object recognition [8] within scenes. A CNN is comprised of one or more convolutional layers and then followed by one or more fully connected layers as in a standard multilayer neural network. It learns directly from images. A CNN can be trained to do image analysis tasks including classification, object detection, segmentation and image processing.

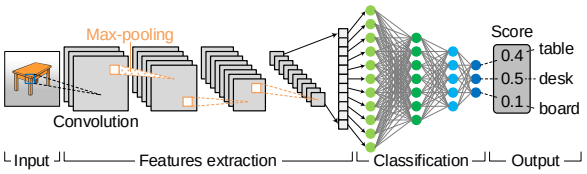

Figure 1. The overall architecture of the CNN.

Figure 1 shows the overall architecture of CNNs consists of two main parts: Feature extractors and a classifier. In the feature extraction layers, each layer of the network receives the output from its immediate previous layer as its input and passes its output as the input to the next layer. The CNN architecture consists of three types of layers: Convolution, max-pooling, and classification.

2.1.1. Convolutional Layer

In this layer, feature maps from previous layers are convolved with learnable kernels. The output of the kernels goes through a linear or non-linear activation function, such as sigmoid, hyperbolic tangent, Softmax, rectified linear, and identity functions) to form the output feature maps. Each of the output feature maps can be combined with more than one input feature map. In general, we have that:

$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j^l\right) \quad (1)$$

where:

x_j^l is the output of the current layer

x_i^{l-1} is the previous layer output

k_{ij}^l is the kernel for the present layer

b_j^l are the biases for the current layer

M_j represents a selection of input maps.

For each output map, an additive bias *b* is given. However, the input maps will be convolved with distinct kernels to generate the corresponding output maps. The output maps finally go through a linear or non-linear activation function (such as sigmoid, hyperbolic tangent, Softmax, rectified linear, or identity functions).

2.1.2. Sub-sampling Layer

The subsampling layer performs the down sampled operation on the input maps. This is commonly known as the pooling layer. In this layer, the number of input and output feature maps does not change. For example, if there are *n* input maps, then there will be exactly *n* output maps. Due to the down sampling operation, the size of each dimension of the output maps will be reduced, depending on the size of the down sampling mask. For example, if a 2×2 down sampling kernel is used, then each output dimension will be half of the corresponding input dimension for all the images. This operation can be formulated as:

$$x_j^l = down\left(x_i^{l-1}\right) \quad (2)$$

Where *down*(.) represents a sub-sampling function. Two types of operations are mostly performed in this layer: Average pooling or max-pooling. In the case of the average pooling approach, the function usually sums up over *n*×*n* patches of the feature maps from the previous layer and selects the average value. On the other hand, in the case of max-pooling, the highest value is selected from the *n*×*n* patches of the feature maps.Therefore, the output map dimensions are reduced by n times. In some special cases, each output map is multiplied with a scalar.Some alternative sub-sampling layers have been proposed, such as fractional max-pooling layer and sub-sampling with convolution.[9]

2.1.3. Classification Layer

This is the fully connected layer which computes the score of each class from the extracted features from a convolutional layer in the preceding steps. The final layer feature maps are represented as vectors with scalar values which are passed to the fully connected layers. The fully connected feed-forward neural layers are used as a soft-max classification layer. Figure 2 shows the basic operations in the convolution and sub-sampling of an input image.

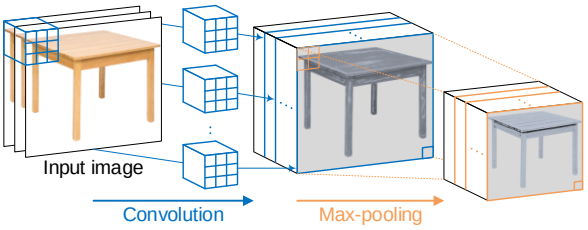

Figure 2. Feature maps after performing convolution and pooling operations.

As the fully connected layers are expensive in terms of computation, alternative approaches have been proposed during the last few years. In most cases, two to four layers have been observed in different architectures, including LeNet, AlexNet, and VGG Net. In this study, the Inception-v3 model has been proposed to be used for image classification with advantages over other models.

