Development of Architecture and Software Implementation of Deep Neural Network Models Repository for Spatial Data Analysis

Stanislav A. Yamashkin[1]
Institute of Electronic and Lighting Engineering
National Research Mordovia State University
Saransk, Russia

Anatoliy A. Yamashkin[2]
Geography Faculty
National Research Mordovia State University
Saransk, Russia

Ekaterina O. Yamashkina[3]
Institute of Information Technologies
MIREA — Russian Technological University
Moscow, Russia

Milan M. Radovanovic[4]
Geographical Institute "Jovan Cvijic"
Serbian Academy of Sciences and Arts
Belgrade, Serbia

***Abstract*—The article presents the key aspects of designing and developing a repository of deep neural network models for analyzing and predicting the development of spatial processes based on spatial data. The framework of the system operates on the basis of the MVC pattern, in which the framework is decomposed into modules for working with the system's business logic, its data and graphical interfaces. The characteristics of the developed web interfaces, a module for visual programming for editing of models, an application programming interface for unified interaction with the repository are given. The stated aim of the study determined the structure of the scientific article and the results obtained. The paper describes the functional requirements for the repository as a signific part of software design, presents the developed formalized storage scheme for neural network models, describes the aspects of development of a repository of neural network models and an API of the repository. The developed system allows us to approach the solution of the scientific problem of integrating neural networks with the possibility of their subsequent use to solve design problems of the digital economy.**

Keywords—Repository; deep learning; artificial neural network; spatial data; visual programming; software design

I. Introduction

The development of spatial data infrastructures (SDI), aimed at assessing the state of natural-socio-production systems (NSPS) and forecasting emergency processes and phenomena, plays a system-forming role in solving the problem of strengthening the connectivity of the territories of countries and regions [1]. The data integrated in systems of this class is characterized by a large volume and heterogeneity, as a result of which machine analysis algorithms become the core of systems of this class, which make it possible to solve design problems of various types in the field of analysis of spatial data on natural, social and economic objects that have a distributed geospatial organization [2]. Solving problems of classification, clustering, pattern recognition [3], decision-making and forecasting based on large arrays of spatial data [4] plays an important role in the economies of countries and regions [5]. Artificial neural networks are of great importance among models and algorithms for data analysis [6].

This article is devoted to solving the scientific problem of the formation of architecture and the development of software implementation of the repository of deep neural network models for the analysis of spatial data, integrated into a single system to support the process of making managerial decisions in the field of ensuring conditions for sustainable development of territories.

The solution to the problem of effective use of neural networks meets many unresolved challenges, an important place among which is the problem of integrating deep neural network models into a single system in order to form a convenient toolkit for specialists in the field of data analysis. Currently, the generally accepted practice is the consolidation of data analysis algorithms in repositories - information systems focused on the formation of the ability to search, store, develop and efficiently use the accumulated project-oriented solutions.

The stated aim of the study determined the structure of the scientific article and the results obtained. The paper describes the functional requirements for the repository, presents the developed formalized storage scheme for neural network models, and describes the aspects of development of a repository of neural network models and an API of the repository.

II. Related Works, Materials and Methods

The task of designing a repository of neural networks has its own specifics, due to the fact that the process of training deep models is often characterized by high requirements for computational resources [7, 8]. On the other hand, the very formation of the topology of neural networks, the selection of hyper-parameters of models is a non-trivial task that can be solved in many ways and requires the involvement of expert systems to make decisions [9, 10]. Finally, the generated neural network model repository should be deeply integrated with existing machine learning frameworks (Tensorflow [11], PyTorch [12]) to ensure high practical efficiency of the repository and reduce the gap between data scientists and software developers [13].

Currently, there are a number of examples in the field of developing deep neural network model repositories, among which Wolfram Neural Net Repository [14] and the Amazon Web Services (AWS) marketplace [15] should be highlighted. The presented repositories are characterized by different architectural and structural organization, individual software solutions. For example, Amazon Web Services matches each model with a set of different criteria and filters to find the model. At the same time, we note that when solving the problem of analyzing spatial data, its own specificity is formed, which imposes restrictions and new requirements on the implementation of the repository of neural network models presented in [16].

