method is the initial division of the image into a grid of cells. In addition, each cell defines the class of the object in the area related to this cell[7]. The main idea of this approach is to define an object as a problem of regression to spatially separated bounding blocks and associated class probabilities[8].

The Single-Shot MultiBox Detector method is represented by two components: a neural network for image classification and a convolutional layer for detecting and classifying objects in the image. This method, like YOLO, divides the image into a grid, but also operates with the concept of an anchor region[9]. Multiple anchor regions can be assigned to each grid cell. Each of them is determined and responsible for the size and shape of the object inside the cell.

The SSD method is used to describe architectures that use a single convolutional neural network to directly predict the location of regions and their classes, without applying a second stage of classification. In this entire method, at the output of the neural network, several thousand forecasts are formed for the possible regions of the location of objects of different shapes at different scales, then with the help of suppression of non-maximums, a selection of several most likely areas is made. Such a single structure, simultaneously taking into account different image scales, provided the SSD method with the highest indicators in terms of speed and quality of object detection compared to other modern approaches[10].

The operation of the SSD method is based on a fixed set of rectangles, which check the presence of an object on each of them[11]. Suppose we have some m×n feature map, which is obtained from one of the convolutional layers of a neural network. Let's go through it with a convolution with a 3×3 kernel, which at the output produced 4+Cl channels, where Cl is the number of classes.

That is, we divide our image with a grid, because each feature at the output of the convolutional layer absorbs information about the pixels of a square in the original image and therefore can detect an object located in this square. The earlier layer we use to extract the feature map, the larger its size (i.e., m and n) and the smaller objects we will be able to detect.

Unlike R-CNN, where there is at least a minimal probability of finding an object in candidate regions, there is no region filtering step in SSD[12]. As a result, a much larger number of describing rectangles at different scales is generated compared to R-CNN, and most of them do not contain an object. In order to solve this problem in SSD[13], firstly, the suppression of non-maximums of combining similar rectangles into one is used. Secondly, the hard negative mining technique is used, according to which only a part of negative examples is used in each iteration of training, in SSD the ratio of the number of negative examples to positive is 3 to 1.

For each object marked on the image, there can be several predictors from whom we are ready and want to get a description of the object. Let's introduce an indicator function that is equal to one if the i-th anchor has an IoU greater than 0.5 with the j-th object in the image, and zero if not.

The overall target loss function is the weighted sum of localization loss (loc) and confidence loss (conf) (1):

$$L(x,c,k,g) = \frac{1}{N}(L_{conf}(\overline{x},c) + \alpha L_{loc}(x,l,g)), \quad (1)$$

where N is the number of matching blocks by default. If $N = 0$, wet sets the loss to 0. The localization loss is the smooth $L1$ loss between the predicted box parameters (l) and the ground truth field (g). We regress to offsets for the center (cx; cy) of the default frame (d), as well as for its width (w) and height (h).

L_{class} - is responsible for the correct definition of the class of the object and is summarized by many anchors.

L_{loc} - is the sum of all anchors to which the object is mapped, i.e., $X_{ij} \neq 0$ and fines for errors in defining the rectangle of the object are summed up.

If we have chosen m layers, then evenly spreading from the scales we get (2):

$$S_k = S_{min} + \frac{S_{max} - S_{min}}{m-1}(k-1). \quad (2)$$
$$k = 1, \cdots, m$$

Also, for each scale, in order to choose not only square anchors, we will set a set of aspects (3):

$$A = \{a_r | r = 1, \cdots, T\} \quad (3)$$

Then the dimensions of the anchors will be calculated according (4):

$$w_k^r = S_k\sqrt{(a_r)}, \quad (4)$$
$$h_k^r = \frac{S_k}{\sqrt{(a_r)}}$$

Anchor positions are chosen simply. If we attach squares to a layer with a feature of size m×n, then the centers of the squares will be at the points (5):

$$\left(\frac{i+0.5}{m}\right), \left(\frac{j+0.5}{n}\right). \quad (5)$$
$$i = 0.1, \cdots, m; j = 0.1, \cdots, n$$

