International Journal of Artificial Intelligence,
ISSN 0974-0635; Int. J. Artif. Intell.
2024, Volume 22, Issue Number 2
Copyright © 2024 [International Journal of Artificial Intelligence]

AutoML Technologies for Animal Monitoring

V. A. Sobolevskii[1] and **V. V. Mikhailov**[2]

[1]St Petersburg Federal Research Center of the Russian Academy of Science,
39, 14th Line V.O., St Petersburg, 199178, RUSSIA
Email : arguzd@yandex.ru

[2]St Petersburg Federal Research Center of the Russian Academy of Science,
39, 14th Line V.O., St Petersburg, 199178, RUSSIA
Email : mwwcari@gmail.com

ABSTRACT

Decision making in animal husbandry is critical to improving animal health and welfare, increasing production efficiency and minimizing environmental impact. Deep learning algorithms can be trained to recognize signs of animal condition in photographs and patterns of animal behavior in videos. The results of recognizing animal condition and behavior can provide valuable information about animal health and welfare. However, the use of deep learning algorithms has its limitations in the form of the need for large amounts of high-quality data and machine learning experts. In this paper, a system for complex synthesis of deep learning models for animal monitoring based on Automated Machine Learning approach has been presented. This system can be used by non-experts in machine learning and users not capable of programming to create animal monitoring programs based on deep learning algorithms. The versatility of the created system was proved by testing it on the tasks of recognizing fundamentally different species of animals - reindeer and brant.

Keywords: AutoML, monitoring, recognizing, computer vision, deep learning.

Mathematics Subject Classification: 68T05, 68T07

Computing Classification System: G.4

1. INTRODUCTION

Decision-making in livestock production is critical to improving animal health and welfare, increasing production efficiency and minimizing environmental impact. Good decision-making depends on accurate and timely information regarding the current state of management facilities. Regular collection of such information is carried out by monitoring systems. The monitoring data set should be extensive and include information on the animals themselves (age, weight, health status, feed intake, weight gain), on feed, on the animals' impact on the environment, including waste management and water use. Animal monitoring can help detect early signs of disease or illness. Early detection of health problems can provide rapid treatment, which can reduce the risk of disease spread and improve the overall health and welfare of the animals. Monitoring can also help track the productivity of individual animals or groups of animals. Monitoring feed intake and weight gain can help ensure that animals are receiving the right amount of feed for their growth and development.

For captive grazing and commercial wildlife populations, monitoring provides information on animal distribution, herd size, sex and age structure and fecundity. Monitoring data on commercial populations determines removal limits, which allows for the regulation of numbers to maximize production, considering the capacity of the forage base. In the case of endangered populations, monitoring data serve as a basis for decision-making on methods to protect and restore animal numbers.

Expansion of the list of monitoring parameters in agricultural production tasks leads to the need for automation of data processing and intellectualization of this process. One of the most promising and rapidly developing areas of artificial intelligence is currently deep learning. In agriculture, deep learning is beginning to be used to analyses large datasets and recognize objects in photos and videos (Kundu et al., 2021; Noran, 2023). The second direction is the creation of automatic monitoring and forecasting systems, based on deep learning models (Okinda et al., 2020; Wen-Hao, 2020; Turnip et al., 2022).
Deep learning algorithms can be trained to recognize signs of animal condition in photographs and patterns of animal behavior in videos. The results of recognizing the condition and behavior of animals can provide valuable information about their health. Another area where deep learning has been applied is in monitoring environmental conditions. Factors such as temperature, humidity and air quality can have a significant impact on animal health. Deep learning algorithms can be trained to analyze sensor data from environmental monitoring systems and identify patterns that may indicate sub-optimal conditions. This can help producers act to improve environmental conditions and prevent potential health problems.

Deep learning can also be useful for monitoring large herds, free-ranging herds and semi-wild herds. An example of such a livestock industry is reindeer husbandry (Mikhailov, 2022). In this industry, monitoring the number of individuals in a herd and their age and sex characteristics is an important task. Since herds may contain several thousands of individuals, manual herd counting is time consuming. Therefore, the use of deep learning models for automatic recognition and counting of animals in images from airplanes, drones or space satellites allows to quickly obtain the necessary information about the state of large herds and make timely decisions.

However, the use of deep learning algorithms has its limitations. One important issue is the need for large amounts of high-quality data. Deep learning algorithms rely on large datasets to learn patterns and make accurate predictions. In animal husbandry, collecting and labelling large amounts of data can be time-consuming and expensive. In addition, the accuracy of deep learning algorithms can be affected by the quality of the data used for training. Therefore, careful selection and preparation of high-quality data is crucial for successful application of deep learning in animal husbandry.

Another significant challenge is the need for deep learning experts. The development and implementation of deep learning algorithms requires special expertise in programming and data science. In animal husbandry, this may require co-operation between experts in animal science and data science. In addition, interpreting the results obtained from deep learning algorithms may require expertise in animal behavior and physiology.

2. RELATED WORK

To solve the problems described above, this paper proposes to use Automated Machine Learning (AutoML) technologies. The basic tasks that are solved by AutoML technologies are to develop and implement methodologies and technologies for automatic generation and training of machine learning models in general and deep learning in particular. This research area is new, but there are already many research and development groups working on this topic. Today, software that allows automatic generation of deep learning models for solving a variety of tasks is becoming more and more active.
Examples of such software solutions are auto-sklearn (Feurer et al., 2019), AutoKeras (Jin et al., 2020) and a number of others. However, these software systems can often be used only on one platform. Or they are presented only as a software library, which does not allow to use them as independent software solutions. Also, as an example, we can cite an extension for MatLab platform (Shure, 2020), which provides both automation of model training processes and compilation of created models into C++ executable files. However, a deep learning specialist is required to work with these software solutions. These software solutions are tools for automating the work processes of deep learning specialists and cannot be used by users without specialized knowledge.

