DEVELOPMENT AND STUDY OF PERFORMANCE IN RELATION TO MOBILE COMPUTER VISION MODULE

Mikhail G. Gorodnichev[1], *Nikita M. Bulygin*[2], *Rinat A*[3]. *Gematudinov, Khizar A*[4]. *Dzhabrailov*[5], *Marina S. Moseva*[6]

[1,2,3,4,5,6] Moscow Technical University of Communications and Informatics, Moscow, Russia

[1]m.g.gorodnichev@mtuci.ru, [2]mkiit@yandex.ru, [3]rinatg86@mail.ru.ru, [4]hizarmuslim@mail.ru, [5]mmoseva@yandex.ru

Mikhail G. Gorodnichev, Nikita M. Bulygin, Rinat A. Gematudinov, Khizar A. Dzhabrailov,Marina. S. Moseva. Development And Study Of Performance In Relation To Mobile Computer Vision Module-- Palarch's Journal Of Archaeology Of Egypt/Egyptogy 17(3), 653-666. ISSN 1567-214x

Keywords: Computer Vision, Cluster Systems, Neural Networks, System Performance, Distributed Programming.

ABSTRACT:

In the current era of rapid IT development and computerization of society, highly effective systems for large data analysis play an important role in various spheres of human life. The constantly-growing volume of data, as well as its rapid modification in order to improve the efficiency of use, create the necessity in modernization of existing methods and development of new approaches to data processing. In this paper, we study algorithms and methods of concurrent programming, and load distribution with the maximum use of available computing resources. In this work, in accordance with various design architectures of convolutional neural networks, we prepared the software for image recognition tasks with load distribution of the cluster system for the single-board computers "Raspberry pi". In this article, we describe the software testing in various situations.

INTRODUCTON

The processing speed is a property that was always an important part for the different fields of professional work. The response rate often determines the value of the result for specific tasks, after all a negative result is also a result. In order to increase the processing speed in various fields, one uses the machines and mechanisms that are widely popular in our time. A machine, such as a

computer, can perform actions that the developer has put in it at most in the form of various algorithms. There is a problem when instance processing lacks speed, because the machine does not "know" how to respond to it, as such behavior is not part of its code. A computer is an auxiliary device which has similar features of functioning with the human brain and function to help it. The key difference between a person and a computer is the feature of self-sufficiency. The display of human self-sufficiency is the ability to self-learning. The solution to the problem of machine processing speed through the "acquisition" of the ability to learn comes with the occurrence of machine learning technology – a method that implies not a direct solution to the problem, but the development of the program in the process of many similar problems processing.
One uses the image recognition techniques in a variety of areas, such as security and control systems, in-depth study and forecast in medicine, ip telephony, marketing, entertainment, and support for people with disabilities. The image recognition bases on the machine learning. In the process of self-learning, the machine "learns" to solve problems by the difference in images and their further classification. There are many methods of education like with a teacher, without a teacher, etc. One uses each of them in the areas that correspond to the advantages of a particular learning model. Three main properties characterize the various image recognition methods – taught hours, accuracy and speed. There are also other properties that are equally important, such as an object classification error, etc.

The methods of data exchange within the module

In order to ensure fast data transfer within the device and between the boards, one needs to develop a data exchange scheme. The Mosix method directs the migration of the processes to the cores of a single local processor, as well as the migration to the cores of processors that locate both on the local and remote devices. In order to maintain the migration, mosix uses TCP/IP to transmit messages over the network. There will be the message transfer over the network by a channel between the cluster boards created with a switch-device and RJ45 cables. The bandwidth of the boards is 1 Gbit / s, this spectrum is sufficient for TCP/IP messaging between cluster nodes. There are two basic schemes of cluster operation: a scheme with a manager - with a cluster there is a selection of the main device that manages process migration, and the manager is not used as a handler; a scheme only with handlers – in such scheme, one uses all devices in the cluster as data handlers. In view of low compute capacity of the raspberry pi3 B and NanoPi single-Board computers, it is more efficient to use each Board as a processor, without the selection of the main Master machine, which capacity will be without realization. Each node require a static address of the same range in order to ensure the stability of the cluster. The dynamically varying addresses can disrupt the migration process at any time by changing their address and "Drop" from the cluster. In order to ensure the visibility of processing for migration algorithms, you need to run data processing over a convolutional neural network by Mosix itself. Let's consider the working stages of the module. The user starts CNN via Mosix. CNN creates function execution threads and gives them to the CPU processor for calculation.

The processor assigns the execution of a defined process to a specific core **(Figure 1)**