The SSD architecture is the most suitable for real-time image processing (especially when using Mobile Net networks)[3][13], but it must be taken into account that high accuracy requirements cannot always be met. This method has the following advantages:

- the speed of determining the object in real time;
- one pass through a convolutional neural network;
- searching for objects on layers, which does not degrade the quality of the input image. That is, more bins by default results in more accurate detection, although it affects speed;
- having MultiBox on multiple layers also results in better detection due to the fact that the detector works with multi-permit features;
- SSD mixes objects with similar categories (e.g., animals). This is probably because the seats are shared by several classes;
- the SSD-500 (the highest resolution variant using 512×512 input images) achieves the best MAP on Pascal VOC2007 at 76.8%, but at the expense of speed, where its frame rate drops to 22 fps. The SSD-300 is thus a better compro-mise, with 74.3 mAh at 59 fps.

3 CONSTRUCTION OF THE OBJECT DETECTION ALGORITHM ON THE RASPBERRY PI PLATFORM

The analysis of existing object detection methods allowed us to highlight the main drawback: the SSD method gives the worst performance for small objects, because they may not be displayed on all object maps. Increasing the resolution of the input image alleviates this problem, but does not completely solve it.

To implement the task, the Raspberry Pi platform was used. To implement object detection, the SSD300 method was used, whose algorithm was described above. Since the recommended input image size for this method is 300×300, a camera with the appropriate image resolution was chosen.

Also, let me show you a diagram of the algorithm for detecting objects on the Raspberry Pi platform. The algorithm of this method is shown in Figure 1.

Step 1. We take the image out, if the object is in the camera's field of view.

Step 2. We pass the original image through a series of convolutional layers and obtain a set of feature maps for different scales.

As a result of the analysis of the existing methods and the analysis of their advantages and disadvantages, we choose the SSD300 method for the implementation of the system. The main advantage of this version of the method is that the input image is reduced to the size of 300×300 pixels. Which in turn improves the processing time of this image[13].

Step 3. We apply a 3×3 convolutional filter to each point of each feature map to obtain rectangles.

Step 4. For each rectangle, the spatial displacement probability of finding the object is simultaneously estimated

Step 5. In the learning process, the real descriptive objects of the rectangles are compared with the predicted ones to exclude late detections.

Step 6. We compare the object class in the image with the object class of the training model.

Step 7. Combine all layers of the image and suppress non-maximums.

Step 8. We display the result.

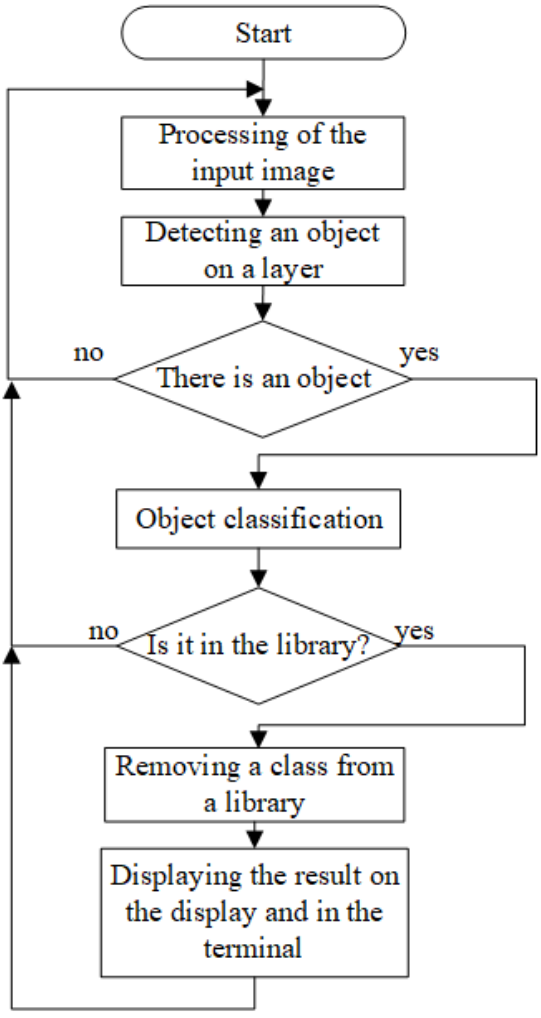

Figure 1: Block diagram of the object detection algorithm.

