The system of convolution neural networks automated training

Vladislav A. Sobolevskii[a]

[a] *St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS), 14th line V.O., 39, St. Petersburg, 199178, Russia*

Abstract
In this paper the research related to the creation of a program complex, which realizes the automated generation of service-programs for the artificial intelligence systems based on the convolution neural networks is presented. The presented program complex to accelerate and simplify the generation and training of convolutional neural networks.

Keywords
Machine learning, convolutional neural networks, service-oriented architecture, internet of things

1. Introduction

In modern world the recognition technologies of photo and video images are being implemented more intensively. The development of this sphere became possible due to the appearance of new convolution neural network (CNN) architectures and the modification of existing ones. The given type of architecture turned out to be successful enough for solving the tasks of image analysis, segmentation and semantic recognition. The higher the CNN accuracy and capabilities are, the more complex CNN become. Some of the most successful and widespread CNN architectures at the moment have a plenty of heterogeneous layers [1-3]. This leads not only to the increase of work quality, but to the complication in creating and training such networks.

At the same time, the number of tasks that can be solved using CNN rises. The given tasks not always demand the application of the most complex and foremost CNN architectures, but they are still quite difficult and regular users without any knowledge of deep learning methods and their implementation skills would not be able to create and adapt these networks correctly. It can be said that the quantity of such tasks is growing faster than the number of professionals capable of solving them.

This leads to the fact that the task of creating the systems of CNN generation automation process for one or the other spheres is becoming very relevant [4-6]. At the same time, the demand for a system suitable for solving typical tasks from different spheres is becoming more acute. There are many tasks of one class (for example, the recognition of certain tree species in space images, landscape peculiarities, specific nature objects etc), the solving principle of which has been already discovered or they are being handled on the basis of an individual CNN production [7-9] or not being solved at all due to the lack of specialists.

Additionally, a lot of CNNs are produced in forms of program prototypes (for instance, using MatLab) and such prototypes require improvement for implementing into the existing monitoring systems which are designed at specific stacks of applied programming languages (C++, Java, Python etc). In its turn, this makes the further development and the following implementation of prototypes more complicated.

For solving the given tasks, the system of convolution neural networks automated training was designed based on the service-oriented approach within the project presented in this article. The approach of artificial neural networks automated generation is not new and there are some works upon this topic [10-13]. All these works point to the fact that the automation of machine learning models production process will allow to fasten the process of developing program products for solving a multitude of tasks. The system described in the article elaborates the idea of automation and has module extensible structure

Models and Methods for Researching Information Systems in Transport, Dec. 11-12, St. Petersburg, Russia
EMAIL: arguzd@yandex.ru (V. A. Sobolevskii)
ORCID: 0000-0001-7685-4991 (V. A. Sobolevskii)

which allows to add and combine trainable architectures, training algorithms, data normalization, validation etc. Moreover, due to genetic algorithms, the given system is capable of automated CNN generating and training which allows non-professionals who are not aware of neural networks setting details to use it for solving typical tasks. The work result of this system is not only a built architecture, but a generated executable file with additional REST and SOAP wrappings that without any preliminary preparations will allow to start the produced CNN as a service and apply to it from other systems and program complexes. This presents the system as a tool for a quick and effortless solving of simple typical tasks by regular users.

By present time, the designed system had already been used for generating simple deep neural networks that were introduced into third-party program products for solving specific applied tasks [14-15]. In suggested article the capabilities of the given program complex which were improved using CNN automated training are described.

2. The service-oriented approach in neural networks automation generation

The service-oriented architecture (SOA) of applications implies a module approach to the program application development [16]. In the considered situation the given paradigm is implemented at several levels.

At the level of the program complex itself SOA maintains the modularity and interchangeability of CNN generation and training algorithms. Thus, the whole process of automated generation and training is divided into some consecutively evoked program modules:

- the input data normalization module;
- the generation module of chosen CNN or the module of pre-trained CNN architecture initialization;
- the CNN training module (including verification and validation submodules).

Each of these modules is presented in several realization variants (for various CNN architectures) and certain realizations are chosen depending on the requirements. In addition, these modules are evoked from an external automated training module (it is currently implemented on the basis of a genetic algorithm) which was developed with an expectation of changeability. The other algorithms of solution search can be used instead of it and there is no need to make significant modifications to other modules for the use of these algorithms.