2.2. Inception-v3 model

Inception-v3 model is considered one of the CNN architectures for image classification. It was first introduced by Szegedy et al.[4] This model contains more than 25 million of fitted parameters and was trained by the one of the top hardware experts in the industry.

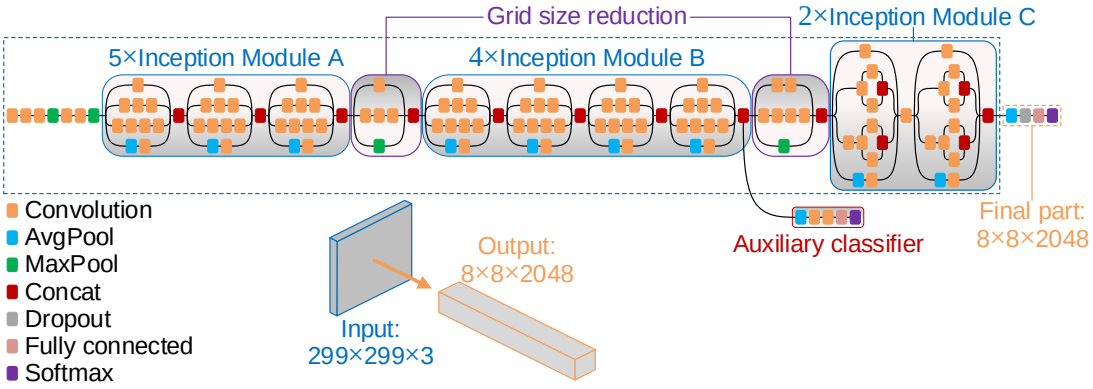

Figure 3. The architecture of the Inception-v3 model.

The CNN models are black boxes that construct image features. The Inception-v3 model uses the image feature extraction module which was trained on ImageNet. The ImageNet is an accessible database for highresolution images designed for developers and researchers in the field of image processing.Generally, the Inception-v3 model consists of two main parts: (1) the CNN to extract image features; and (2) the image classification with the softmax and the fully-connected layers. The softmax layer is used as the final layer of a neural network-based classifier in order to provide normalized class likelihoods for the outputs.

The main difference between Inception-v3 model and traditional neural network[8] are Inception blocks. They involve input transforming with multiple filters and linking their results together.

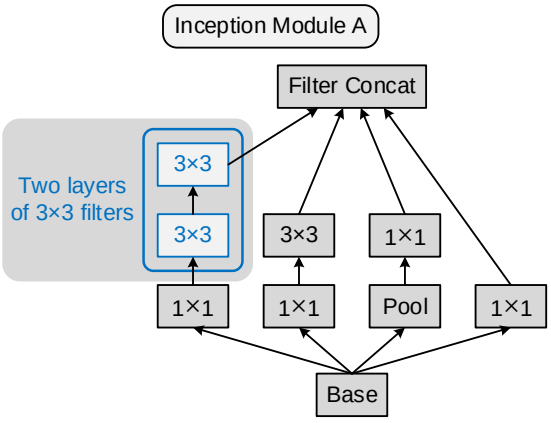

Figure 4. Inception Module A using factorization.

The purpose of using Inception blocks is to reduce the number of connections/parameters without decreasing the network efficiency. The architecture of the Inception-v3 model is shown in Figure 3, where:

- Inception Module A that used filter layers to adjust the image for a smaller size, this used two layers of 3×3 filters (Figure 4) instead of on layer of 5×5 traditional filter and the result was number of parameters is reduced by 28%.[4]
- Similarly, Inception Module B used one layer of 3×1 filter followed by one layer of 1×3 filter (Figure 5) replaces one layer of 3×3 traditional filter and the result was number of parameters is reduced by 33%.[4]