The development of the repository is based on the ontological model of the repository [17], which defines the principles of systematization of deep models for the analysis of spatial data by classes of problems to be solved, the nature and dimension of the analyzed data, architecture and topology, and efficiency properties (Fig. 1). The deep neural network models repository was created on the basis of the object-oriented analysis and design paradigm using the unified modeling language UML for visualization, specification, design and documentation of software systems [18].

From the point of view of software implementation, the described software system operates on the basis of the MVC pattern [19], which presupposes the decomposition of the project framework into controllers (modules designed to describe software business logic), models (components for manipulating data) and views (sets of templates for forming adaptive web interfaces). To implement the functional of visual editing of the model, the JavaScript programming language was used, the component was tested in the formation of models of neural network architectures.

An important function of the deep neural network model repository is to provide end users with different roles in the system adaptive web interfaces for quickly obtaining systematized information about the optimal deep neural network model to use, which should include a structured description, performance indicators, architectural and topological organization, and so on. the same recommendations for flexible tuning of model hyperparameters, examples of applied use in solving project-oriented problems. To organize the work of the storage of neural network models, a multi-model approach was used, and software interfaces for exchanging data with external systems are implemented on the basis of the GraphQL paradigm [20].

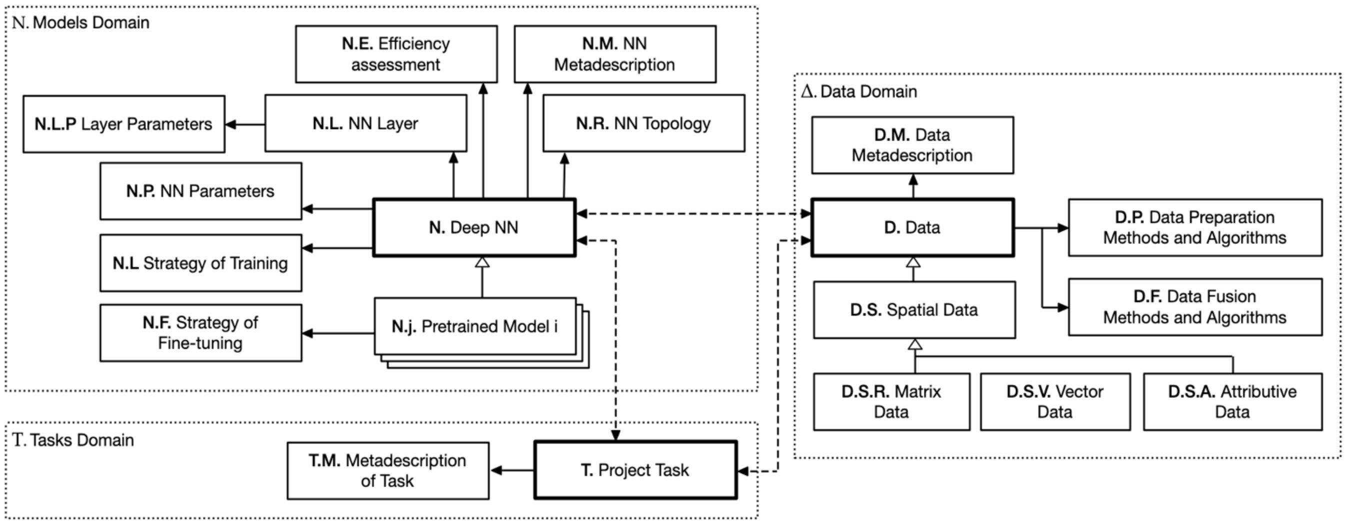

Fig. 1. Ontological Model of the Repository.

III. Research Result

A. Functional Requirements for the Repository

In order to systematize the functional requirements for the repository, we will form a UML use-case diagram (Fig. 2). The initial use case of the repository is described by the precedent "L. Authorization in the repository", the purpose of which is to ensure the differentiation of rights to read, use and edit neural network models. The designated functionality is implemented by providing a form for entering authentication data to an unauthorized user in the system.

After authorization in the system, the user gets access to individual modules of the repository based on the existing rights. From the point of view of software implementation, the described software system operates on the basis of the MVC pattern, which presupposes the decomposition of the project framework into controllers, models and views, which makes it possible to increase the cohesion of individual repository modules and reduce the coupling between them.

The Administration use-case group includes two use cases: "AU, User Management" (managing the distribution of roles, adding, editing, deleting metadata about users) and "AL, Management of logs" (forming the ability to view system logs with the ability to search and filter in order to moderate the operational processes of working with the repository).