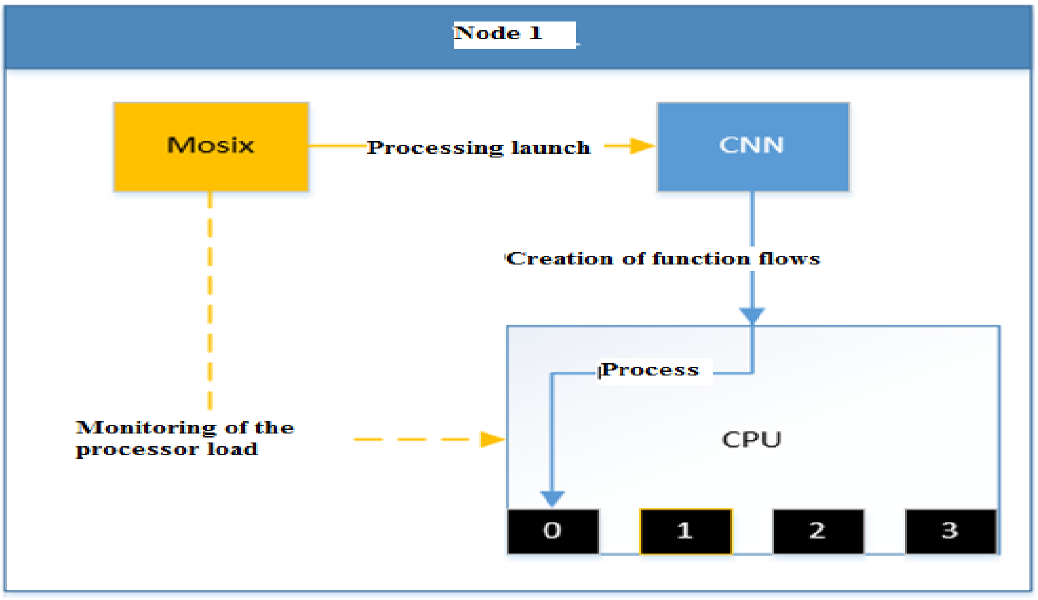


Fig. 1 - Launch of the process module

At the next stage, mosix continuously monitors the processor utilization. After the load on a particular core reaches the limit threshold, mosix performs a process migration algorithm to the core of the same node (figure 2). Mosix constantly monitors all node processors in the cluster, so it can realize a migration to the processor core of a remote node (figure 3).

Figure 3 shows a schematic diagram of the module data exchange, with the designation of each block, board, and auxiliary devices.

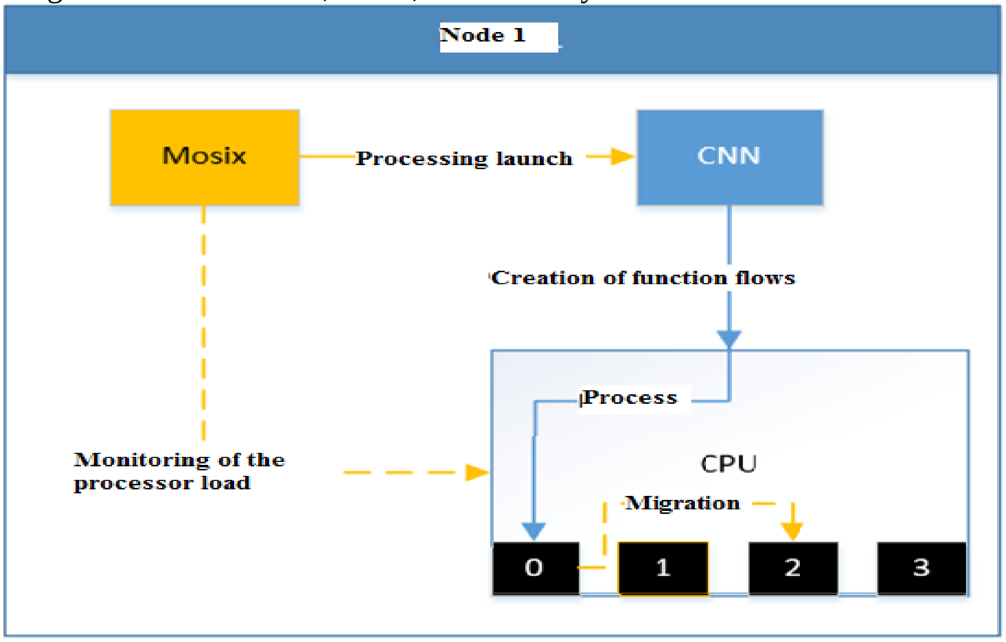


Figure 2 - Process migration

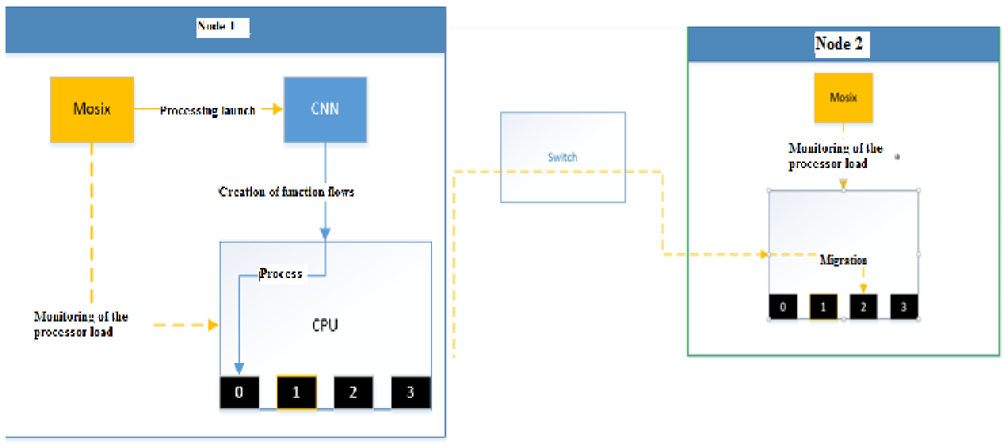

Node 1
Mosix
Processing launch
CNN
Creation of function flows
Process
CPU
Monitoring of the processor load
0
1
2
3
Switch
Node 2
Mosix
Monitoring of the processor load
Migration
0
1
2
3

Figure 3 - Process migration to the processor core of the remote node

Development of software and hardware module for machine vision

In order to use the module, one needs to build a scheme which consists of two single-board computers that will act as nodes in the cluster in this scheme (figure 4). Connect the boards to a switch that will route packets between the nodes. Connect the boards to the Switch via an RJ45 cable. Each board will have its own power supply via a micro-USB cable. Connect the peripheral hardware necessary for operation and launch of the program to the first node: keyboard, mouse, screen, web camera. Figure 3.2 shows the assembled functional diagram.