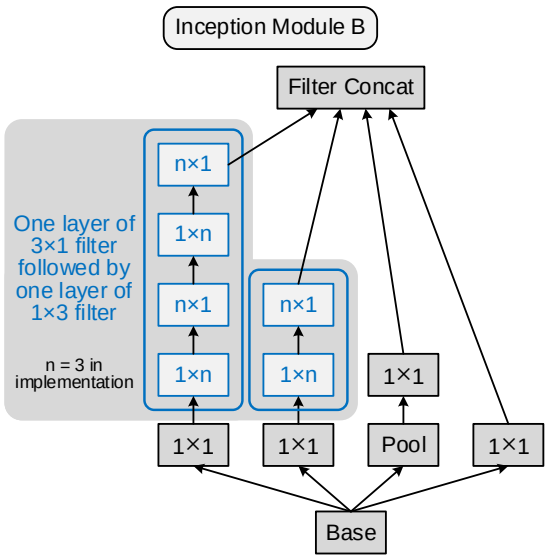

Figure 5. Inception Module B using asymmetric factorization.

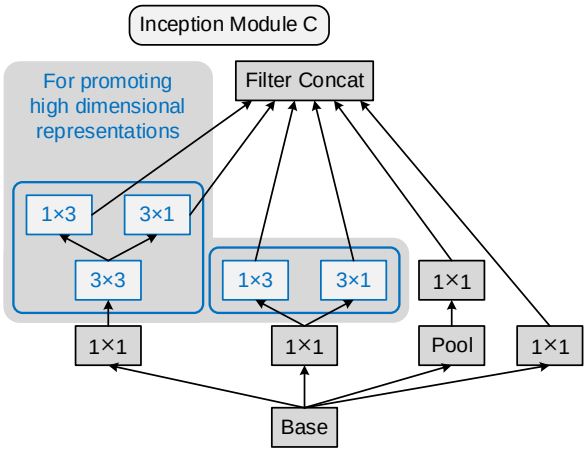

Figure 6. Inception Module C using asymmetric factorization.

- And Inception Module C is also proposed for promoting high dimensional representations (Figure 6) according to the description by Szegedy et al.[4]

Table 1. The output and input sizes of Inception-v3 model.

Type	Patch size/stride	Input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/1	71×71×80
conv	3×3/1	35×35×192
3×Inception A	As in Figure 4	35×35×288
5×Inception B	As in Figure 5	17×17×768
2×Inception C	As in Figure 6	8×8×1280
Pool	8×8	8×8×2048
Linear	logits	1×1×2048
Softmax	classifier	1×1×1000

Table 1 describe in the sizes across each layer of Inception-v3 model. There are three traditional inception modules of Inception Module A at the 35×35 with 288 filters each as depicted in Figure 4. This is reduced to a 17×17 grid with 768 filters using the grid reduction technique. Inception Module A is followed by five instances of Inception Module B as depicted in Figure 5. This is reduced to a 8 × 8 × 1280 grid with the grid reduction technique. At the coarsest 8 × 8 level, there are two Inception modules as depicted in Figure 6, with a concatenated output filter bank size of 2048 for each tile.

2.3. Object recognition method

In this study, we used the Inception-v3 model and Tensroflow 1.9.0 library on the Raspberian operating system running on Raspberry Pi 3B+.

By applying the Tensorflow library[10] and using the Inception-v3 model described above to perform object recognition. This classifier is trained to image classification and then predict the accuracy of the input image. This classifier is considered as a function that takes some data as inputs (object images) and assigns a value prediction (*Score*) to it as outputs. This is done by an automatic technique called supervised learning. Generally speaking, this technique begins with the following few standard steps:

- Step 1 Start with sample image processing.
- Step 2 is to use the input images to train the classifier using pretrained models and learning algorithms.
- Step 3 is finding patterns in the trained images and then predicting the accuracy levels (*Score*).

Figure 7 shows a canonical data model (CDM) of the three steps in this study. This figure shows the data entities and their relationships in the developed machine learning workflow. Also, Python version 3.7 was used in this study to develop this image classifier by implementing the Inception-v3 model in TensorFlow.