The most elaborate and feature-rich AutoML systems to date have been created by large companies. One example of such systems is Google Cloud AutoML (Zeng and Zhang, 2020), developed by Alphabet Inc. The center of this system is a cloud-based software platform that solves the problem of automating the creation of deep learning models for various tasks. This system also has and graphical user interface, which simplifies its operation. This software platform is built on the principles of scalability, so it is actively extended with different deep learning models that can be used for fundamentally different tasks. However, this system is also a tool for deep learning professionals and can hardly be used by non-professionals. It is worth noting that, to date, the result of this system is a cloud-based software service, which needs to be refined for integration into third-party software. Also, this system has all the disadvantages of a cloud platform. There is a critical dependence on the Internet connection. Without a stable connection, working with the platform will be difficult or generally impossible. Since data is stored on servers owned and managed by a third-party company, there is a risk of unauthorized access or data leakage. Users themselves have limited control over the infrastructure and resources used to store and process their data in the cloud. It may then be difficult or generally impossible to switch to another platform or migrate the trained model to local servers. Also, there is a risk of losing access to the trained model due to natural disasters or force majeure.

To summarize, there are already developments and research in AutoML that can be used to solve the monitoring problem in livestock production. However, the vast majority of these systems are tools for deep learning specialists and cannot be used by non-specialists. Also, many of these systems will not be able to solve the problems associated with the lack of training data.

3. MATERIALS AND METHODS

3.1. Setting goals AutoML for animals monitoring

In this paper, we propose a comprehensive approach to automated creation of recognition models based on deep learning, aimed primarily at non-specialists in deep learning and the creation of deep learning models when there is a lack of training data. To achieve this effect, the AutoML concept is used in the following tasks:
- structural-parametric synthesis of a deep learning model and optimization of its hyperparameters;
- transfer training of the deep learning model;
- creation of easy-to-use software for automated synthesis of deep learning model.

The first task can be formally represented in the following form:

$$Q^a(N_i^a(p, L^a(p, g^a)), q) \to max_{i\in[1,n]},$$

if the condition:

$$P_i(N_i^a) \geq P_{targ},$$

where *p* is the training sample provided by the user and used in the processes of generation and training of deep learning models; *q* - test sample (including data for model validation) provided by the user and used in the processes of generation and training of deep learning models; *a* - index of the deep learning architecture used to solve the problem set by the user; g^a - set of hyperparameters of the deep learning architecture under the index *a*, varied in the learning process; L^a - algorithm of automatic selection of hyperparameters of deep learning architecture under index *a*; N_i^a - *i* model of deep learning architecture under index *a*, which was trained by the algorithm of automatic selection of hyperparameters L^a on the sample *p*; *i* - index of the deep learning model; *n* - a given number of instances of generated and trained deep learning models, among which the most accurate model is searched; Q^a - function of calculation of the accuracy of the *i* deep learning model of the architecture under the index *a*; P_i - function of calculation of accuracy of the trained *i* model; P_{targ} - user-defined boundary value of the required accuracy of the trained model.

The task of structural-parametric synthesis of a deep learning model is reduced to training several models. In each of them hyperparameters are selected using a specialized algorithm. After that, the most accurate model is selected using a known accuracy calculation function.

The second task was chosen to implement transfer learning (Hosna et al., 2022). Its essence is that deep learning models are trained in several stages. At the first stage, the model is trained on a large array of images of different objects (different animals, people, cars, etc.). At the next stages the model is further trained on images of target objects (images of those objects that need to be recognized). This approach allows to significantly reduce the required number of marked images of target objects.

The third task is to create software that is extensible and easy to use. Firstly, an extensible architecture will allow new deep learning architectures to be implemented quickly afterwards. Second, a graphical interface can simplify the model development and training process. Also, part of this task is to implement the concept of «No Code» development (Woo, 2020). This concept will provide the ability to create and customize a deep learning model without the need to write software code. The described approach to software development can simplify and speed up the processes of deep learning architecture selection, hyperparameter tuning and data preprocessing. Also, visualization of the results of the created deep learning models is provided, which facilitates interpretation of the results and helps to make informed decisions about the applicability of the created model.

3.2. Synthesis of deep learning models

Since this work is focused on unified work with different deep learning architectures, a unified algorithm should be used to comprehensively automate the processes of their generation and training. It is required to generate and train different models in a uniform way. But the number and type of hyperparameters differ for different deep learning architectures. Therefore, for their automated training it is necessary to consider algorithms capable of working with a dynamically expandable set of variables. Due to this approach, when introducing a new deep learning architecture into the developed software system, it will not be necessary to make any modifications to the algorithm. The software system will accept the new array of variables, regardless of the number and type of variables. After that, it will automatically vary these variables in the process of automated training of the deep learning model.

The fastest algorithms that satisfy this requirement belong to the classes of evolutionary algorithms and swarm intelligence algorithms. Algorithms of these classes efficiently and not resource-intensive implement random directed search when solving an optimization problem with a dynamic set of variables. Also, the advantage of these classes of algorithms is their flexibility and scalability. Due to what in the considered problem the conditions of optimal search can be varied depending on the chosen deep learning architecture. The most suitable for the task at hand are: genetic algorithm (McCall, 2005), bee swarm algorithm (Akbari et al., 2009) and clonal selection algorithm (Ülker and Ülker, 2012).