The Neural Network Model Management use case group includes the use cases that form the core of the system. The integrating case "M. Model database management" is decomposed into the "MN. Navigation in the catalog of models" (through the filter system), including the "MR. Recommended system for model selection" (allowing to provide relevant search, selection and configuration of deep neural networks, fine-tuning of models for solving specific design problems in the field of spatial data analysis). The model data management CRUD also includes the use case "MA (Model addition)" and expanding its functionality "ME (Model modification)", which allows you to create and edit deep neural networks.

A separate description of the functional component "MV Visual editing of the model" as a constituent block of the described CRUD-subsystem, allows visualizing deep learning neural network models in the form of a graph-diagram, with the possibility of interactive online editing of the topology and model architecture through a thin client (web browser). To implement the functionality of visual editing of the model, the JavaScript programming language was used, the component was tested in the formation of models of neural network architectures presented in the Keras open neural network library (Fig. 3).

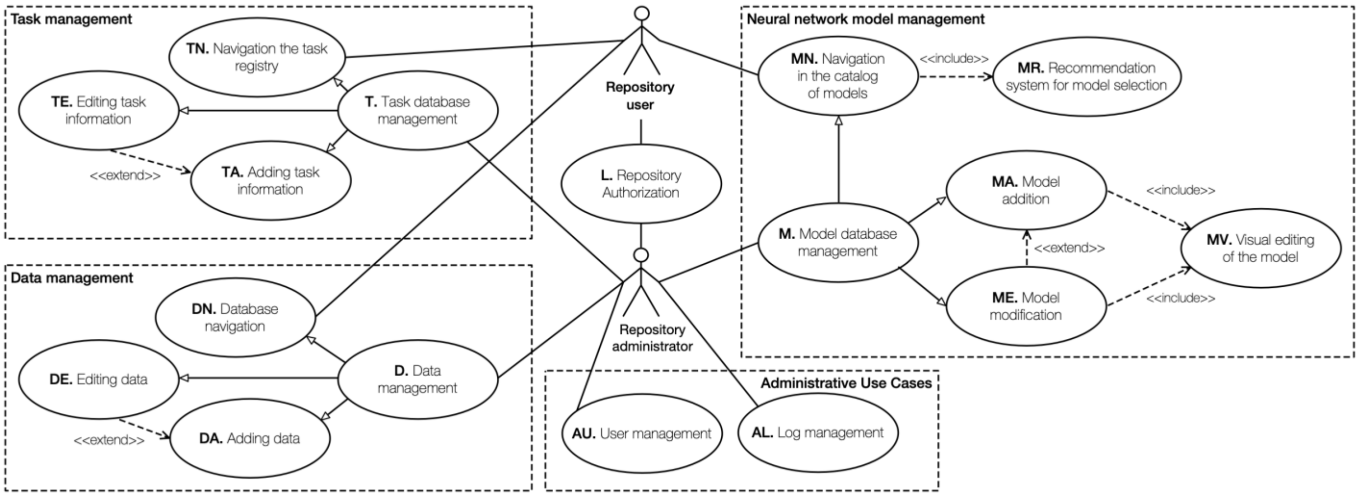
Task management
TN. Navigation the task registry
TE. Editing task information
T. Task database management
TA. Adding task information
<<extend>>
Repository user
L. Repository Authorization
Repository administrator
Data management
DN. Database navigation
DE. Editing data
D. Data management
DA. Adding data
<<extend>>
Neural network model management
MN. Navigation in the catalog of models
<<include>>
MR. Recommendation system for model selection
MA. Model addition
<<include>>
M. Model database management
<<extend>>
MV. Visual editing of the model
ME. Model modification
<<include>>
Administrative Use Cases
AU. User management
AL. Log management

Fig. 2. Deep Neural Network Model Repository use Case Diagram.