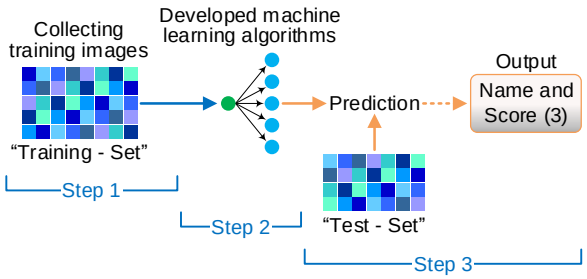

Figure 7. Canonical data model of the developed machine learning workflow.

Firstly, the size of the input image was cropped to 299×299 pixels which is the region of interest, in order to be compatible with the ImageNet database, where the Inception-v3 model was created and trained. Subsequently, the image data is fed to the Inception-v3 model to extract and classify from which the result is the object name and the number of correct predictions in the input image.

The number of predictions (*Score*) is calculated by the following formula:[11]

$$Score = \frac{P_{true}}{\sum P_{made}} \quad (3)$$

where:

P_{true} is number of corret predictions

P_{made} is total number of predictions made.

To perform the recognition, this study installed TensorFlow on a Raspberry Pi 3B+ and ran a simple image classification on the pre-trained Inception-v3 model.

We used the following basic command to install Tensorflow:

```
pip3 install --user tensorflow
```

Then, we have downloaded its 1.9.0 model with the following command:

```
gitclone https://github.com/tensorflow/models.git
```

After downloading, we access the path *models/tutorials/image/imagenet* with the following command:

```
cd models/tutorials/image/imagenet
```

Figure 8. The input image (the desk).

This sutdy ran the python3 program called *classify image.py* to classify the image with the input image (the desk) as shown in Figure 8. This will put the input image into the neural network, which returns predictions about what the image is (the object name) with its number of predictions (*Score*) as shown in Figure 9.

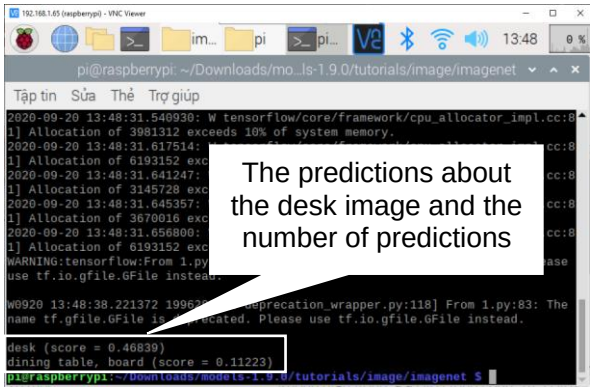

Figure 9. Object recognition program results.

As Figure 9 shows, the neural network predicted correctly, with *Score* = 46,839%. It also gives the result of possible a *dining table*, but has a smaller value than *Score* = 11,223%.

2.3. Develop the program

From the basis of the Inception-v3 model above, we have build an object recognition program with the size of the input image as 299×299, standardized according to the object recognition process. This study presents a state diagram of image classification for recognizing objects as shown in Figure 10.

Initially, the program will initiate the initial conditions. This process takes 5-7 seconds. After the program has finished the initialization process, the program is ready to go. Meanwhile, the program will wait for the user to press the shutter button to operate. When the user presses the shutter button, the program will notify the sound and take picture. This process takes about 2 seconds. After the image is captured, the image data is fed into the Tensorflow library for processing, and the Inception-v3 model performs image classification.

The object name and *Score* will be exported as text data. Only those with the highest *Score* will be output. In this study, we take up to 5 objects with the highest *Score*. After that, the program will output the sound of the object's name in Google Vietnamese voice. The program will stop working when the user presses the OFF button.

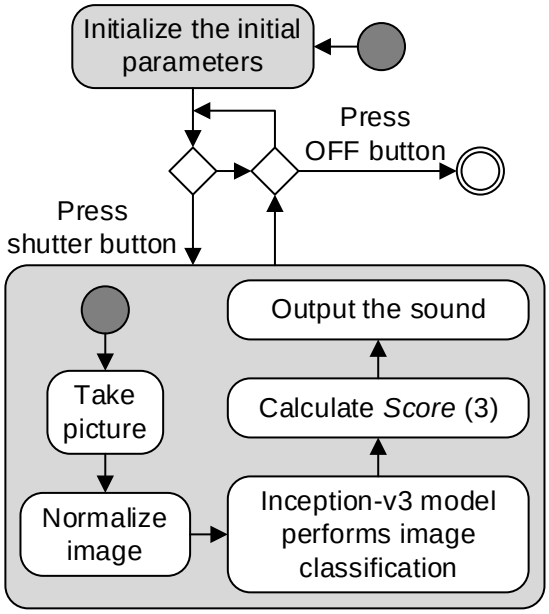

Figure 10. State diagram for recognizing objects.