According to the results of the research, it appeared that the bee swarm and clonal selection algorithms have certain limitations for application in this problem. The bee swarm algorithm chooses the initial search point randomly, but further search is conducted in the area around this point. While clonal and genetic algorithms search the entire available solution space from the very beginning. On the other hand, the clonal selection algorithm, unlike the bee swarm and genetic algorithms, is more difficult to modify and scale. This algorithm requires describing the interaction of changing parameters at the selection stage, while calculating the degree of influence of parameter changes on the accuracy of the generated model.

To summarize, the classical genetic algorithm was taken as the basis for the algorithm of complex automation of the processes of generation and training of deep learning models. On the one hand, swarm intelligence algorithms have limitations in random search. On the other hand, modifications of the genetic algorithm itself lose to it in universality. At the same time, universality in the considered task is a more important characteristic than accuracy in solving specific types of tasks. It is worth noting that the closest analogues for solving the problem of automated synthesis of deep learning models also use modifications of the evolutionary algorithm (Nikitin et al., 2020). This demonstrates the effectiveness of this class of algorithms in solving the described problem.

To increase the degree of universality, additional modifications were made to the genetic algorithm (figure 1):

- The genotype was represented as an expandable array whose elements can be either integer or real. This is due to the fact that, firstly, hyperparameters of deep learning models in turn can be both integer and real, and secondly, new deep learning architectures can be added to the developed software system in the future and it would be difficult (in software implementation) to avoid modification of the created algorithm with a different genotype format.
- The mutation operator (gene inversion) for child deep learning models is not applied due to the specific genotype format.
- The crossover (crossing operator) is random and multipoint. At each crossover, the gene is split into a number of parts equal to the number of array elements. Such a modification of the genetic algorithm is specific to the task at hand, since it is not known in advance which hyperparameters and how they should be changed to improve the accuracy of a particular deep learning model.

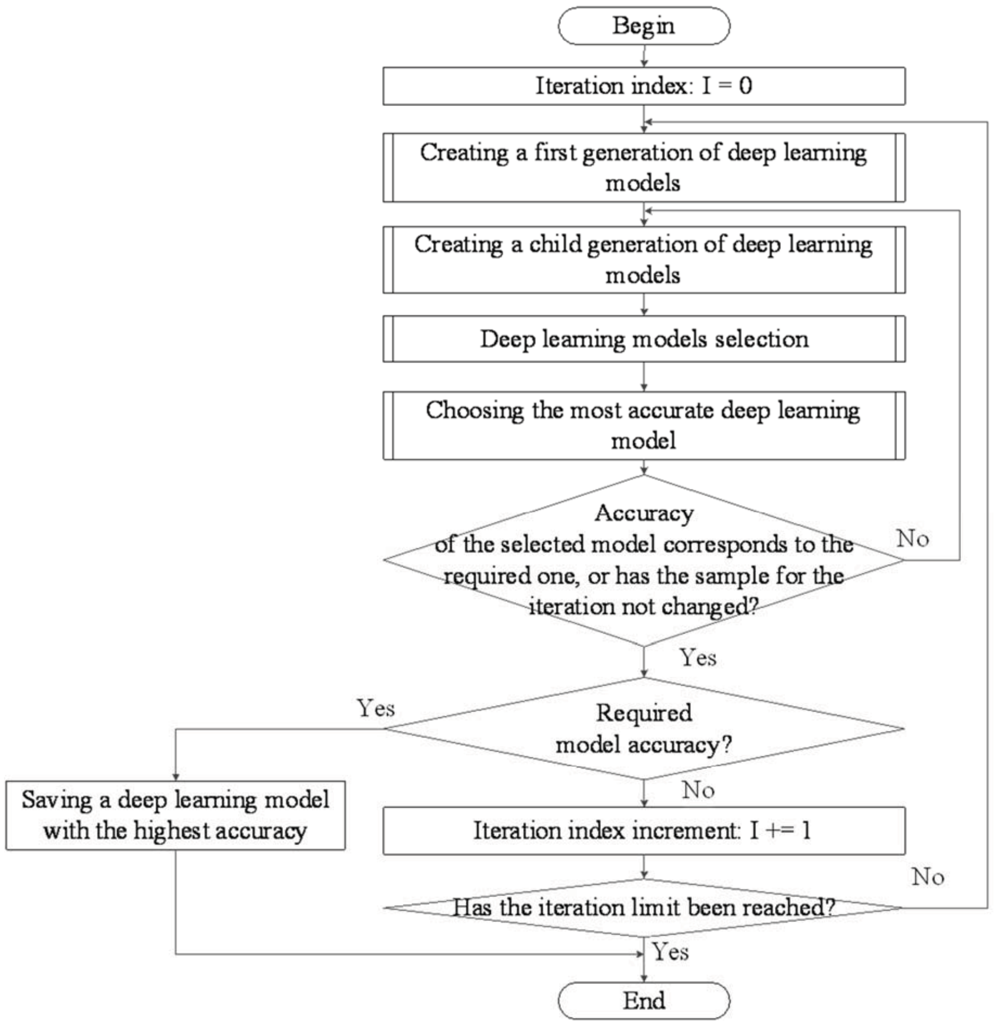
Begin
Iteration index: I = 0
Creating a first generation of deep learning models
Creating a child generation of deep learning models
Deep learning models selection
Choosing the most accurate deep learning model
Accuracy of the selected model corresponds to the required one, or has the sample for the iteration not changed?
No
Yes
Yes
Required model accuracy?
No
Saving a deep learning model with the highest accuracy
Iteration index increment: I += 1
No
Has the iteration limit been reached?
Yes
End

Figure 1. Scheme of an algorithm for the automated synthesis of deep learning models

The described modified genetic algorithm is used in the presented work for automated creation of deep learning models that already solve the problem at hand. Its use accelerates and simplifies the creation of such models. Even users without specialized knowledge can use the created program, since the selection of hyperparameters for the model is completely taken over by the presented algorithm.