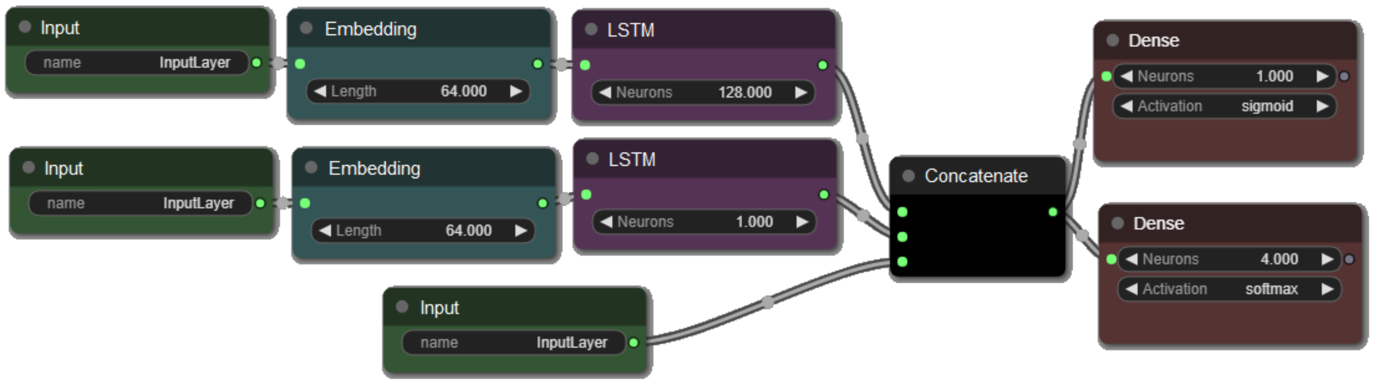

Fig. 3. Visualization of a Neural Network Model in the Form of a Graph Diagram: A Model with Three Inputs and Two Outputs.

According to the previously developed ontological model, the domain of deep neural network models should be associated with the domains of design tasks and analyzed data. Use cases corresponding to the designated subject areas are also implemented in the repository in the form of CRUD subsystems "Administration of the task base" (based on the use cases of "TN, Navigating the task registry", "TA, Adding task information", "TE, Editing task information") and "Data management" (formed by use cases "DN, Database navigation", "DA, Adding data"; "DE, Editing data").

B. Formalized Storage Scheme for Neural Network Models

The development of a formalized storage scheme for models of deep machine analysis of spatial data in the form of a meta-language made it possible to convert them into representations used by modern machine learning frameworks [4].

In order to optimize the integration processes and the practical use of deep neural network models, a formalized scheme for storing models in the form of a meta-language was developed, which makes it possible to convert them into representations used by modern machine learning frameworks (for example, Keras). Let us present the description of the designed meta-language from the point of view of the set-theoretic approach and take the set of repository models $MODELS$ (1) (the power of which is determined by the number of repository models) as a universal set:

$$MODELS = \{MODEL_i | 1 \leqslant i \leqslant N \wedge i \in \mathbb{Z}\} \quad (1)$$

The topology of the model of a specific deep neural network model $\mathrm{MODEL_i}$ can be represented in the form of a graph-scheme $\mathrm{GRAPH_i}$ and a structured meta-description $\mathrm{META_i}$ (2). In turn, the meta-description of the model is generated based on the $\mathrm{DESCRIPTORS_i}$ descriptor array (including the model name, annotated description, performance metrics, etc.) and the $\mathrm{COMPILATION_i}$ assembly parameters. The graph-diagram of the $\mathrm{GRAPH_i}$ model is a directed graph, the set of vertices $\mathrm{LAYERS_i}$ of which determines the set of layers of the deep neural network model, and the set of arcs $\mathrm{LINKS_i}$ determines the structure of the neural network, establishes directional connections between the layers.

$$MODEL_i = \langle META_i, GRAPH_i \rangle = \quad (2)$$

$$\langle \langle DESCRIPTORS_i, COMPILATION_i \rangle, \langle LAYERS_i, LINKS_i \rangle \rangle$$

A tuple of model assembly parameters $\mathrm{COMPILATION_i}$, includes an object $\mathrm{optimizer_i}$, which describes methods and algorithms for optimizing the neural network model (including stochastic gradient descent (SGD), adaptive model estimation (Adam), root mean square propagation (RMSProp) and others), as well as parameters their functioning. The loss function $\mathrm{loss_i}$ defines the parameters that the model should strive to minimize during training to solve regression and classification problems. Finally, the $\mathrm{metrics_i}$ object defines the function used to evaluate the performance of the model, while the calculated metrics values are not used in the training of the model. Various metrics can be used in the formation of models for solving problems of multiclass and binary classification, regression, segmentation. The description of the tuple of model assembly parameters from the point of view of the set-theoretic approach is as follows (3):