4 OBJECT DETECTION SYSTEM ON RASPBERRY PI

Based on the algorithm described above, we will develop an object detection system using the Raspberry Pi platform.

In order to test the theories regarding the effectiveness of the methods of detecting objects defined above, it is necessary to conduct a research experiment. For this, a system of detecting objects using the proposed method was developed.

The main functional capabilities of the system are:

- detection of the centers of coordinates of the position of the object;
- object classification;
- detection of the result of system operation.

The SSD method is implemented in the Python 3 programming language using the TensorFlow[14] and Keras deep learning libraries.

This system consists of 3 elements: camera, Raspberry Pi and display. During the execution of the program, the camera constantly reads the image on the video stream and processes it. It transfers the received results to the terminal on Raspberry Pi, it is shown in Figure 2.

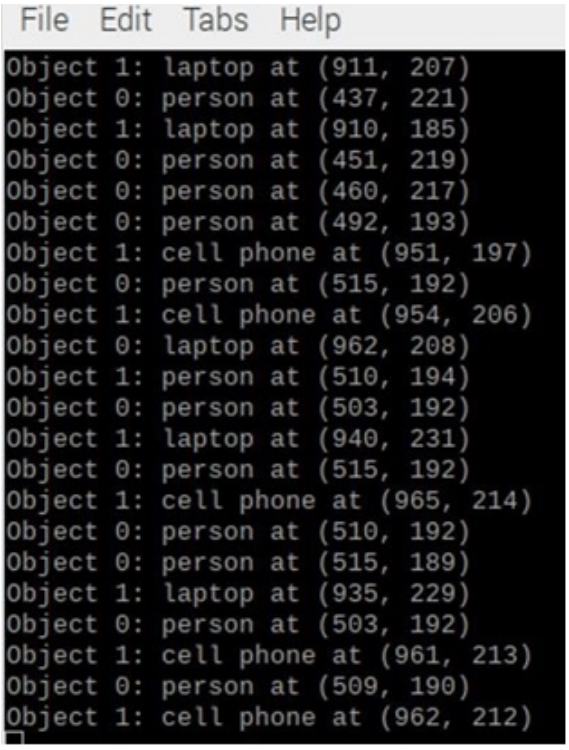

Figure 2: The result of executing the program in the terminal on Raspberry Pi.

At the same time, it is transmitted to the display, which is shown in Figure 3.

Figure 3: The result of program execution on the display.

5 CHECKING THE EFFICIENCY OF THE PROPOSED SOLUTIONS

To evaluate the quality of object detection, such metrics as the measure of the intersection of the found and reference rectangles that contain objects (Intersection, I), completeness (Recall, R) and accuracy (Precision, P) of object detection are used.

The measure of the intersection of the found and reference rectangles I (6) show how clearly the convolutional neural network found rectangles relative to the rectangle of the reference marking.

$$I = \frac{S_i}{S_j + S_{gt} - S_I}, \tag{6}$$

where S_i - the area of intersection of the true and calculated rectangle;

S_j - the area of the rectangle found by the algorithm;

S_{gt} - the area of the reference rectangle (grounded truth).

The completeness of R (7) shows the sensitivity of the algorithm to errors of the 2nd kind, that is, omissions, and is equal to the ratio of the number of correctly found objects to the number of these objects in the reference markup.

$$R = \frac{t_p}{t_p + f_n}, \tag{7}$$

where t_p - true positives are those objects expected to be seen and received at the output;

f_n – false negatives – objects that we expected to see, but the algorithm did not identify them (misses).