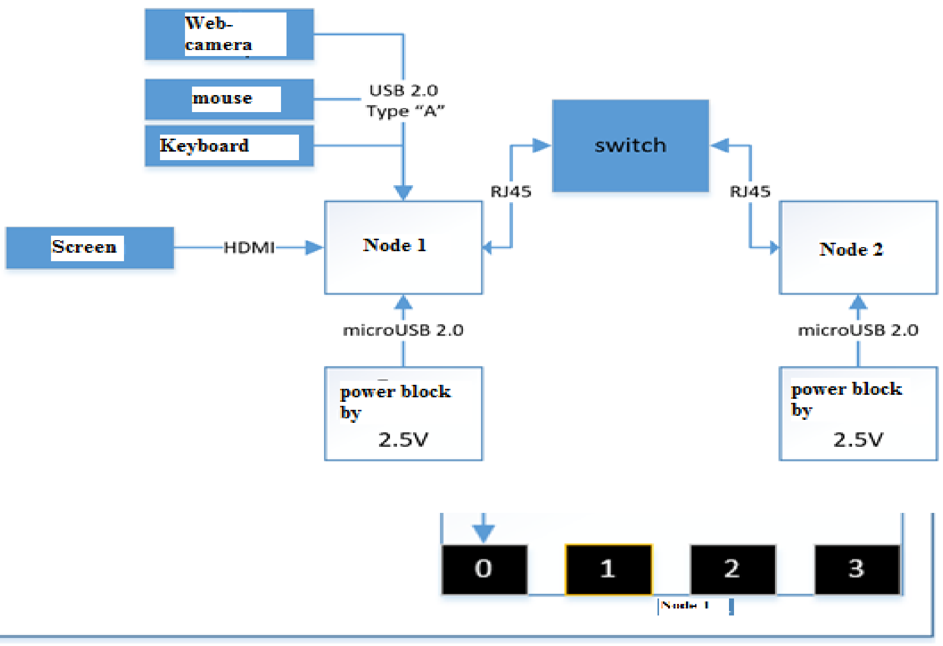


Fig. 4 - Functional diagram of the hardware and software module for machine vision

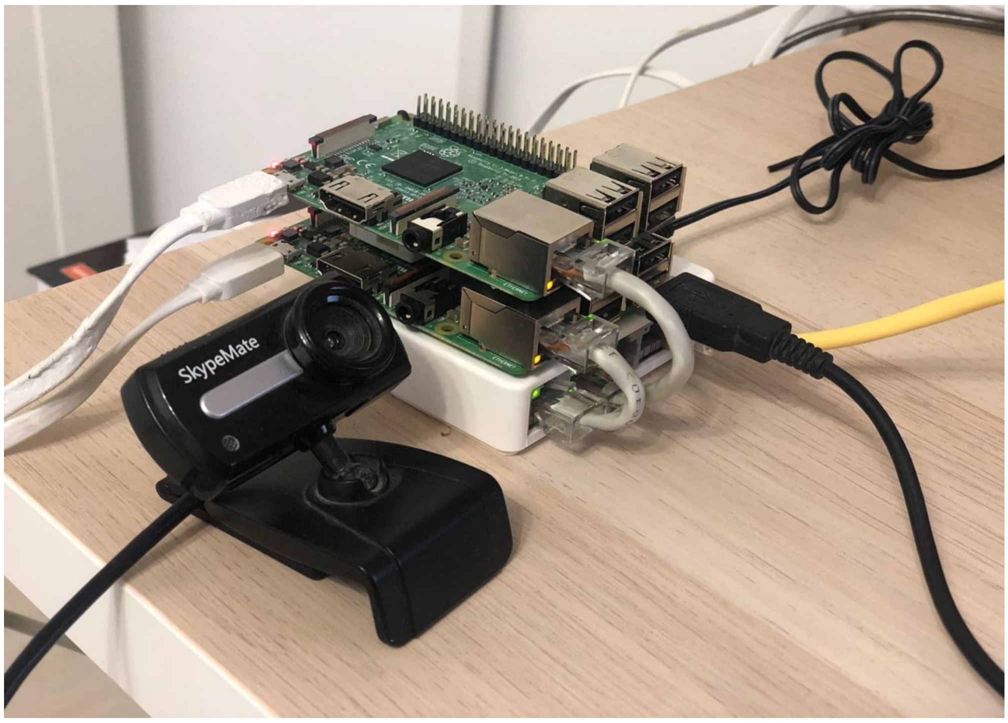


Fig. 5 - Assembled machine vision module

Installation of convolutional neural networks

The installation task of neural networks is the coding that performs image recognition by a defined model. Whereby, it is necessary to parallelize the execution flows of neural network functions in order to ensure the efficient migration of processes between the processor cores of cluster nodes.