$$COMPILATION = \langle optimizer_i, loss_i, metrics_i \rangle \quad (3)$$

An important component of the graph model of a deep neural network model i is the $\mathrm{LAYER_{ij}}$ layer (4), which, from the point of view of a formalized description of the model, can be represented as a set of objects that determine the type and architecture of the layer $\mathrm{TYPE_{layer_{ij}}}$, a set of interfaces of the $\mathrm{INTERFACES_{ij}}$ layer, $\mathrm{PROPERTIES_{ij}}$ arguments that determine the features the functioning of the layer, descriptors of the $\mathrm{VISUAL_{ij}}$ visualization, which determine the features of the model visualization within the framework of adaptive web interfaces:

$$LAYER_{ij} = \langle TYPE_{layer_{ij}}, INTERFACES_{ij}, PROPERTIES_{ij}, VISUAL_{ij} \rangle \quad (4)$$

The $\mathrm{TYPE_{layer}}$ object defines the architecture of the layer and decisively specifies the data processing and hyperparameter settings when training the model. The base layers of the model (the BASE group) are represented by the following architectures: an input layer (Input), a fully connected layer (Dense), as well as a custom programmable layer (Custom Layer), the architectural organization of which can be specified by the user. Layers for building convolutional models (the CONVOLUTION group) includes convolutional layers that process data of different dimensions (Convolution), layers of separable convolution (Depthwise Separable Convolution), as well as layers of subdescription (Pooling). Recurrent layers are represented by architectures of fully connected recurrent layer (RNN), long short-term memory layer (LSTM), managed recurrent blocks (GRU), long short-term memory convolutional layer (Convolutional LSTM). Finally, the meta-language for describing deep neural network models supports layers of reshaping (Reshape), merging and merging (Fusion), element-wise merging based on mathematical operations (Add, Average, Maximum, Minimum, Multiply, Subtract). Finally, regularization layers) and decimation (Regularization) Thus, many categories of the architectural organization of the layers of the repository can be represented as a tuple (4):

$$TYPE_{layer_{ij}} = \langle BASE, CONVOLUTION, RECURRENT, RESHAPE, FUSION, MERGING, REGULARIZATION \rangle \quad (4)$$

The parameters of the interface $\mathrm{INTERFACES_{ij}}$ of the $\mathrm{LAYER_{ij}}$ layer (inputs $\mathrm{INPUTS_{ij}}$ and outputs $\mathrm{OUTPUTS_{ij}}$) are defined by a tuple of parameters $\langle \mathrm{name}, \mathrm{type} \rangle$, where the name object specifies the interface name, type is the data type and dimension (5).

$$INTERFACES_{ij} = \langle INPUTS_{ij}, OUTPUTS_{ij} \rangle = \langle \{INPUTS_{ij\lambda} | 1 \leqslant \lambda \leqslant \Lambda \wedge \lambda \in \mathbb{Z}\}, \{OUTPUTS_{ij\mu} | 1 \leqslant \mu \leqslant M \wedge \mu \in \mathbb{Z}\} \rangle \quad (5)$$

The set of arguments $PROPERTIES_{ij}$ sets the hyper-parameters that determine the model of the layer $LAYER_{ij}$ (6). This set of named arguments may differ for layers of different architectures and may include, for example, specifying methods and algorithms for initializing layer weights and regularization, activation functions:

$$PROPERTIES_{ij} = \langle initializer,\ activation,\ regularization, \{keywordArguments_k \mid k \in \mathbb{Z}\}\rangle \quad (6)$$

Finally, in order to provide visualization of deep neural network models within the framework of adaptive web interfaces, the meta-language for describing the model involves storing the parameters of the $\mathrm{VISUAL_{ij}}$ group, which are not responsible for the functional and qualitative features of the neural network, but determines the aspects of its visual graphical display (7). The object of the $\mathrm{META_{ij}}$ group is characterized by the title information $\mathrm{title_{ij}}$ and the $\mathrm{scheme_{ij}}$ rendering scheme, which determines the color scheme of the layer within the framework of web interfaces and other styles. The elements of the $\mathrm{POSITION_{ij}}$ tuple define the $\mathrm{top_{ij}}$ and $\mathrm{left_{ij}}$ coordinates of the layer's location on the render canvas, and the $\mathrm{SIZE_{ij}}$ tuple defines its $\mathrm{width_{ij}}$ and $\mathrm{height_{ij}}$ dimensions.