The state diagram in Figure 10 will be coded in python running on Raspberian (the operating system of Raspberry Pi 3B+).

2.4. Architecture of recognition device

The object recognition model using the Raspberry Pi 3B+ is depicted in the diagram in Figure 11.

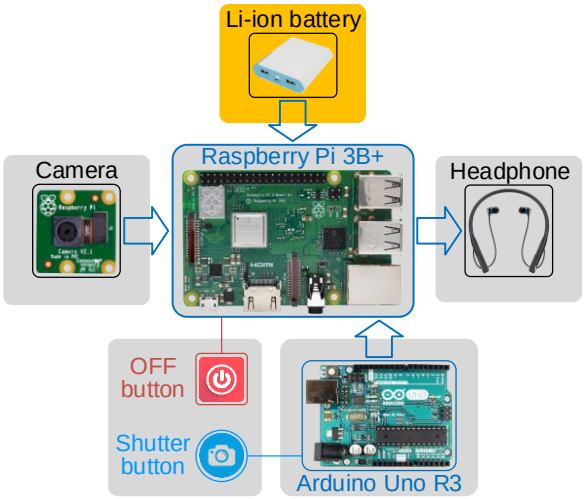

Figure 11. Device overview block diagram.

In which, Raspberry Pi 3B+ is a small, low-cost, energy-saving single computer board. It is the device's processing center with the function of image processing, analysis and classification. It runs on the Raspberian operating system. In addition, the Arduino Uno R3 has the function of

connecting to the shutter button and UART communication with the Raspberry Pi 3B+. Moreover, the Arduino Uno R3 is also used to develop device later, depending on the needs of the field, there will be suitable sensors. The Raspberry Pi camera is used to receive images that are transferred to the Raspberry Pi 3B+ by the FFC/FPC 15 pin cable. It uses 5MP resolution OV5647 sensor. It has good picture quality, high resolution with HD quality. This is suitable for practical application. And headphone (speaker) are used to bring audio signals from Raspberry Pi 3B+ out, users can also use Bluetooth headphones. The power supply of the device is a Li-ion battery pack with a voltage of 5V, it has a compact design and large capacity. This study has conducted experiments and results show that the operating time of this source is about 5 hours of continuous operation. Device model after complete assembly as shown in Figure 12.

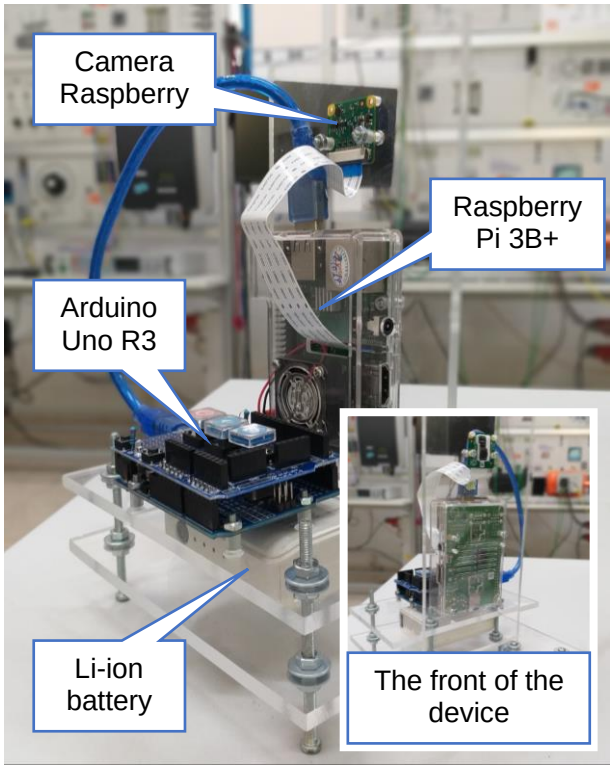

Figure 12. Model of object recognition device.