3.3. Deep learning model for animals monitoring

The Mask Regions with Convolution Neural Networks (MRCNN) architecture (He et al., 2017) was used to directly solve the animal monitoring problem. The key feature of this architecture is the combination of a convolutional neural network architecture called Faster Regions with Convolution Neural Networks (FRCNN), which is responsible for solving the classification problem, and the Mask Head module, which is responsible for solving the image segmentation problem. The result of MRCNN is the combined response of these two models.

The MRCNN architecture was included in the software system under development for two reasons. Firstly, it is one of the most accurate architectures to date for both recognition and segmentation tasks. Solving the segmentation problem, in turn, is important because herd animals are often tightly grouped. Because of this, they may overlap each other in many images. Therefore, in order to correctly recognize each individual, it is necessary to locate the boundary between them, i.e. to solve the segmentation problem. Secondly, the example of this architecture demonstrates the possibility of automated training of complex organized architectures that require a specialized approach to the learning process.

Also, transfer learning has been implemented in the developed system. With this approach, the training of the model for solving the task will not be performed from the beginning. The MRCNN model already trained on the Microsoft COCO Dataset (Microsoft Common Objects in Context) (Lin et al., 2014) was taken as a basis. This dataset is by far the largest scale dataset used to train deep learning models for detection and segmentation tasks. This dataset consists of 328 thousand images. All of these images are already labelled and formed into training samples. Therefore, using this dataset for the basic training of MRCNN allows all the basic concepts of different object classes to be defined for it.

After basic training on the basis of MS COCO Dataset, the obtained model can either be used immediately, if the objects to be recognized were included in the training dataset, or it can be further trained for a specific target task. I.e. retrain it to recognize and segment objects that were not included in the training dataset. In the case of retraining, a much smaller sample size will be required than if the network had to be trained from the beginning. However, this sample size is different for each task and it is not possible to calculate an average value.

3.4. Architecture animals monitoring software

Several approaches were used in this paper to create extensible and easy to use software. The main approach used was service-oriented architecture (SOA). SOA is a software development paradigm in which software modules provide the ability to work with the methods and functions implemented in them to other software modules through a service shell via communication protocols over the Internet. The SOA paradigm is independent of the software libraries and specific communication protocols used.

SOA implies a modular approach to software development. This approach is based on the principles of transparent access to functions and scalability, which allows extending the functionality of a software product not by modifying existing modules, but by adding new services. Transparent access (or transparency) hereinafter refers to such access to objects and functions of different modules, which is carried out using the same operations, regardless of whether they are local or remote. That is, the software interface for accessing a particular object or function must be consistent for that object, regardless of where it is actually stored in the system and how it is modified during the development or maintenance of the software product (Anthony, 2016). Also, SOA follows the concept of Internet 2.0, which is a significant advantage today.

To follow the given SOA paradigm, it is required that the synthesized deep learning models, from the software point of view, have a typical software or web interface and use common data exchange protocols. In the developed software, an offline service containing a deep learning model is generated. This model is created and trained to solve a specific problem using user-specified data. The service consists of the deep learning model itself, as well as software shells implementing the REST and SOAP interface.

This approach will significantly simplify the operation of animal monitoring deep learning models synthesized by the system. Firstly, the generated services for deep learning models are cross-platform and can be run on different operating systems without prior installation and configuration. At the same time, the implementation of several software shells makes it possible to access the deep learning model as a service through the most common web interfaces. Secondly, the generated deep learning model can still be used as a simple software library by accessing it through appropriate software interfaces. At the same time, the generated models do not have to be installed on the same computer. They can be distributed among different computers or they can be located in cloud storage. Thirdly, the synthesized models can be accessed directly through the user interface of the software presented in this paper.
To ensure transparency and scalability of the developed system, classes implementing program functions of different deep learning architectures were encapsulated in separate modules. The work with these modules was realized using the design template «Facade» (Gamma et al., 1994). In this template, the modules are accessed through a common programming interface, which redirects the call, depending on the input parameters.

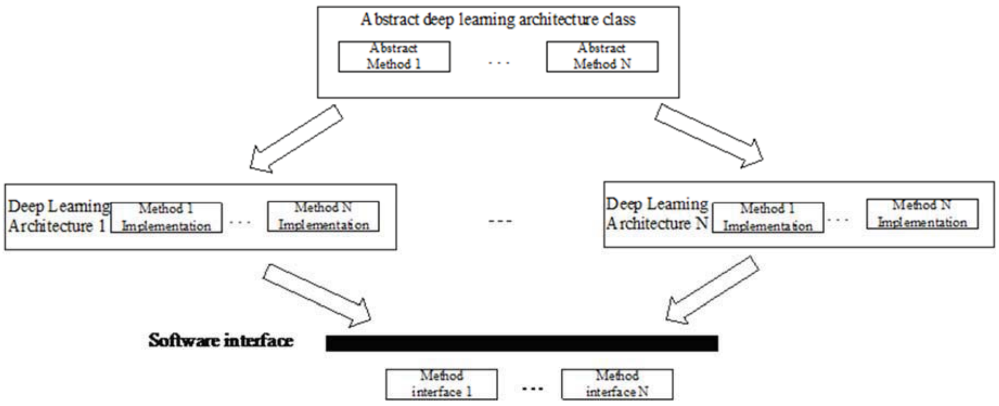

Figure 2. Scheme for implementing the «Facade» pattern

As can be seen from figure 2, the software module of each deep learning architecture inherits from a common abstract class and implements all the methods that are necessary for the synthesis of the deep learning model of the corresponding architecture. These modules are accessed through a common programming interface, which, depending on the chosen architecture, invokes a specific implementation of the corresponding method.