$$VISUAL_{ij} = \langle META_{ij}, POSITION_{ij}, SIZE_{ij}\rangle = \langle\langle title_{ij}, scheme_{ij}\rangle,\ \langle top_{ij}, left_{ij}\rangle,\ \langle width_{ij}, height_{ij}\rangle\rangle \quad (7)$$

To combine the $\mathrm{LAYERS_i}$ layers of the i model, a set of $\mathrm{LINKS_i}$, links is introduced into a single model, in which each object is characterized by a set of parameters defining the source layer ($\mathrm{LAYER_{origin}}$) and its output interface ($\mathrm{OUTPUT_{origin}}$), as well as the destination layer ($\mathrm{LAYER_{target}}$) and its input interface ($\mathrm{INPUT_{target}}$) (8).

$$LINK_{ij} = \langle\langle LAYER_{origin}, OUTPUT_{origin}\rangle,\ \langle LAYER_{target}, INPUT_{target}\rangle\rangle \quad (8)$$

An application programming interface designed on the basis of the REST architectural pattern and GraphQL paradigm provides the possibility of a unified interaction in order to exchange data with a repository for importing and exporting neural network models and obtaining information about them).

C. Development of a Repository of Neural Network Models and an API of the Repository

A comparative analysis of documentation, use cases, market needs made it possible to identify several databases suitable for solving the problem of storing neural network models and built on the basis of various paradigms: PostgreSQL (an object-relational DBMS that can be successfully used for structured presentation of data from different knowledge domains of a repository) , Neo4j (graph storage that can be used to store data about the topology of neural network models), InfluxDB (a database for storing time series that allows you to keep temporary training statistics, track progress and save the trained weights during training).

Each individual paradigm for organizing a repository of deep machine learning models does not answer all the questions that arise when solving the problem of systematizing information from data domains, deep neural network models and design problems. A comprehensive answer to this problem can be provided by multi-model database management systems, which are hybrid storages that can be centralized in the data center, or presented on a cloud scale, the operation of which is based on the superposition of the capabilities of different classes of DBMS. The result of the competent use of multi-model data management systems for the repository of deep neural network models should be a purposeful enhancement of the qualitative characteristics of the generated storage of neural networks, including scaling and modularity, fault tolerance and reliability.

To solve the problem of deploying an application programming API in order to exchange data about a specific neural network for use in further computations, it is proposed to use the GraphQL paradigm, which makes it possible to form a group of microservices that are weakly related to each other. This allows greater scalability, more precise customization for user requests and reduces the load on the global nodes of the system.

The application programming interface of the repository of neural network models provides the possibility of unified interaction with the system for data exchange in order to import and export neural network models and obtain information about them). It should describe the ways (that is, a set of functions, classes, constants, or structures) through which a software system can interact with a model store. The API interface, implemented using GraphQL technology, defines the syntax that describes the format for requesting data from the repository server, and also provides an environment for executing such requests. The advantages of using this technology are as follows: GraphQL allows the user to specify exactly what data he needs without reloading the request with unnecessary information; making it easy to combine data from different sources.

When organizing a microservice architecture, it becomes possible to create a system of methods and algorithms for obtaining data of various types, thereby reducing the load on the system as a whole. GraphQL is also capable of providing seamless access between existing services. The convenience of using this technology is that after setting up GraphQL, the entire set of microservices acts as a single whole, and GraphQL regulates the flow of requests. This allows you to get a fully automated system that does not require developers to write additional interfaces when interacting between microservices and system interfaces.

IV. Conclusion

The article describes a project devoted to solving the scientific problem of accumulating and systematizing deep neural network models by designing and developing a repository of learning algorithms for analyzing and predicting the development of spatial processes.

From the point of view of software implementation, the framework of the system operates on the basis of the MVC pattern, which involves the decomposition of the project framework into controllers, models and views. Emphasis is made on the development of adaptive web interfaces that allow using the repository using a computer connected to the Internet.

The visual model editing module allows you to visualize neural network models in the form of a graph diagram, with the possibility of interactive online editing of the topology and model architecture. An API based on the GraphQL paradigm provides a unified interface to communicate with the repository to import, export, and retrieve information about models.

The designed architecture of the repository of deep neural network models and its software implementation allows us to approach the solution of the scientific problem of integrating neural networks, pre-trained models with the possibility of their subsequent use to solve design problems of the digital economy.