Accuracy P (8) shows the sensitivity of the algorithm to errors of the 1st kind, i.e., false positives, and the ratio is equal to the number of correctly found objects to the total number of rectangles found by the algorithm.

$$P = \frac{t_p}{t_p + f_p}, \tag{8}$$

where f_p - false positives are objects that should not be at the output, but the algorithm erroneously returned them at the output (false activations).

To test the effectiveness, we will take 3 methods:

- the YOLO method;
- the SSD300 method;

The above methods have been tested using a camera and a computer.

- the SSD300 method created on the Raspberry Pi platform (SSD300_Raspberry).

This method has been developed and implemented on the Raspberry Pi platform. The same camera as in the previous methods was used to obtain the image.

The parameters for evaluating efficiency are calculated according to formulas 6-8.

During the experiment, 100 images were processed. To construct graphs, the threshold coefficient in the object detection algorithm is changed in the range from 0 to 1 with a step of 0.01. The threshold coefficient is the minimum value of the probability estimate at which the decision to detect the object will be made. Figures 4-6 show the graphs of dependencies of accuracy, completeness, and crossing measure on the given threshold.

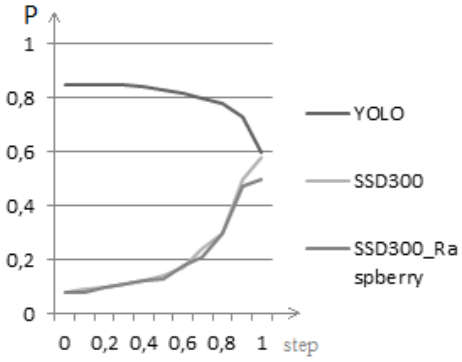

Figure 4: Dependencies of the accuracy (P) of the methods: YOLO, SSD300 and SSD300_Raspberry from the given threshold.

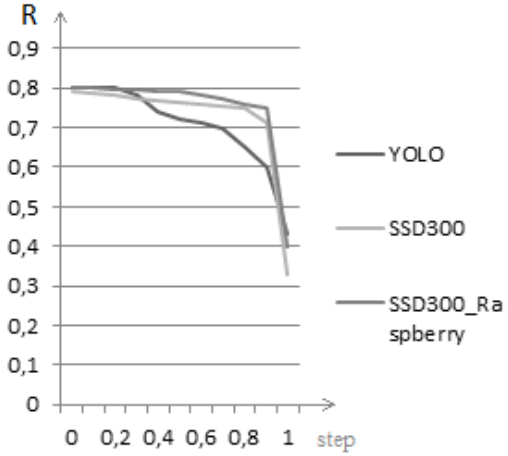

Figure 5: Dependencies of the completeness of the activation (R) of the methods: YOLO, SSD300 and SSD300_Raspberry from the given threshold.

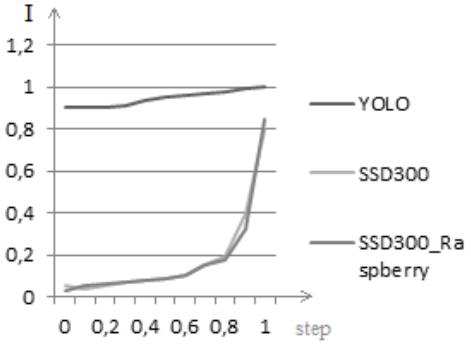

Figure 6: Dependencies of the measure of the intersection of the found and reference rectangles (I) when using the methods: YOLO, SSD300 and SSD300 Raspberry from the given threshold.

To assess the quality of the system, the mAP (mean average precision)[15] metric is calculated, which is the average value of AP across all classes of objects. AP (Average Precision)[15] is the average value of the maximum precision for different completeness values. The area under the graph of dependence of accuracy on completeness (AUC – area under curve) and mAP were used as integral estimates of the quality of the detectors. The named indicators are presented in the Table 1.

As a result of the assessment of the quality of the system, it was found that the SSD300 Raspberry method works faster by 0.2 ms, and detects objects more accurately by 3% than the SSD method on a computer. However, compared to the YOLO method, it is slower by 20 ms, but the accuracy is higher by 8.7%. From this, it can be concluded that the system

on the Raspberry Pi platform is better suited for the implementation of a system that recognizes objects in real time.