2.5. Experimental results

After the hardware and software of the device is completed as above. The authors conducted 3 experiments. In each experiment, the authors placed 3 random objects on the table and performed manipulations for the device to identify. The operation steps include:

- Step 1: Point the Camera at the object to be recognized.
- Step 2: Press the shutter button to take pictures.
- Step 3: Wait for the device to output the sound which is the name of the recognized object in Vietnamese.

When the device has recognized the object, the screen (it connected to the Raspberry Pi 3B+ via HDMI) also displays the text of the names of the recognized objects and is accompanied by a value *Score* which indicates accuracy of recognition results. The total Score of all recognized objects is 1. On a multi-object frame, objects are large and clearly have a *Score* value of quite high, and the remaining objects are small with a *Score* value lower.

2.5.1. Experiment 1

In experiment 1, the authors performed the 3-object recognition device: corn, banana, and flowerpot. In which, Figure 13 shows the implementation process. Figure 14 shows the images and results recognized by the device. Figure 15 shows the output of a Python program on a Raspberry Pi 3B+.

Figure 13. Conducting recognition in experiment 1.

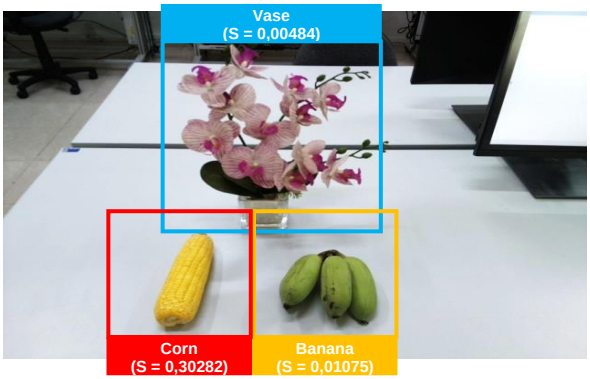

Figure 14. Recognition results of experiment 1.

As shown in Figure 15, the program has been recognized ear (*Score*=0.58129), corn (*Score*=0.30282), banana (*Score*=0.01075), vase (*Score*=0,00484).

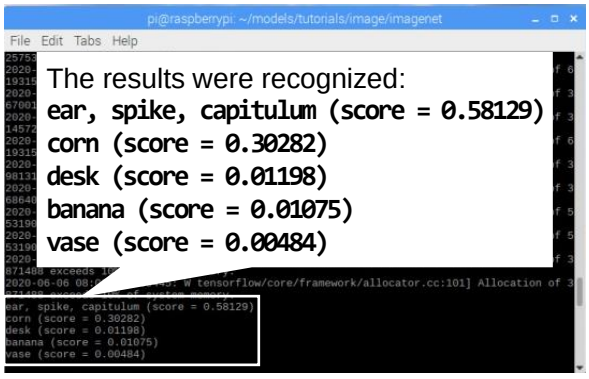

Figure 15. Recognition results in experiment 1 were received on a screen connected to a Raspberry Pi 3B+.

Next, the program uses the Trans library in the Python programming language. This is to convert the above object name text results (English) into Vietnamese. At the same time, the device uses the gTTs library to convert text into speech, here is Google Vietnamese voice. After, the device converts the text to audio, the audio file will be saved and played to the user. The process of converting from English results to written audio is similar for the other 2 experimental results. Finally, the device converts the text to audio, the audio file will be saved and played to the user. The process of converting from English results to audio is similar for the other 2 experimental results.

2.5.2. Experiment 2

In experiment 2, the authors performed recognition of sharp objects with 3 subjects, including: meat cleaver, screwdriver and water bottle. Implementation process and experimental results are shown in Figure 16, Figure 17 and Figure 18.