You Only Look Once.

Initially, the Yolo Architecture based on sequentially arranged convolution layers. Within the passage through several layers of convolution and the activation function, in this case one uses a sigmoid function, the image converts to a tensor of fixed dimensions 14x14x1024. Further, thanks to the Stride operation, the image descends in N pax, which increases the speed of image processing, at the expense of the next layer input size. After the passage through the convolution layers, the obtained values transfer to fully connected layers, which at the output give a vector, then transformed into a 7x7x30 tensor.

The extreme obtained tensor contains all the necessary information for further detection. One overlays the image with a 7x7 grid, each cell of which contains a part of the image. A vector with the dimension of 30 contains values for two frame windows, for one object with the same center. The first 5 values are the coordinates of the first frame window, the width and height of the first frame window, and the confidence to accurately detect an object inside the frame window. The second five values contain the same information, only for the second frame window. The last 20 values contain the identifiers of the classes.

Then, the sufficiency of each class is under calculation for each framing window. The IoU and NMS operations apply for all reported window sufficient values. The resulting final frame windows have a display on the original image.

In the future, the architecture design obtained the improvements, and this paper we will use the 3rd version, which has 106 convolution layers.
The complex is effective for the tasks of power distribution required for calculation of image detection flows on the input image. But first of all, the convolutional neural network must pass self-learning process. In this paper we organized the learning procedures on CUDA cores by means of GPU compute capacity, since the learning on the CPU would be inefficient in this case.
Each neuron of the network performs various operations of input multiplication by weights, summarization and multiplication by nonlinearity. The name of mode in which the neural network in the stream passes images through neurons is - inference.

After the launch of the inference, the stream from the video camera divides into frames and transfer to the input of the neural network. The Yolo architecture involves only one image passing through the network. Initially, the image marks the regions of interest where the object approximately locates. Each area has a selection frame with a defined class. While passing through a neural network, the Yolo model overlays a grid on the image, and divides it into separate boxes. Each of the generated boxes predicts the coordinates of the detection area with further sufficiency estimation of these fields and the rating probability. Then, one multiplies the sufficiency score for each detected zone by the rating probability. As a result, on can identify a single item by several areas of interest. So then you need to reduce a lot of frames, leaving only one for one object. In order to minimize the detection zones, one uses the method of Non-max suppression (NMS). At the first stage, the NMS discards all boxes where the probability of IoU object detection is less than 0.6. Then, one takes the box with the highest probability among all candidates for the object as a forecast, and then discards any remaining box with an intersection of more than 0.5 with the forecast box. As a result, the NMS method outputs a single cell with the highest probability of object detection.

In order to measure the speed of a convolutional neural network, one uses the parameter of frame per second (FPS). In this way, one can measure the exact number of processed model images by means of CPU power. In order to calculate FPS correctly, one needs three parameters: the number of frames before the counter update, the current time, and the time since the last counter update. From the initial moment of time, the frame count increases until a second of time passes, as a result, the number of frames for the past second displays on the user interface and the counter is reset to zero.

Faster R-CNN.

Unlike Yolo, R-CNN is a two-stage convolutional neural network model. The image is sent to the network input not entirely, but divided into areas of interest by the method of Selective Search. This method produces about 2000 areas of different sizes and aspect ratios, but to further image processing through the neural network, one needs to bring the image to a single format. In oder to determine whether an object belongs to a class, we use the Intersection over Union (IoU) metric, which one defines as the ratio of the intersection area-candidate region with a rectangle which in reality covers the object in relation to the single square of these rectangles. In case of indication in relation to specified threshold overestimation of the defined limits, one can define the area-candidate as containing the required object.

In accordance with the above-mentioned, in order to increase the processing speed, in the model "Faster" the generation of candidate-areas proceeds after the image passes through the convolution layer. There is a data transfer through a small neural network of a window size 3x3 via the Region Proposal Network (RPN) method by the extracted CNN features. The reported values pass into two parallel fully connected layers: box-regression layer and box-classification layer. As a result of the two layers, RPN outputs requests for candidate-areas. Those areas that contain a high probability rate of object content, pass further to the object detection module.

At the second stage, the image with the obtained candidate-areas pass through the main convolutional neural network of the Faster R-CNN model, where it becomes the subject to the object detection process. Each candidate pass to two fully connected layers. The Softmax layer evaluates whether this candidate belongs to one of the existing object classes. A regression layer that specifies the object's Bounding Box regressor.

Mask R-CNN.

As defined in the previous step, Mask R-CNN is a progressive extension of the Faster R-CNN architecture, which considers its disadvantages. One can divide the Mask R-CNN architecture into two main stages: Calculation of image features by CNN, and the alignment of object classification stages results, frames forecast, and their border shape selection.

The mask itself is a matrix which one overlaps on the borders of the detected object. The matrix values take either " 1 "- this cell contains an object, or" 0 " - this cell does not contain an object. There is a separate mask identification for each class, regardless of what are the images on the candidate-area. Then, the independent classifier selects the most accurate mask.

RolAlign operates the calculation service of feature matrix for the area-candidate. It is ineffective to form a mask in accordance with RolPool, because the feature map obtained from CNN has a smaller size than the original frame. RolPool rounded fractional values to integer values, so the mask became inaccurate. RolAilgn uses bilinear interpolation for the four nearest integral points.