Table 1: Performance test results.

Method	Characteristics		
	AUC	mAP, %	Time, ms
YOLO	0.882	66.4	76
SSD300	0.573	72.1	58
SSD300_Raspberry	0.534	75.1	56

6 CONCLUSIONS

We hope you find the information in this template useful in the preparation of your submission.

Detection of objects in real time is a complex task that requires further improvement of existing or creation of new methods for the implementation of this task. Using the SSD method, we have an advantage in the speed of work, but the disadvantage is the quality of object classification.

In the course of the research, already existing methods of detecting objects in the image were analyzed. Based on the results of the analysis, an opinion was formed about the further development of this topic. Based on the developed methods and algorithms, with the help of tools and tools of Python programming technology and the Raspberry Pi platform, a real-time object detection system was designed. Recommendations have been developed to improve the operation of the methods, due to the improvement of technical equipment. To conduct a study of the effectiveness of the methods of detecting objects on the image, a comparative analysis was performed using an integral assessment of the quality of the detectors, using the area under the graph of the dependence of accuracy on completeness and mAP. During the comparison, integral indicators were invented that confirm the effectiveness of the developed accounting methods.

REFERENCES

[1] D. H. Ballard and C. M. Brown, Computer Vision, New Jersey, USA: Prentice Hall, 1982.

[2] S. Dasiopoulou, V. Mezaris, I. Kompatsiaris, V.-K. Papastathis, and M. G. Strintzis,"Knowledge-assisted semantic video object detection," IEEE Trans. Cir. Syst. for Video Techn., vol. 15, no. 10, pp. 1210-1224, 2005.

[3] Y. Jang, H. Gunes, and I. Patras, "Registration-free face-ssd: Single shot analysis of smiles, facial attributes, and affect in the wild," Comp. Vis. Image Und., vol. 182, pp. 17-29, 2019

[4] M. Babichev and V. Lytvynenko, Eds., Lecture Notes in Data Engineering, Computational Intelligence, and Decision Making. Boston, MA: Springer Cham, 2022, pp. XVII, 721.

[5] G. Stockman and L. G. Shapiro, Computer Vision. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001

[6] . Jiang, L. Zhao, S. Li, and Y. Jia, "Real-time object detection method based on improved yolov4-tiny," arXiv preprint arXiv:2011.04244, 2020

[7] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in Twenty-second int. joint conf. on AI. Citeseer, 2011.

[8] H. Zhou, F. Jiang, and H. Lu, "Ssda-yolo: Semisupervised domain adaptive yolo for cross-domain object detection," Comp. Vis. and Im. Und., vol. 229, p. 103649, 2023

[9] M. A. Feroz, M. Sultana, M. R. Hasan, A. Sarker, P. Chakraborty, and T. Choudhury, "Object detection and classification from a real-time video using ssd and yolo models," in Comp. Int. in Patt. Rec.: Proc. of CIPR 2021. Springer, 2022, pp. 37–47

[10] S. Kanimozhi, G. Gayathri, and T. Mala, "Multiple real-time object identification using single shot multibox detection," in 2019 ICCIDS. IEEE, 2019, pp. 1-5.

[11] K. Wadhwa and J. K. Behera, "Accurate real-time object detection using SSD," Int. Res. J. of Eng. and Techn., vol. 7, no. 5, 2020.

[12] X. Xie, G. Cheng, J. Wang, X. Yao, and J. Han, "Oriented r-cnn for object detection," in Proceedings of the IEEE/CVF Int. Conf. on Comp. Vis., 2021, pp. 3520-3529.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in Computer Vision–ECCV 2016: 14th Eur. Conf., Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer, 2016, pp. 21-37.

[14] O. Ferm, "Real-time object detection on Raspberry Pi 4: Fine-tuning a SSD model using tensorflow and web scraping," 2020.

[15] L. Liu and M. T. Özsu, Eds., Mean Average Precision. Boston, MA: Springer US, 2009, pp. 1703-1703.