The use of the "Facade" pattern allows you to easily extend the software by adding new architectures to it. In case of adding a new architecture, it is enough to create a class for it, inherited from an abstract class, and implement all the necessary methods. In practice, it may be enough just to connect a third-party software library and override the called functions for it. As a result, the whole integration of the new deep learning architecture can be reduced to just adapting third-party software libraries to the corresponding software interface.

Based on the approaches described above, the architecture of the developed software was designed as follows.

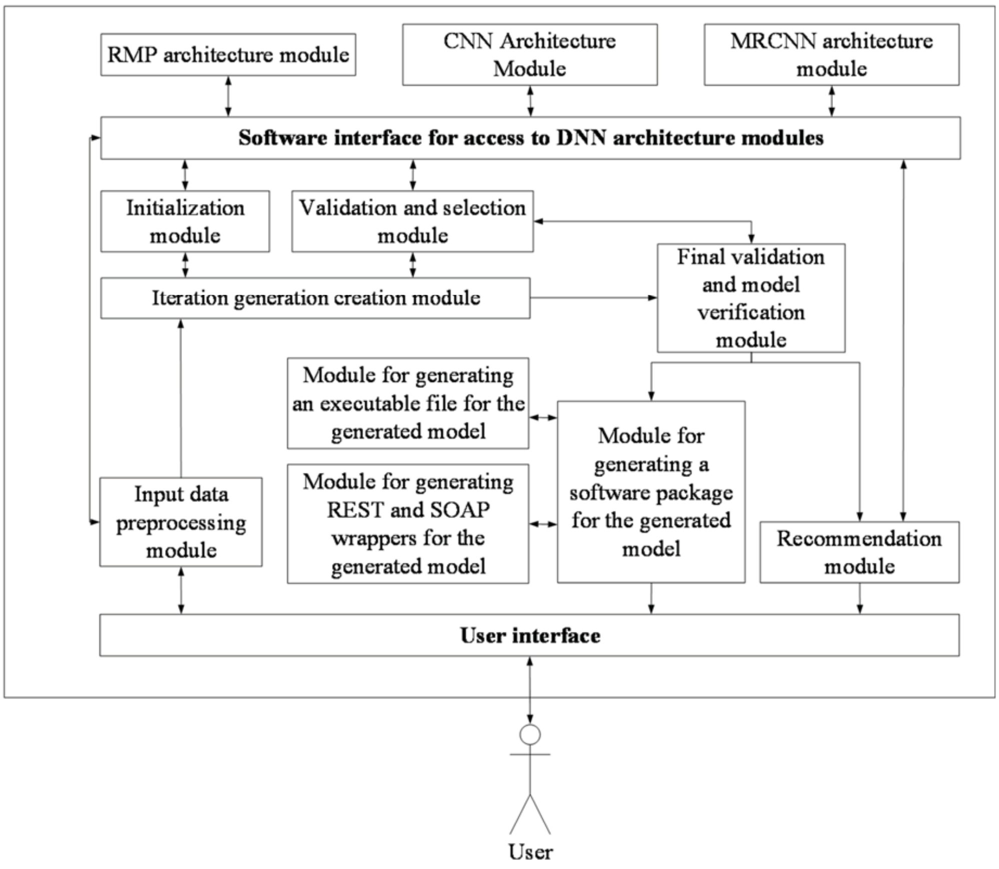
RMP architecture module
CNN Architecture Module
MRCNN architecture module
Software interface for access to DNN architecture modules
Initialization module
Validation and selection module
Final validation and model verification module
Iteration generation creation module
Module for generating an executable file for the generated model
Module for generating a software package for the generated model
Module for generating REST and SOAP wrappers for the generated model
Input data preprocessing module
Recommendation module
User interface
User

Figure 3. The architecture of the developed software

Figure 3 shows that the software implements the concept of «No Code» development. Thanks to this concept, the user is able to create fully functional software implementations of deep learning models without having to write software code himself. It also allows for faster and easier integration of the created models into third-party software.

4. RESULTS

According to the architecture presented above, the software for automated creation of deep learning models AutoGenNet was created. The main purpose of this software is that a user without writing any code can create a program based on a deep learning model to solve an animal monitoring task. The created program can be used stand-alone or integrated into existing third-party software.

The main task on which the developed software was tested was the task of monitoring reindeer herds. The currently used methodology of counting reindeer numbers in tundra populations (Taimyr, Yakut populations, Chukotka reindeer, and migrating herds of reindeer in Canada and Alaska) is based on the ecological peculiarities of the species. Their essence is that reindeer in hot weather during the flight of insects gather in herds of many thousands in a limited area in the northern part of the summer range (subzones of Arctic and typical tundra). Herds in the aggregations are photographed and the number of animals in them is counted manually "by head". The developed system was used to automate this process.

For transfer training of the MRCNN model, a training sample of only 100 aerial images and a test sample of 30 images were prepared. Using the AutoML approach, the hyperparameters that resulted in the best accuracy of the retrained model were selected. The main hyperparameters of the MRCNN retraining were: 20 training epochs, 60 training steps per epoch, a learning rate of 0.0058 and a detection miss threshold of 0.7.