The obtained result of RoIAlign combines with the bounding rectangles and class IDs, then the Mask of the detected object generates and displays on the source image.

In general, one can represent the Mask R-CNN architecture as a modified Faster R-CNN model, the main change of which is the splitting of the detected object's mask with the image, which displays by the name of this model. In case of comparison between two models, one can divide the Mask into 3 stages. The first step is to use the RPN which we described above, as to calculate the proposals of the bounding rectangles for the corresponding candidates on the detected objects. The second stage is to the point a faster R-CNN model which extracts functions by the RoLPool layer from each proposed candidate box. Then, "Faster" performs classification and regression of the bounding rectangle. And the final third stage - the mask overlays by a small R-CNN network, which outputs a class mark and a frame offset for each candidate object. Then, it adds to a mask in the form of a binary matrix with the indication of pixels which contains the detected object. Th name of the last stage is also FCN network - a semantic segmentation algorithm which uses convolution blocks and maximum pool layers, unpacking the image to 1/32 of its original size, then calculating the class forecast, and using sampling and deconvolution layers to resize the image to its original state.

The working results of the developed machine vision module and evaluation of the convolutional neural networks inference by the compute capacity of the cluster

In order to evaluate the effectiveness of this solution, there will be assessment of these two indicators. FPS (Frame per second) - an estimated metric which directly displays the speed of the convolutional neural network inference, the ratio of of the processed frames volume in relation to the time interval measured in seconds. By the evaluation of this metric in relation to the number of nodes used in the cluster, one can see the direct relationship in the increase of the module's performance improvement by the increase of nodes, and consequently the total number of CPUs to the speed of each implemented architecture for the convolutional neural networks. By the average accuracy calculation of each network, or mAP (mean average precision), one can get the relationship of the network prediction quality with regard to the increase in number of cluster nodes.

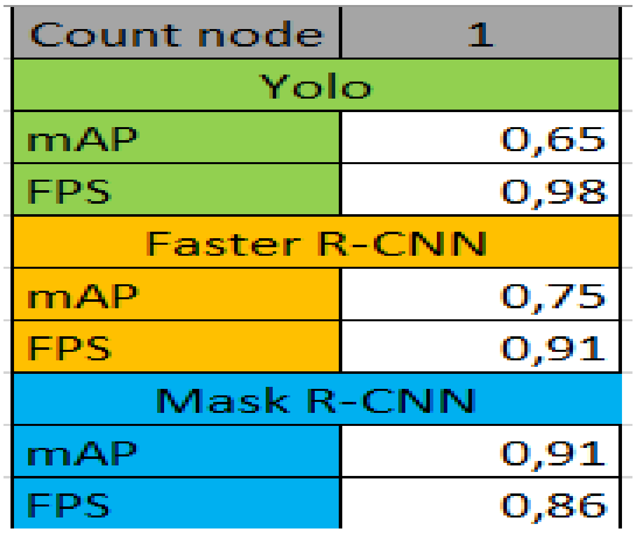
There will be the evaluation of two metrics during the inference of each convolutional neural network in turn, in case of work with one or two cluster nodes, and with 4 nodes.

In order to calculate the frame rate per second, we used three parameters: the number of frames; the time since the last frame appeared; and the current time. The number of frames increases by the end of the neural network inference cycle, then we subtract the time from the last appearance of the frame from the current time. If the result is greater than one, it means that the second has passed, and there is a display of the new FPS value.

At the first stage of testing, the inference launch of convolutional neural networks will occur on the 1st, 2nd, and 4th nodes of the cluster system, respectively.

We launch a cluster with a single worker node. Then, we run the inference of convolutional neural networks Yolo, Faster R-CNN, Mask R-CNN alternately by measurement of two metrics for FPS and mAP assessment. The test data is a video-taping which contains a record from an installed video camera on the street. This video contains a variety of objects, different sizes, movement speed, and distance from the camera. Thanks to the varied objects recognition for video from the security cameras, the results of load testing have maximum proximity to the use of the developed machine vision module under the conditions of production operations. The used models display information about FPS and sufficiency of each object detection.

In accordance with the results, we can conclude that a single cluster node can achieve only <1 FPS for each implemented convolutional neural network model, due to the limited number of cores per cluster node and the maximum computation rate of the single-board computers. The average accuracy evaluation of object recognition by convolutional neural networks shows that thanks to a simple algorithm of the Yolo network architecture, it was possible to achieve the best FPS level, but the lowest value of the mAP indicator. The network recognized the smallest number of objects among the models under the testing. The models "Faster" and "Mask" showed better results in object recognition accuracy, but due to the disadvantages of the first model, but considered in the second, the accuracy of Mask is a higher by small degree in percent. The architecture of Mask, or rather its last stage of masking, also affects the speed of the network inference and produces the lowest indicators among the models under testing.



Count node	1
Yolo	
mAP	0,65
FPS	0,98
Faster R-CNN	
mAP	0,75
FPS	0,91
Mask R-CNN	
mAP	0,91
FPS	0,86

Fig. 7 - Result of the network inference on one node of the cluster

Then, we run the inference of convolutional neural networks Yolo, Faster R-CNN, Mask R-CNN alternately by two nodes of the cluster system. By the built-in Mosix “mosmon" tool, one can track the load level on a specific cluster node.