The results returned as expected, the device recognized it as meat cleaver (*Score*=0.39394), screwdrivers (*Score*=0.04013), water bottle (*Score*=0.23964). This makes the device available in situations where hazardous objects are identified.

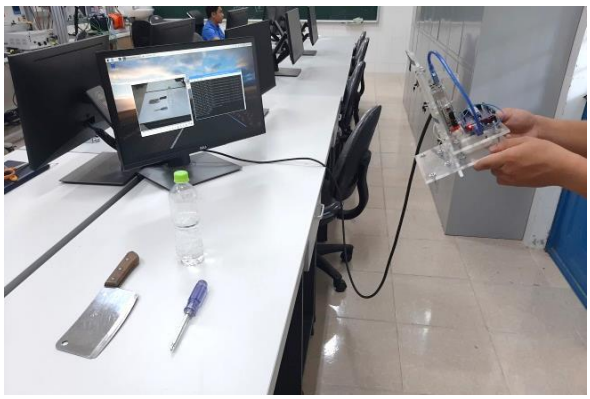

Figure 16. Conducting recognition in experiment 2.

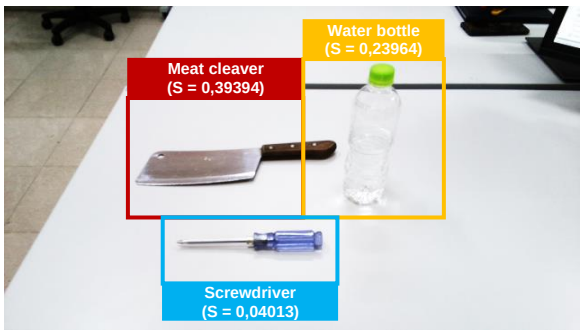

Figure 17. Recognition results of experiment 2.

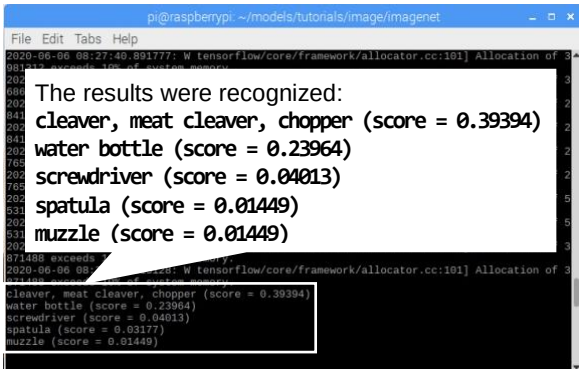

Figure 18. Recognition results in experiment 2 were received on a screen connected to a Raspberry Pi 3B+.

2.5.3. Experiment 3

In experiment 3, the authors conducted 3 objects for the object recognition device, including: monitors, computer mice and computer keyboards. Implementation process and experimental results are shown in Figure 19, Figure 20 and Figure 21.

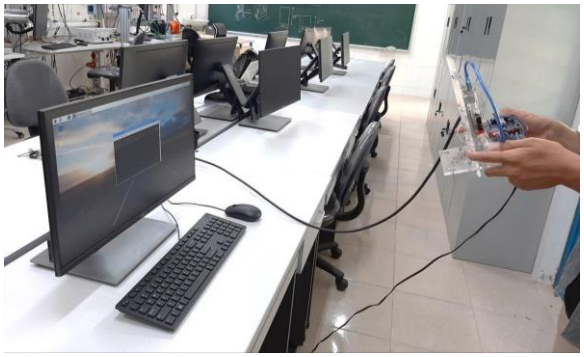

Figure 19. Conducting recognition in experiment 3.

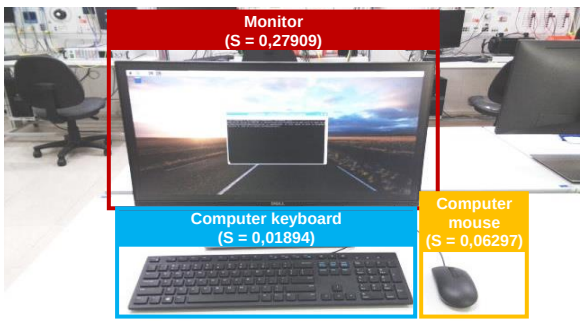

Figure 20. Recognition results of experiment 3.