This approach is based on the principles of transparency and scalability which allows to expand the program product functionality by adding new modules, not by modifying the existing ones.

It is obvious that the given approach would not allow to implement the automated training of all possible CNN architectures. However, the generation and training processes of typical architectures have a precise and consecutive algorithm. Having implemented the given algorithm in the program complex it would be possible to solve the task of typical neural network solutions streaming (conveyor) implementation as the main one.

The service-oriented approach in the developed program complex occurs in the fact that all modules should not be necessarily installed to one and the same personal computer (PC). Modules can be distributed between different PCs or placed in cloud storages. Thus, the given program complex can be implemented in the form of a distributed system that blends into the SOA paradigm completely.

At the program product operation result level SOA is maintained by the implementation of autonomous service containing CNN trained to solve a specific task. This service is cross-platformed and it can be launched without any prior installing and additional software setting on the basis of some operation systems (which is possible due to the cross-platform of the given modules implementation language - Python [17]). Respectively, such module can be used in the systems maintaining both SOA paradigm and the Internet of Things (IoT) via interfaces REST and SOAP [18-20].

3. The algorithm of convolution neural networks automated training

The difficulty in CNN production and training lies in the fact that they are being trained only having a marked training dataset which describes the class of recognizable objects. The recognition of different object classes requires various CNN architectures and their parameter settings. Due to the CNN complexity this task becomes very resource-intensive. This is one of the CNN key restrictions of CNN trained with a teacher. Now the approach which consists in multitasking CNN creation for different science fields that can solve the whole class of tasks is often used [21-23]. The given approach has some advantages, particularly the higher accuracy for selected objects. However, the development of each of these CNNs is more resource-intensive and demands participation of specialists able to project the architectures of such networks. The alternative solution described in this article is the automated training of models. This kind of solution implies simultaneous training of some CNNs based on prepared information dataset for the following situational choice of the most precise model which leads to the necessity to solve the task of models parametrical adaptation quality assessment. At the same time, the formation task of training dataset in common case does not require special knowledge [24]. The automated system (AS) described in the article is relevant in such cases when the development of a wholesome CNN able to solve the task in the most accurate way is unprofitable. Using this system, it is possible to create CNN able to solve the assigned task cheaper and faster with an accuracy specified by user.

The algorithm of CNN selection was implemented in the following way:

1. In the first parent population a fixed CNN number (M) is generated with randomly set parameters.
2. N_d of new CNNs is generated, the parameters of which are selected randomly out of two occasionally chosen parent CNNs, and also N_r of CNN, the parameters of which are set completely randomly considering the given value ranges for these parameters.
3. Further, the CNN selection is performed using the roulette method (formula 1) [25]

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}, \qquad (1)$$

where p_i is the choice probability of i-CNN, f_i is the value of fitness function for i-CNN, N is the quantity of CNN in population. The roulette method was chosen as the most universal one, because the algorithm is supposed to be used for different classes of tasks. Although the use of specific algorithms would have fastened the operation speed for some task classes, but it inevitably would have slowed the operation speed for other classes. The inaccuracy estimation calculated using CNN target parameter value relatively to the real value of a test dataset (formula 2) lies in the basis of the fitness function

$$f_i = \frac{1}{\sqrt{\frac{\sum_{j=1}^{M}(\varepsilon_{ij} - \omega_j)^2}{X}}}, \qquad (2)$$

where ε_{ij} is the output value of a target parameter, which was forecast by i-network in response to an input test j-vector, ω_j is the real value of a test dataset in response to an input test j-vector, X is the quantity of test vectors.

The result of a calculation according to the given formula is a "fitness level" value, which is inversely proportional to the mean squared error of i-CNN at the test dataset. As a result of selection, M is selected to the current generation out of ($M + N_d + N_r$) CNN with the maximum pi value (choice probability of i-CNN).