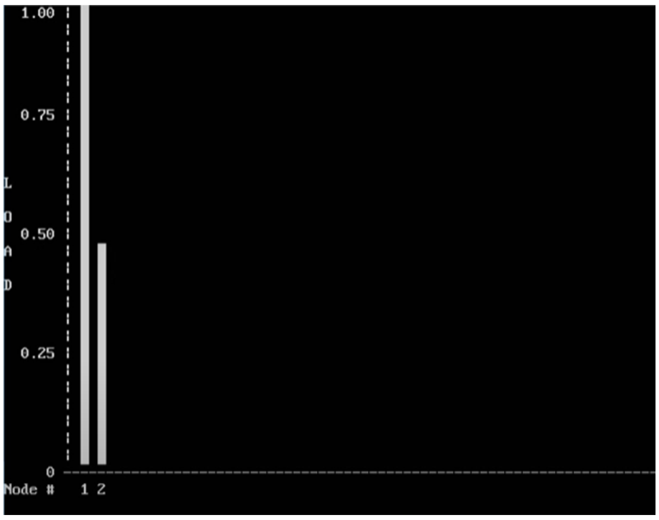
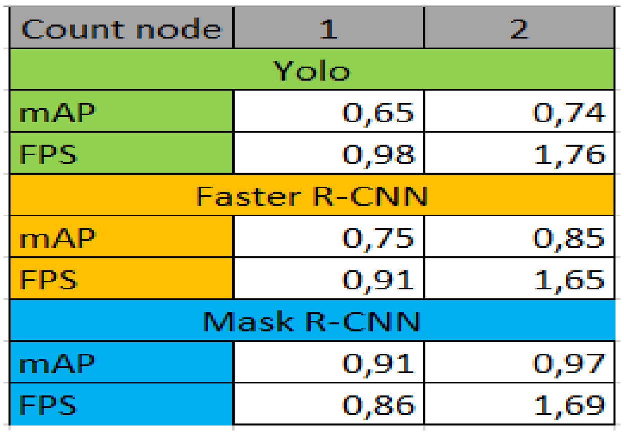


Fig. 8 - Inference of Mask R-CNN mosmon

One can notice the load distribution of the convolutional neural networks inference on the graph of the built-in mosmon tool (figure 8). As one can see in the graph, the load distributes between two nodes of the Mosix cluster system. In case of two cluster nodes use, the frame rate per second increased by an average of 0.60 fps. This FPS value is not sufficient for the production operation of the module, but the increase in metrics shows a positive improvement while the cluster nodes increase. The accuracy of the convolutional neural network Yolo has increased slightly, which indicates the shortcomings of the model architecture. the network still has the best FPS score among the tested network models. The use of 2 cluster nodes had a significant impact on the accuracy of the model Faster R-CNN with the increase of the mAP by 0.10, in contrast to the model Mask R-CNN, whose increase is less significant by just 0.6.

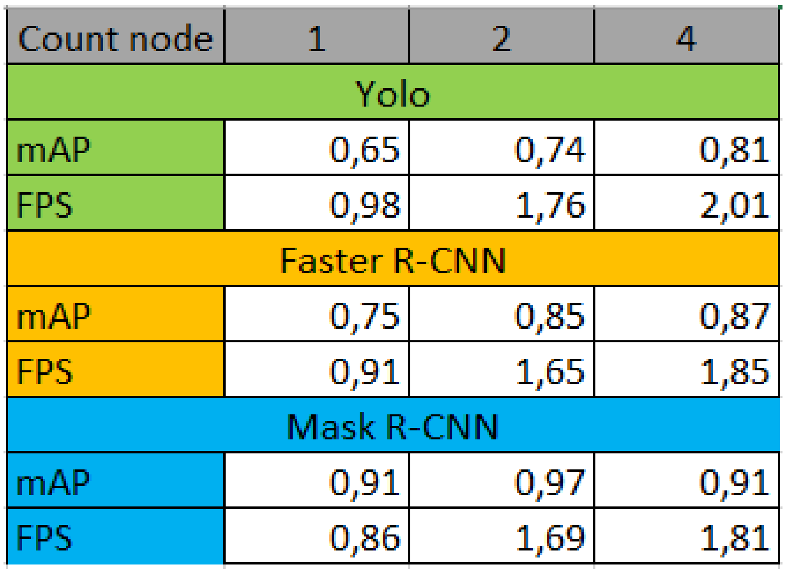


Count node	1	2
Yolo		
mAP	0,65	0,74
FPS	0,98	1,76
Faster R-CNN		
mAP	0,75	0,85
FPS	0,91	1,65
Mask R-CNN		
mAP	0,91	0,97
FPS	0,86	1,69

Fig. 9 - Result of network inference on two cluster nodes

At the last stage of testing, we will run an inference of implemented models of convolutional neural networks Yolo, Faster R-CNN, and Mask R-CNN on 4 nodes of the cluster (figure 9). The increase in the cluster nodes gives a significant increase in the number of cores within the cluster system – 16.

The inference of neural networks with the use of 4 cluster nodes showed a smaller increase in evaluation metrics than in the previous iteration of testing. In accordance with the results, one can see a slight increase in FPS of all three neural networks models, on average by 0.20. The increase in average accuracy is also insignificant in comparison with the previous iteration of testing. The Yolo model still detects objects that are closer to the camera with high accuracy, which is not true for the recognition of the remote objects. The increase in the average accuracy of R-CNN models is also insignificant, but there are objects under recognition both near located and remote objects.