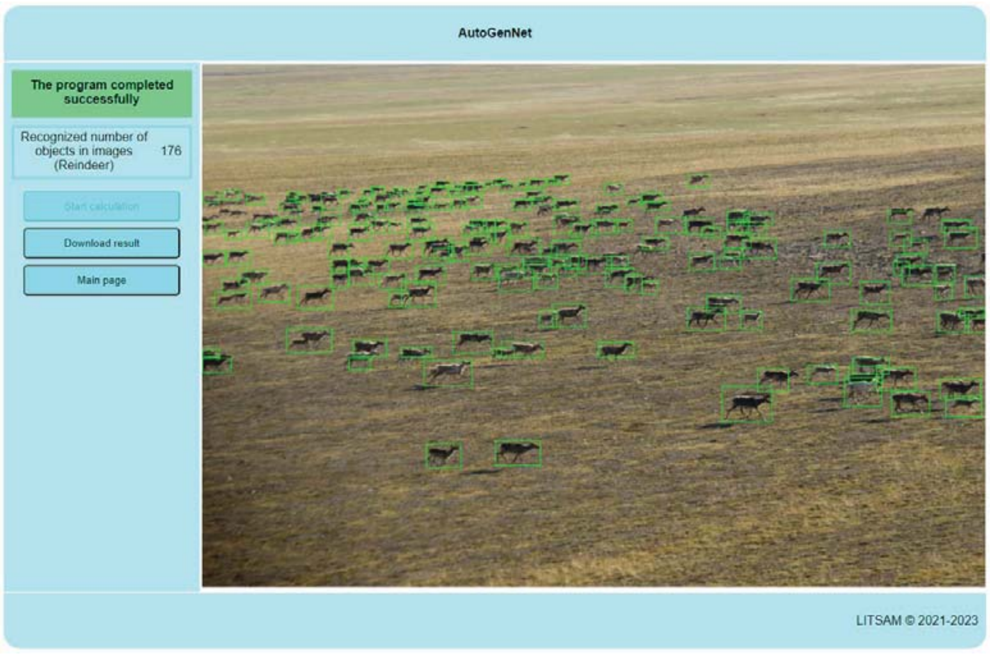
AutoGenNet
The program completed successfully
Recognized number of objects in images (Reindeer) 176
Start calculation
Download result
Main page
LITSAM © 2021-2023

Figure 4. Result of reindeer recognition by the retrained MRCNN

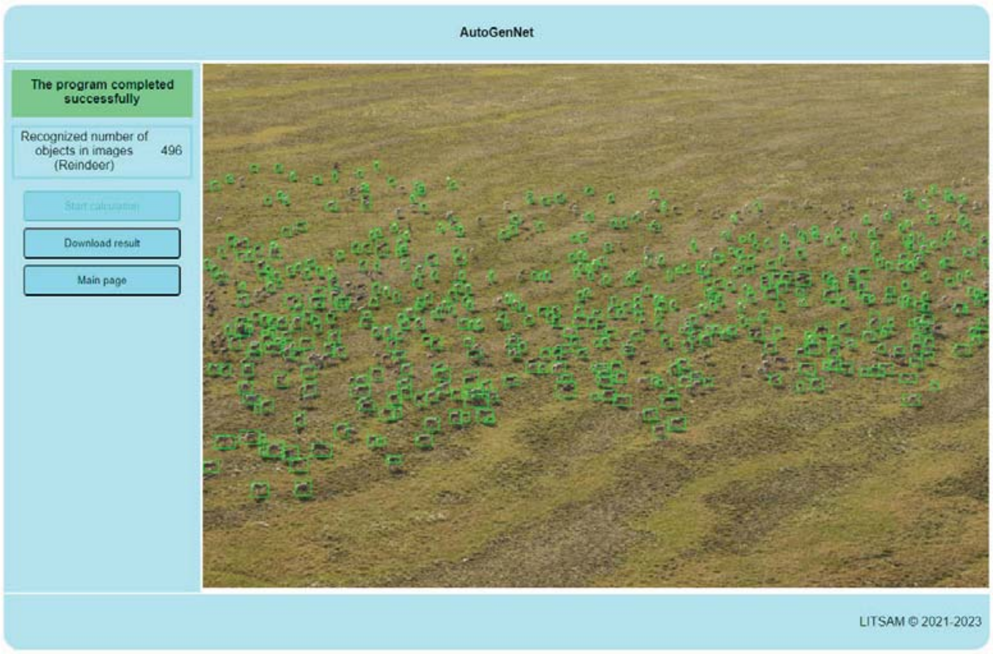
AutoGenNet
The program completed successfully
Recognized number of objects in images (Reindeer) 496
Start calculation
Download result
Main page
LITSAM © 2021-2023

Figure 5. Recognizing and counting reindeer in a large herd against a homogeneous background