4. For all CNN the mean squared error of the target parameter value calculated by them relatively to the real test dataset value is computed. If at least one CNN shows the mean squared error lower than the set value, the cycle stops. The CNN with the lowest mean squared error is treated as a "winner". Otherwise, the return to point 2 takes place. In addition, the population of each iteration is stored separately. If the population of a current iteration coincides completely with a previous population, it means that during all iteration the CNN configuration with the most accuracy has not been found and the unconditional transition to step 5 is carried out.
5. If a CNN with the mean squared error lower than the set value is not found, the cycle launches from the step 1 with a new parent population, for which new random parameter values are set. If the solution is not found after I iteration, the task is declared to be unsolvable with specified

settings and the output from the algorithm is performed.

4. Technologies used in the developed program complex

This program complex is developed in programming language Python, the main assets of which relate to its cross-platform, extensibility and large amount of sided program libraries used for solving specified tasks. The suggested programming language was chosen because at the moment it happens to be the main solution for deep learning systems development and also because it allows to realize SOA paradigm easily [26, 27]. Keras and TensorFlow libraries are used for training algorithms implementation.

Such stack of technologies is explained by the fact that the program does not face the implementation task of untypical solutions. On the contrary, the quick realization of already known architectures is required. The use of already developed, tested and optimized libraries satisfies the set task completely. At the same time, the key requirements are extensibility and scalability. Respectively, the program complex realization on the basis of a constantly extending program platform will allow to add new CNN architectures and their work tools at the cost of one program interface. The cross-platform of the described stack and the support of SOA paradigm will allow to scale the program complex to different hardware.

It is important to mention separately that CUDA SDK is also included in the used program libraries, which allows to exploit hardware acceleration during artificial neural network training using NVidia video cards [28, 29]. The use of this technology makes the process of CNN training significantly faster [30].

5. The approbation of automated convolution neural network training program complex

For approbation of the program complex prototype performing additional training of Mask R-CNN (MRCNN) CNN architecture trained on COCO dataset was developed. The given configuration was chosen because of the balance between universality and accuracy [31]. By default, MRCNN is already capable of recognizing fundamentally different object classes, from automobiles to animals. That is why, by proper additional training, it would be able to recognize a wide range of objects that are not included into COCO dataset.

The program complex was tested on the calculation task of the amount of deer in a herd from air photography. Besides the fact that deer do not belong to the COCO dataset and MRCNN is not able to distinguish them by default from the range of other creatures (sheep, gazelles, cows, horses), the specificity of this task has something to do with the fact that photos are made from various angles and distances, at different landscapes and during all seasons, which result in the fact that deer can be shot under different angles, in various scales and can have diverse colouring. What is more, due to the size of herds, deer often cover one another in photos. This leads to the fact that the described task in non-trivial and the application of CNN trained at common amount of data is impossible. In figure 1 the recognition results of one out of two images using MRCNN without additional training are shown.

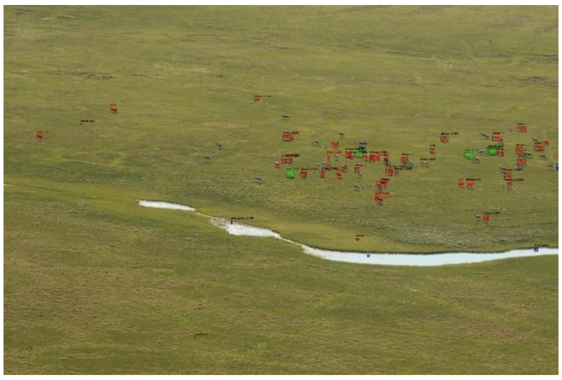
Figure 1: The deer recognition and calculation using basic MRCNN trained at COCO dataset

It can be noted that there are plenty of false negative errors evoked by the COCO dataset specificity, in which there is an insufficient number of images with similar scaling of objects. To get rid of false operations, it is required to train the network using images marked for the specified task. That is why MRCNN was additionally trained using the CNN automated training system prototype. The training was conducted in the automated mode based on the training dataset specified by a user. The following parameters of a training process were varied in the prototype:

- the quantity of training epochs;
- the quantity of training steps in each epoch;
- the speed of training;
- the threshold of detection skipping.

The CNN declared to be the winner by a system was trained on 3 epochs, with 53 training steps in each, 0,0058 training speed and 0,86 threshold of detection skipping. The described network for the same image recognized correctly 58 out of 93 deer and did not perform any false negative error (figure 2).