Count node	1	2	4
Yolo			
mAP	0,65	0,74	0,81
FPS	0,98	1,76	2,01
Faster R-CNN			
mAP	0,75	0,85	0,87
FPS	0,91	1,65	1,85
Mask R-CNN			
mAP	0,91	0,97	0,91
FPS	0,86	1,69	1,81

Fig. 10 - Result of network inference on four cluster nodes

In accordance with the findings, we constructed the relationships graphs of changes in estimated metrics on the number of nodes used in the cluster system **(figure 11-12).**

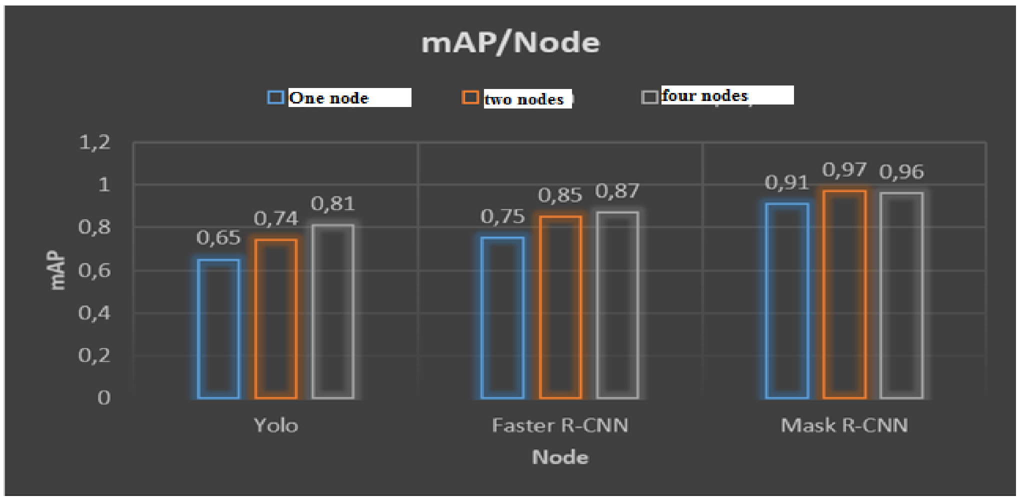


Figure 11 - nAP relationship to the number of cluster nodes

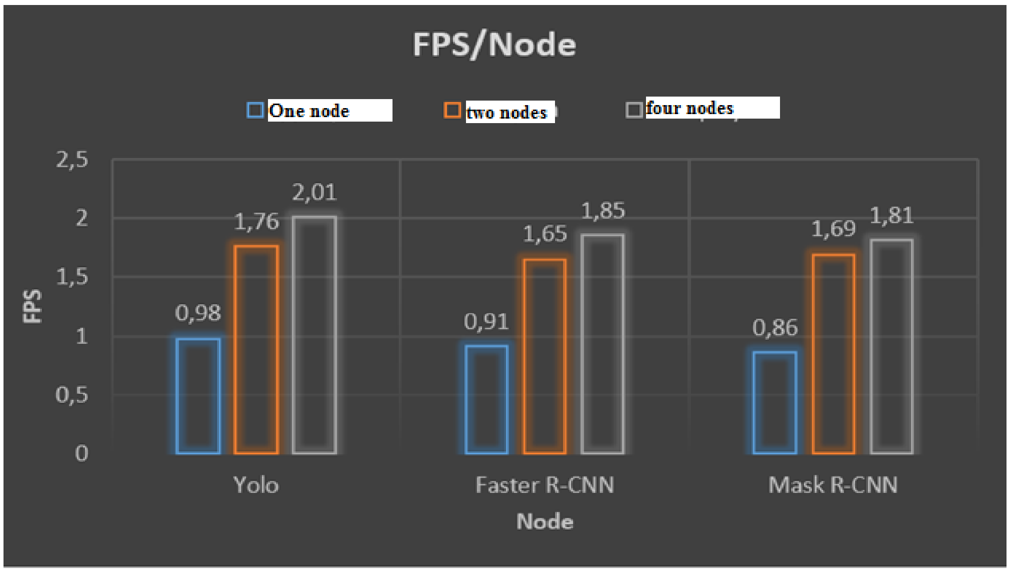


Figure 12 - FPS relationship to the number of cluster nodes

In accordance with the obtained results, one can conclude that by increase in compute capacity by use of load distribution through the combination of single-board computers Raspberry pi 3 model B in a cluster system, one can achieve a significant increase in the processing time speed by the frame per second, and a significant increase in the average accuracy of object recognition. Also, in accordance with their results, one can find that although the increase in cluster nodes increases the measured parameters, but with a larger increase in nodes, the increase will significantly decrease. This relationship refers to the fact that the algorithms for the image recognition of convolutional neural networks are largely sequential, which imposes a limit on the number of threads that are the cluster management unit within the task performance of migration processes between the processors. One also needs to consider the advantages and disadvantages of convolutional neural network models, which currently assume the maximum possible threshold for the average recognition accuracy.

CONCLUSION

Over the past decade, machine vision systems have gained tremendous momentum in their development, and in the future, the demand for the introduction of such systems in the market will only increase.