Acknowledgment

This work was supported by Grant of the President of the Russian Federation under Project no. MK-199.2021.1.6.

References

[1] M. F. Goodchild, "Citizens as voluntary sensors: spatial data infrastructure in the world of Web 2.0," International journal of spatial data infrastructures research, vol. *2, no.* 2, pp. 24–32, 2007.

[2] R. A. Schowengerdt, Remote sensing: models and methods for image processing, 3rd ed. Orlando, FL, USA: Academic Press, 2006, pp. 387–456.

[3] Cham, D. D., Son, N. T., Minh, N. Q., Thanh, N. T., Dung, T. T., "An analysis of shoreline changes using combined multitemporal remote sensing and digital evaluation model," Civil Engineering Journal, vol. 6, no. 1, pp. 1-10, Jan. 2020, DOI. 10.28991/cej-2020-03091448.

[4] Damuluri, S., Islam, K., Ahmadi, P., & Qureshi, N., "Analyzing navigational data and predicting student grades using support vector machine," Emerging Science Journal, vol. 4, no. 4, pp. 243-252, Aug. 2020, DOI. 10.28991/esj-2020-01227.

[5] Hammal, S., Bourahla, N., & Laouami, N., "Neural-network based prediction of inelastic response spectra," Civil Engineering Journal, vol. 6, no. 6, pp. 1124-1135.

[6] X. X. Zhu, D. Tuia, L. Mou, G. S. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: A comprehensive review and list of resources," IEEE Geoscience and Remote Sensing Magazine, vol. 5, no. 4, pp. 8–36, Oct. 2017, DOI. 10.1109/MGRS.2017.2762307.

[7] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," IEEE Geoscience and Remote Sensing Magazine, vol. 4, no. 2, pp. 22–40, Jun. 2016, DOI. 10.1109/MGRS.2016.2540798.

[8] C. Tao, H. Pan, Y. Li, and Z. Zou, "Unsupervised spectral–spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification," IEEE Geoscience and Remote Sensing Let., vol. 12, no. 12, pp. 2438–2442, Dec. 2015, DOI. 10.1109/LGRS.2015.2482520.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436, May 2015, DOI. 10.1038/nature14539.

[10] W. Li, H. Liu, Y. Wang, Z. Li, Y. Jia, and G. Gui "Deep learning-based classification methods for remote sensing images in urban built-up areas," IEEE Access, no. 7, 36274-36284, Mar. 2019, DOI. 10.1109/ACCESS.2019.2903127.

[11] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, and X. Zheng, "Tensorflow: A system for large-scale machine learning," 12th {USENIX} symposium on operating systems design and implementation, pp. 265-283, 2016.

[12] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," arXiv preprint arXiv:1912.01703, 2019.

[13] M. Xiu, Z. M. J. Jiang, B. Adams, "An Exploratory Study on Machine-Learning Model Stores," arXiv preprint, 1905.10677, 2020.

[14] Wolfram Repository of Neural Network Models, Apr. 2021, [online] Available: http://resources.wolframcloud.com/NeuralNetRepository.

[15] Amazon Web Services Marketplace – Machine Learning. Accessed: May. 2021. [Online]. Available: https://aws.amazon.com/marketplace/solutions/machinelearning.

[16] E. Yamashkina, S. Kovalenko, and O. Platonova, "Development of repository of deep neural networks for the analysis of geospatial data," IOP Conference Series: Materials Science and Engineering, vol. 1047, no. 1, 012124, Feb. 2021.

[17] S. A. Yamashkin, A. A. Kamaeva, A. A. Yamashkin, E. O. Yamashkina, "Matters of Neural Network Repository Designing for Analyzing and Predicting of Spatial Processes," International Journal of Advanced Computer Science and Applications, vol. 12, no. 5, pp. 17–22, May 2021.

[18] M. Fowler, "UML distilled: a brief guide to the standard object modeling language," Addison-Wesley Professional, 2004.

[19] D. P. Pop, A. Altar, "Designing an MVC model for rapid web application development," *Procedia Engineering*, vol. *69*, pp. 1172–1179, 2014.

[20] O. Hartig, J. Pérez, "Semantics and complexity of GraphQL," In Proceedings of the 2018 World Wide Web Conference, pp. 1155-1164, Apr. 2018.