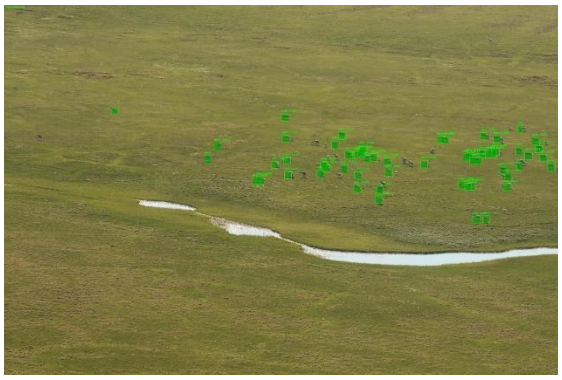
Figure 2: The deer recognition and calculation using additionally trained MRCNN

Of course, the trained CNN did not reach the maximum possible accuracy, but it can be improved in the future. What is more, the recognition accuracy may be increased by using the other CNN architectures. But the prototype testing can be considered successful because program and service coverages were generated for additionally trained MRCNN which will allow to use the received CNN for solving the set task right away. Due to the unified interface, it will be possible to perform the implementation of the most accurate CNNs in the future. Even if in the following versions a different CNN architecture is used, the program and service coverage interface will not change, and it will not be required to introduce changes into the programs at the client side.

6. Conclusion

Nowadays, the program complex is at its prototype stage and it is used for the development of some off-site applications. First of all, to start the full operation the improvement of user application interface is needed. As at the prototyping step the product is used by specialists in machine learning, the current interface is not adapted for using by regular users. Because of this, the accessibility for the wide user audience which is one of the key tasks facing the program complex is not being solved at the moment.

In addition, because of the high-performance requirements, during the given program product functioning the program complex transition to highly productive servers is needed for the commercial use. The calculation specifity during CNN training puts a range of requirements to the hardware and the commercial use implies the parallel training of several models that can load the system significantly. Despite the calculation parallelism put in the program complex architecture using SOA, it is demanded to perform the additional research and stress-tests to outline the specific requirements to the hardware.

Acknowledgements

This work was supported by the RFBR grant №19-37-90112 and the budgetary theme 0073-2019-0004.

References

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, Communications of the ACM (2017), volume 60, issue 6, pp. 84 – 90.

[2] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 3rd International Conference on Learning Representations (2015).

[3] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, 3th European Conference on Computer Vision (2014), volume 8689, issue 1, pp. 818 – 833.

[4] Z. Geng, Y. Wang, Automated design of a convolutional neural network with multi-scale filters for cost-efficient seismic data classification, Nature Communications, volume 11, issue 1, 2020.

[5] M. Witsuba, A. Rawat, T. Pedapati, Automation of deep learning, Proceedings

of the 2020 International Conference on Multimedia Retrieval (2020), pp. 5-6.

[6] B. Baker, O. Gupta, N. Naik, R. Raskar, Designing neural network architectures using reinforcement learning, 5th International Conference on Learning Representations (2017).

[7] Ateeq-ur-Rauf, A. R. Ghumman, S. Ahmad, H. N. Hashmi, Performance assessment of artificial neural networks and support vector regression models for stream flow predictions, Environmental Monitoring and Assessment, volume 190, issue 12, article 704, 2018.

[8] Z. Alizadeh, J. Yazdi, J. H. Kim, A. K. Al-Shamiri, Assessment of machine learning techniques for monthly flow prediction. Water (Switzerland), volume 10, issue 11, article 1676, 2018.

[9] J. Lantrip, M. Griffin, A. Aly, Results of near-term forecasting of surface water supplies, Proceedings of the 2005 World Water and Environmental Resources Congress, Anchorage, Alaska, US, 2005. doi: 10.1061/40792(173)447.

[10] I. Bello, B. Zoph, V. Vasudevan, Q. V. Le, Neural optimizer search with Reinforcement learning, 34th International Conference on Machine Learning (2017), volume 1, pp. 712-721.

[11] H. Cai, T. Chen, W. Zhang, Y. Yu, J. Wang, Efficient architecture search by network transformation, 32nd AAAI Conference on Artificial Intelligence (2018), pp. 2787-2794.