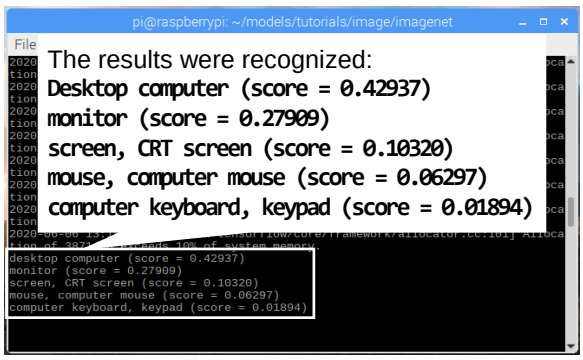

Figure 21. Recognition results in experiment 3 were received on a screen connected to a Raspberry Pi 3B+.

The results returned as expected, the device recognized that it was computer mouse (*Score*=0.06297), screen (*Score*=0.27909), computer keyboard (*Score*=0.01894). We finish the experiment.

After 3 times of experiments showing that the device is capable of identifying objects with high reliability, the recognition results are also highly accurate.

3. CONCLUSIONS

The study presented the object recognition study using Raspberry Pi 3B+ and the Inception-v3 model. The result is a device that is compact and portable with simple usage, suitable for everyone. The experimental results in this study show that the names of objects recognized have a relatively high accuracy, along with a relatively high predictive value. However, the device's response time is slow, which can be improved by better hardware devices. With the above results, this device can be used in many different fields such as health, society, education... depending on the purpose of use. In addition, the device also combines with different sensors to create a specific device according to the needs of the user.

Acknowledgement: *This study is conducted within the framework of the student scientific research project for the academic year 2019-2020 under the project code S2019.597.34.*

REFERENCES

1. Nilima Sahoo, Hung-Wei Lin, Yeong-Hwa Chang. Design and Implementation of a Walking Stick Aid for Visually Challenged People, *sensors MDPI*, **2019**, *19*(130), 17.
2. Wing Yip, Zory Stoev. Determining success of the OrCam MyEye/MyReader in patients with visual impairment, *Investigative Ophthalmology & Visual Science*, **2017**, *58*(8), 3271.
3. X. Xia, C. Xu, B. Nan. *Inception-v3 Flower Classification*, 2017 2nd International Conference on Image, Vision and Computing, Chengdu, China, 2017.
4. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna. *Rethinking the Inception Architecture for Computer Vision*, Computer Vision and Pattern Recognition 2016, Cornell University, 2016.
5. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. Dropout: A SimpleWay to Prevent Neural Networks from Overfitting, *Journal of Machine Learning Research*, **2014**, *15*(56), 1929−1958.
6. M. D. Zeiler, G. W. Taylor, R. Fergus. *Adaptive deconvolutional networksformid and high level feature learning*, 2011 IEEE International Conference on Computer Vision, Barcelona, Spain, 2011.
7. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. *Going Deeper with Convolutions*, The IEEE conference on computer vision and pattern recognition, Cornell University, 2015.
8. P. Ramachandran, B. Zoph, Q. V. Le. *Searching for activation functions*, 6th International Conference on Learning Representations, London, United Kingdom, 2018.
9. M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. V. Essen, A. A. S. Awwal, V. K. Asari. A State-of-the-Art Survey on Deep Learning Theory and Architectures, *Electronics (Switzerland)*, **2019**, *8*(3), 1-67.
10. N. R. Gavai, Y. A. Jakhade, S. A. Tribhuvan, R. Bhattad. *MobileNets for flower classification using TensorFlow*, 2017 International Conference on Big Data, IoT and Data Science, Pune, India, 2018.
11 Omar Albatayneh, Lars Forslöf and Khaled Ksaibati. *Image Retraining Using TensorFlow Implementation of the Pretrained Inception-v3 Model for Evaluating Gravel Road Dust*, The ASCE American Society of Civil Engineers, 2020.