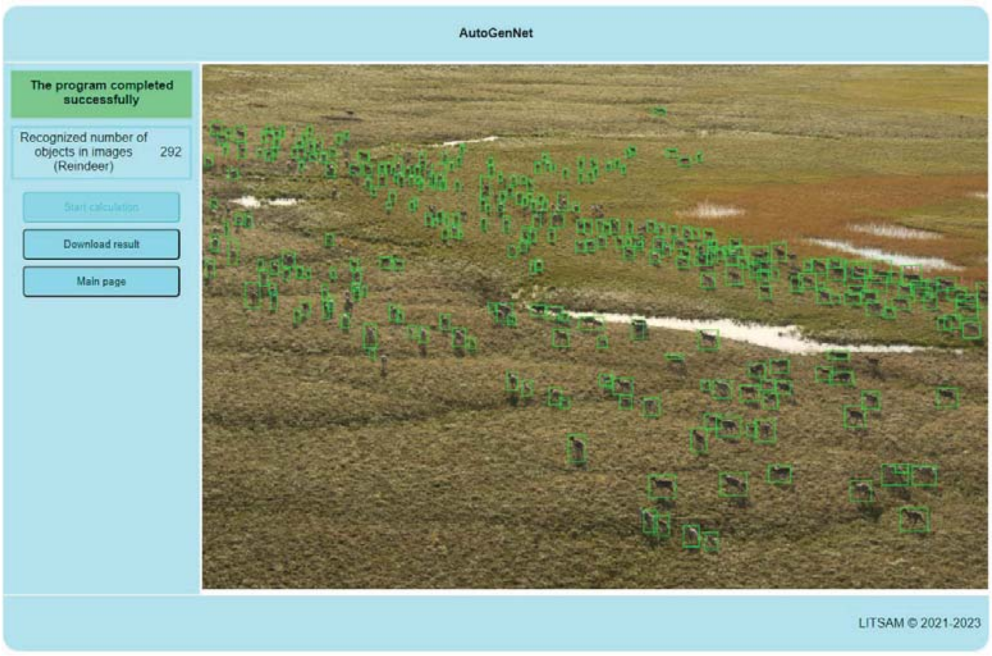
AutoGenNet
The program completed successfully
Recognized number of objects in images (Reindeer) 292
Start calculation
Download result
Main page
LITSAM © 2021-2023

Figure 6. Recognizing and counting reindeer in a large herd against a heterogeneous background

As can be seen from the model results in figure 4, despite the very small training sample, high accuracy was achieved due to transfer learning. The retrained model correctly recognized on average 82% of the reindeer on the test array. It can be seen in figure 5 that the retrained model also works well with dense clusters of animals. The image shows that the network recognizes reindeer at different distances from the camera with the same efficiency. The network is not tied to a specific resolution of objects and is able to work with distorted perspective. The recognition error for this image was about 18%. Based on the reindeer recognition results in figure 6, we can see that the retrained model can also work with images that are noisy with background objects - puddles, lakes, hillocks, polygons, etc. The background objects were never mistaken by the software recognition system for reindeer images. The recognition error in this image was about 13%.

To prove the versatility of the developed software, it was also tested on one more task. The second task was the task of brant monitoring (figure 7). MRCNN for this task was trained on only 29 images. In spite of such a small sample, the trained network correctly recognized on average 72% of brant on the test array. The results of this testing prove the versatility of the developed system and the possibility of its application for monitoring completely different species of animals.

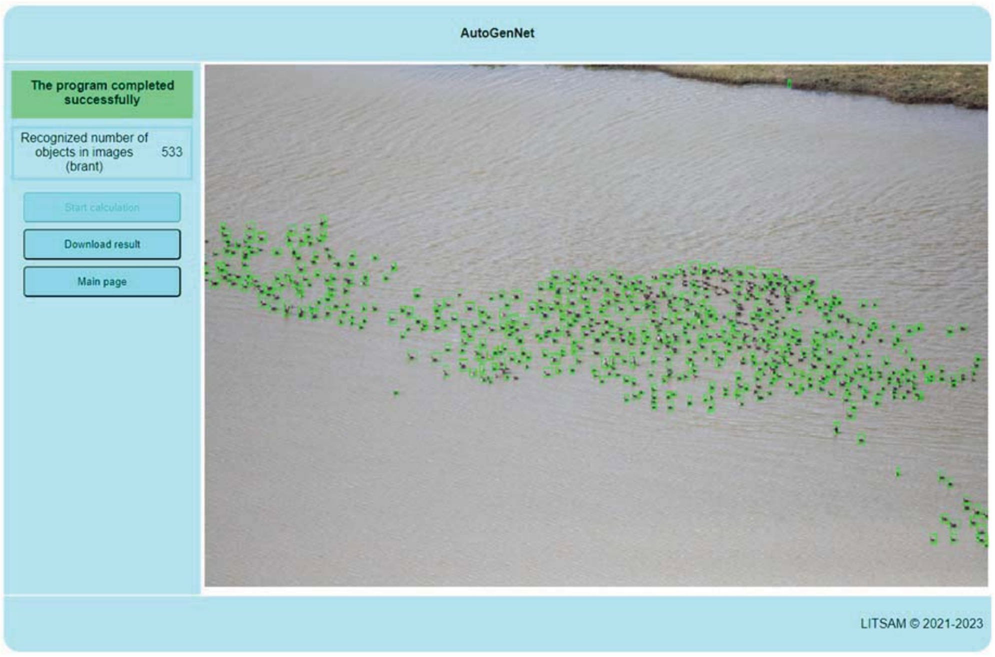
AutoGenNet
The program completed successfully
Recognized number of objects in images (brant)
533
Start calculation
Download result
Main page
LITSAM © 2021-2023

Figure 7. Result of brant recognition by the retrained MRCNN

5. DISCUSSION AND CONCLUSION

In the presented work the problem of complex synthesis of deep learning models for animal monitoring was solved. The solved problem is important for simplifying and accelerating the integration of deep learning methods in livestock industry. That in turn leads to cheaper and faster development of software complexes based on deep learning models that solve monitoring problems. The results of testing the developed software confirm the increase in the degree of automation of the process of synthesis of deep learning models. This is achieved by modifying the genetic algorithm, which performs automated selection of hyperparameters of the created models. Also, the integration of the created deep learning models is simplified due to the automatic generation of software shells for the synthesized models. At the same time, as the test results have shown, the transfer learning approach used allows achieving high accuracy of the created models even in the presence of a small training sample.