[12] J.-D. Dong, A.-C. Cheng, D.-C. Juan, W. Wei, M. Sun, DPP-Net: Device-Aware Progressive Search for Pareto-Optimal Neural Architectures, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 11215, pp. 540-555, 2018.

[13] M. Wistuba, Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 11052, pp. 243-258, 2019.

[14] V. Mikhailov, A. Spesivtsev, V. Sobolevsky, N. Kartashev, Multi-Model Estimation of the Dynamics of Plant Community Phytomass, The 13th IEEE International Conference Application of Information and Communication Technologies, Baku, Azerbaijan, pp. 324 – 328, 2019.

[15] V. A. Zelentsov, A. M. Alabyan, I. N. Krylenko, I. Yu. Pimanov, M. R. Ponomarenko, S. A. Potryasaev, A. E. Semenov, V. A. Sobolevskii, B. V. Sokolov, R. M. Yusupov, A Model-Oriented System for Operational Forecasting of River Floods, Herald of the Russian Academy of Sciences, volume 89, issue 4, pp. 405 – 417, 2019. doi: 10.1134/S1019331619040130.

[16] M. Bell, Introduction to Service-Oriented Modeling, in Service-Oriented Modeling: Service Analysis, Design and Architecture, Wiley & Sons, New York, NY, 2008.

[17] V. John, Guttag Introduction to Computation and Programming Using Python: With Application to Understanding, 2nd Edition, MIT Press, Cambridge, Massachusetts, 2016.

[18] Y. Mesmoudi, M. Lamnaour, Y. E. L. Khamlichi, A. Tahiri, A. Touhafi, A. Braeken, Design and implementation of a smart gateway for IoT applications using heterogeneous smart objects, 4th International Conference on Cloud Computing Technologies and Applications, Cloudtech, 2018.

[19] D. Hanes, IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things, Cisco Press, Indianapolis, Indiana, 2017.

[20] T. Erl, Service-Oriented Architecture: Analysis and Design for Services and Microservices, 2nd Edition, Prentice Hall, Upper Saddle River, New Jersey, 2016.

[21] D. Xu, Z. Tian, R. Lai, X. Kong, Z. Tan, W. Shi, Deep learning based emotion analysis of microblog texts, Information Fusion, volume 64, pp. 1-11, 2020.

[22] U. Ozkaya, F. Melgani, M. Belete Bejiga, L. Seyfi, M. Donelli, GPR B scan image analysis with deep learning methods, Measurement: Journal of the International Measurement Confederation, volume 165, 2020.

[23] A. Dutta, T. Batabyal, M. Basu, S. T. Acton, An efficient convolutional neural network for coronary heart disease prediction, Expert Systems with Applications, volume 159, 2020.

[24] M. Sewak, M. R. Karim, P. Pujari, Practical convolutional neural networks: implement advanced deep learning models using Python, Packt Publishing, Birmingham, UK, 2018.

[25] L. A. Gladkov, V. V. Kureichik, V. M. Kureichik, Genetic algorithms: a textbook, 2nd Edition, Fizmatlit, Moscow, Russia, 2006.

[26] T. Ziade, Python Microservices Development, Packt Publishing, Birmingham, UK, 2017.

[27] G. C. Hillar, Internet of Things with Python, Packt Publishing, Birmingham, UK, 2016.

[28] D. B. Tuomanen, Hands-On GPU Programming with Python and CUDA: Explore high-performance parallel computing with CUDA, Packt Publishing, Birmingham, UK, 2018.

[29] J. Han, B. Sharma, Learn CUDA Programming: A beginner's guide to GPU programming and parallel computing with CUDA 10.x and C/C++, Packt Publishing, Birmingham, UK, 2019.

[30] B. Vaidya, Hands-On GPU-Accelerated Computer Vision with OpenCV and CUDA: Effective techniques for processing complex image data in real time using GPUs, Packt Publishing, Birmingham, UK, 2019.

[31] K. He, G. Gkioxari, P Dollar, R. Girshick, Mask R-CNN, Proceedings of the IEEE International Conference on Computer Vision, volume 2017, pp. 2980-2988, 2017.