The development of machine vision systems will be colossal in the near future, due to the increase in need to control various areas, the growth of traffic flows on the streets or on the roads, the automation of the tube system, airports or offices, the growing need to use ATMs, automation of production processes. This paper uses various approaches to the design of a convolutional neural network. A single-stage approach by such approach as "You only look once", in which the image is under the processing by the network only once, and two-stage methods. Faster R-CNN in which the image was first under the processing for the areas of interest, and then passed the object recognition stage, and Mask R-CNN, which adds to the architecture a model for a mask design along the contour of the object under recognition.

Thus, one implemented the various algorithms and methods for the convolutional neural network construction. There was the development of schemes for parallelization of the load between the processor cores and cores of the cluster nodes. In accordance with the results of testing, we can conclude that the average accuracy of pattern recognition increased by 0.08% with the use of two cluster nodes, and by 0.03% with the use of four cluster nodes. We also managed to achieve a significant increase in the FPS inference of convolutional neural networks by an average of 0.78 fps with the use of two cluster nodes, and by 0.19 fps with the use of four cluster nodes.

We can conclude that the study of algorithms load parallelization and methods of machine vision for the efficiency improvement of the image recognition under the task performance for the control in various areas of interest is a relevant topic, and will only expand in the future.

Acknowledgments

The funding of the reported study refers to RFBR, project number 19-29-06036.

BIBLIOGRAPHY

Marsland S. Machine Learning: An Algorithmic Perspective (Chapman & Hall/Crc Machine Learning & Pattern Recognition) /

Dice D., Shavit N. TLRW: return of the read-write lock // Proceedings of the twenty- second annual ACM symposium on Parallelism in algorithms and architectures. ACM. 2010. P. 284-293 c.

Herlihy M. et al. Software transactional memory for dynamic-sized data structures / Proceedings of the twenty-second annual symposium of Principles of distributed computing. ACM. 2003. P. 92-101

Dice D., Shalev O., Shavit N. Transactional locking II // Distributed Computing. – Springer Berlin Heidelberg, 2006. P. 194-208.

Tomas, G. Visualization of Scientific Parallel Programs / G. Tomas, C. W. Ueberhuber. – Berlin Heidelberg: Springer-Verlag, 1994. – 310 c.

Segaran T. Programming Collective Intelligence: Building Smart Web 2.0 Applications / Segaran T. – O'Reilly Media, 2007. – 360 p.

Garetta R. Learning scikit-learn: Machine Learning in Python / Garetta R., Moncecchi G. - Packt Publishing, 2013 – 118 p.

Bengio, Y. Why does unsupervised pre-training help deep learning? / Bengio Y., Erhan D. - The Journal of Machine Learning Research. — 2010. - 625–660 p.

Bishop C. M. Pattern recognition and machine learning / Bishop C. M. — New York: Springer, 2006. — 12 p.

Bradski, G. The OpenCV library / Bradski G. – Doctor Dobbs Journal. — 2000. — 126 p.

Canny J. A computational approach to edge detection / Canny J. - Pattern Analysis and Machine Intelligence, IEEE Transactions on. — 1986. – 698 p.

.Cao, L. Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes / Cao L., Fei-Fei L. - Computer Vision (ICCV), 2007 IEEE International Conference. — 2007. - 8 p.

Amar L. Randomized Gossip Algorithms for Maintaining a Distributed Bulletin Board with Guaranteed Age Properties / Amar L., Barak A., Drezner Z. and Okun M. - Concurrency and Computation: Practice and Experience. - 2009.

Barak A., Guday G. and Wheeler R., The MOSIX Distributed Operating System, Load Balancing for UNIX. LNCS Vol. 672, Springer-Verlag, ISBN 0-387-56663-5, New York, May 1993

Barak A., La’adan O. and Shiloh A., “Scalable Cluster Computing with MOSIX for Linux,” Proc. 5th Annual Linux Expo, Raleigh, NC, p. 95–100, 1999.

Barak A., Shiloh A. and Amar L., An Organizational Grid of Federated MOSIX Clusters, Proc. 5-th IEEE International Symposium on Cluster Computing and the Grid (CCGrid05), Cardiff, 2005.

Keren A., and Barak A., “Opportunity Cost Algorithms for Reduction of I/O and Interprocess Communication Overhead in a Computing Cluster,” IEEE Tran. Parallel and Dist. Systems, 14(1), p. 39–50, 2003.

Maoz T., Barak A. and Amar L., “Combining Virtual Machine Migration with Process Migration for HPC on Multi-Clusters and Grids,” Proc. IEEE Cluster 2008, Tsukuba, 2008.

Ciresan D. Multi-column deep neural networks for image classification / D. Ciresan U. Meier, J. Schmidhuber. - Computer Vision and Pattern Recognition (CVPR). — 2012. — 3649 p.

Deng, L. Recent advances in deep learning for speech research at Microsoft / L. Deng - Acoustics, Speech and Signal Processing (ICASSP). — 2013. — 1608 p.

Droniou, A. Gated autoencoders with tied input weights / A. Droniou, O. Sigaud - International Conference on Machine Learning. — 2013. — 24 p.

Duin, R. P. W. Open issues in pattern recognition / R. P. W. Duin, E. Pekalska Computer Recognition Systems. — 2005. — 42 p.

Farneback, G. Two-frame motion estimation based on polynomial expansion / G. Farneback - Image Analysis. — 2003. — 370 p.