Perspectives for further development of the topic include the issue of industrial testing of the created software to assess the computational resources required for its regular operation. It is also planned to further expand the set of deep learning architectures, the work with which the created software will be able to provide. The limitations of the existing solution, to date, are related to the small number of deep learning architectures implemented in it. Nevertheless, even at the present moment the developed software provides solution of applied animal monitoring tasks that are in demand in practice.

Funding

This project was supported by funding from Russian Science Foundation grant №22-19-00767 and budget theme FFZF-2022-0004.

6. REFERENCES

Akbari, R., Mohammadi, A., Ziarati, K., 2009. *A powerful bee swarm optimization algorithm*. 2009 IEEE 13th International Multitopic Conference. https://doi.org/10.1109/INMIC.2009.5383155

Anthony, R. J., 2016. *Systems Programming: Designing and Developing Distributed Applications*. Burlington, USA: Morgan Kaufmann Publishers, 2016. – 548 p. https://doi.org/10.1016/C2013-0-13975-X

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., Hutter, F., 2019. *Auto-sklearn: Efficient and Robust Automated Machine Learning*. Automated Machine Learning, 2019, pp 113–134. https://doi.org/10.1007/978-3-030-05318-5_6

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley. P. 424.

He, K., Gkioxari, G., Dollar, P., Girshick, R., 2017. *Mask R-CNN*. arXiv:1703.06870 [cs].

Hosna, A., Merry, E., Gyalmo, J., Alom, Z., Aung, Z., Azim, M. A., 2022. *Transfer learning: a friendly introduction*. Journal of Big Data, volume 9, 2022, article number 102. https://doi.org/10.1186/s40537-022-00652-w

Jin, H., Song, Q., Hu, X., 2019. *Auto-Keras: An Efficient Neural Architecture Search System*. arXiv:1806.10282 [cs].

Kundu, N., Rani, G., Dhaka, V. S., 2021. *Seeds classification and quality testing using deep learning and YOLO v5*. ACM Int. Conf. Proc. Ser., pp. 153–160. https://dl.acm.org/doi/10.1145/3484824.3484913

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C., L., Dollar, P., 2014. *Microsoft COCO: Common Objects in Context*. arXiv:1405.0312 [cs].

McCall, J., 2005. *Genetic algorithms for modelling and optimization*. Journal of Computational and Applied Mathematics, volume 184, issue 1, 2005, pp. 205-222. https://doi.org/10.1016/j.cam.2004.07.034

Mikhailov, V. V., Sobolevskii, V. A., Kolpaschikov, L. A., 2022. *Mask R-CNN-Based System for Automated Reindeer Recognition and Counting from Aerial Photographs*. Communications in Computer and Information Sciencethis, 1562, pp. 137–151. https://doi.org/10.1007/978-3-030-98883-8_10

Nikitin, N. O., Polonskaia, I. S., Vychuzhanin, P., Barabanova, I. V., Kalyuzhnaya, A. V., 2020. *Structural Evolutionary Learning for Composite Classification Models*. Procedia Computer Science, volume 178, 2020, pp. 414-423. https://doi.org/10.1016/j.procs.2020.11.043

Noran, S. O., 2023. *Leguminous seeds detection based on convolutional neural networks: Comparison of Faster R-CNN and YOLOv4 on a small custom dataset*. Artificial Intelligence in Agriculture, volume 8, 2023, pp. 30-45. https://doi.org/10.1016/j.aiia.2023.03.002

Okinda, C., Nyalala, I., Korohou, T., Okinda, C., Wang, J., Achieng, T., Wamalwa, P., Mang, T., Shen, M., 2020. *A review on computer vision systems in monitoring of poultry: A welfare perspective*. Artificial Intelligence in Agriculture, volume 4, 2020, pp. 184-208. https://doi.org/10.1016/j.aiia.2020.09.002

Shure, L., 2020. *Building Optimized Models in a few steps with AutoML*. https://blogs.mathworks.com/loren/2020/06/13/building-optimized-models-in-a-few-steps-with-automl/ Last access: 25.07.2023

Turnip, A., Sihombing, P., Fitriatin, B. N., Simarmata, T., Tampubolon, G. M., Turmuktini, T., Joelianto, E., 2022. *Development of Irrigation System and Nutrition Prediction Model with ANFIS Method for Chili Plants*. International Journal of Artificial Intelligence, volume 20, number 2, 2022, pp. 37-47. https://doi.org/10.1088/1755-1315/1083/1/012081

Ülker, E. D., Ülker, S., 2012. *Comparison Study for Clonal Selection Algorithm and Genetic Algorithm*. arXiv:1209.2717 [cs].

Wen-Hao, S., 2020. *Crop plant signaling for real-time plant identification in smart farm: A systematic review and new concept in artificial intelligence for automated weed control*. Artificial Intelligence in Agriculture, volume 4, 2020, pp. 262-271. https://doi.org/10.1016/j.aiia.2020.11.001

Woo, M., 2020. *The Rise of No/Low Code Software Development—No Experience Needed?* Engineering, volume 6, issue 9, 2020, pp. 960-961. https://doi.org/10.1016/j.eng.2020.07.007

Zeng, Y., Zhang, J., 2020*. A machine learning model for detecting invasive ductal carcinoma with Google Cloud AutoML Vision*. Computers in Biology and Medicine, volume 122, 2020, 103861. https://doi.org/10.1016/j.compbiomed.2020.103861