Implementation of artificial intelligence for analysis of long-term climate variability

E V Fedotova, V S Luferov
Moscow Power Engineering Institute, Krasnokazarmennaya 14, Moscow

Abstract. A simulation workflow is formulated to facilitate an application of modern data-driven approaches to climate problems by considering both theoretical aspects and modern tools of program implementation. A real supervised classification problem is considered as a use case. A convolutional neural network based model is developed to make the supervised classification of large-scale atmospheric circulation regimes. It has been found that a key prerequisite to ensure the success of the thus developed deep learning model is the proper matching of the data preprocessing and model architecture with the nature of the problem being considered. Particularly, applying of the seasonal anomalies transformation, tuning of the convolutional filters dimension and batch size adjustment have been found to be crucial to ensure a satisfactory accuracy of the regime recognition and a stable operation of the model. The generation of feature maps is demonstrated to be useful both as an interpretability tool and a heuristic approach to tuning of hyperparameters.

1. Introduction
The artificial intelligence (AI) methods are continuously gaining popularity during the last decades in an increasing range of applications. An attempt to use AI approaches in climate research are still quite limited. However, the modern methods of nonlinear modeling (e.g. neural networks) seem to suit perfectly well the nature of the climate problems.

First of all, the nonlinear properties of the climate system itself restrict the applicability of the linear approaches to climate modeling. Indeed, a wide range of linearization techniques may be perfectly appropriate for many practical problems. Unfortunately, the nonlinear effects not very well captured by the linear models are often of particular importance, which makes the nonlinear approaches preferable over simpler alternatives in such cases.

Second, an incredible complexity of the climate system itself determines difficulties of the conventional analysis. A classical illustration of this statement is interpretation of the eigenvalues-based climate indices. Understanding of the exact physical processes which determine the influence of the atmospheric teleconnections on regional climate is not a trivial task. The AI approaches make it possible to emulate observed interconnections and obtain some meaningful outputs even if the interpretation of these observed effects is not quite obvious.

Finally, the AI methods may alleviate one of the most typical problems of the climate research, which is only decent quality of the available data, especially on the long-term horizons. Indeed, data management is essential for any kind of models, but the adaptive features of the nonlinear AI methods determine certain toleration towards the non-perfect data quality.
The development of complex AI models is not a trivial task and requires combining an understanding of the object area with computational expertise and programming skills. Subject-specific requirements are very often neglected if the AI methods are discussed. At the same time, interference of the problem features with the requirements of the modeling infrastructure is nowadays at the very edge of the data science research.

The purpose of our work is to define a more or less general workflow to facilitate application of the AI to climate research. The next section will give a very brief overview of the available modern modeling and implementation approaches. Then we will introduce a general outline of the modelling workflow and demonstrate how to use it for a solution of a particular research problem.

2. State-of-the-art
The standard de facto of the modern data-driven modeling including artificial intelligence is an open source paradigm. The free and open source software (FOSS) movement has been progressing since the late 1970s. The creation of the Free Software Foundation in 1985 and the Open Software Foundation in 1998 greatly contributed to the wide use of open collaboration patterns and significantly changed the software development practices [1].

Rapid development of the data science software is based nowadays almost exclusively on the FOSS solutions, such as R and Python languages, PostgreSQL and Mongo databases, Kafka and Hadoop platforms. The tools relevant to the application of AI climate problems are primarily the geospatial and machine learning tools, whose main options will be briefly reviewed.

2.1. Geospatial tools
The FOSS geospatial tools began to flourish about fifteen years ago. Although the history of the oldest initiatives may span at least to the beginning of 1980, with the development of the geographical information system GRASS (Geographic Resources Analysis Support System) [2] being one of the most impressive examples. However, in 2000 extensive growth of the open geospatial community started, and the R-language geographical data community r-sig-geo [3], the OpenStreetMaps [4] project, and the Open Source Geospatial Foundation (to mention only few initiatives) were launched.

Nowadays almost every geographical computational problem can be effectively solved with the existing FOSS approaches, from assessment of the reanalysis datasets to making complex spatial-temporal interpolations and comprehensive data analysis. Indeed, the deep learning approaches naturally not belonging to the typical geospatial approaches are rarely included in the existing geospatial toolsets and usually address very specific problems.

2.2. A subsection
A wide range of the deep-learning computational methods is available via FOSS-projects. Actually, the FOSS paradigm seems to be the only approach that is flexible enough to continuously incorporate newly emerging AI methods. Wide use of the AI open source projects started in the middle of 2000 [5] and intensified a few years ago when the leading modern AI tools, such as Thensorflow [6], PyThorch [7], Caffe [8], H2O [9], and many others were created. The use of the existing AI tools has increased dramatically by their integration with the available programming approaches. The most influencing project of this kind is nowadays the Keras project [10], which incorporates a number of deep learning techniques and provides all necessary infrastructure components to run AI modeling. The Keras is written in Python, but may be imported to a number of different languages, such as R or Java, making the integration between the data processing and AI simulation as smooth as possible.
3. Problem definition

3.1. Modelling workflow

The following steps may be proposed as a more or less universal approach to the research-minded AI modeling keeping in mind the available implementation options:

1. data management;
2. data preprocessing;
3. elaboration of the AI model architecture;
4. tuning of the model hyperparameters;
5. model learning and testing;
6. analysis and interpretation of the modeling output;
7. reporting and sharing of the results.

3.1.1. Data management and preprocessing. The data management, which appears quite trivial, is often the most time-consuming step. The reasons are, first, that the data amount necessary to run the data-driven models properly should be as large as possible. Another point is that the definition of the computational problem should always be fitted to the available data and the research question of interest. Preference of one algorithm over another one cannot be stated formally according to the famous “no free lunch” theorem [11]. That is why the subject expertise is a key advantage for the success of AI modeling for a particular application. Analysis of the data availability is a part of the decision making process when modeling approaches should be defined. An initial exploratory analysis and taking into consideration simpler modeling options often make sense.

Worth mentioning is the technical aspect of data organization. The two main data structures are competing options for a vast majority of the data science problems, namely, databases (relational ones, such as PostgreSQL, or document-oriented ones, such as MongoDB [11]) and dataframes which are in-built classes of R and Python. Theoretically, the databases ensure better memory management. In practice, this advantage is often leveled by the operating system memory management, which allows for a highly satisfactory performance of dataframes. The main advantage of the databases seems to be facilitation of the data reuse, for example, for collaborative data processing or when the data are intended for use by a web-application.

3.1.2. Model development and run. The AI models are quite complicated and offer many degrees of freedom. Search for a suitable model architecture requires some trials and errors. Often it makes sense to compare the results of different modelling approaches or use an ensemble of different models.

Adjustment of the proper AI model parameter values is referred to as hyperparameter tuning. A number of frameworks are available to automate this adjustment process. However, as it will be demonstrated, problem specific is essential for this purpose.

Performance may be an issue even for quite simple AI models. Parallel computing became a classical approach to overcome this performance limitation even for moderate computational resources at hand [13]. We will refer to practical implementation of parallel computations using the existing AI toolsets in the section on modeling.

3.1.3. Interpretation of the results. Proper interpretation of the output is one of the greatest challenges that nowadays restricts the use of AI results [14]. The AI models are usually too complicated to allow for a clear problem-related explanation related to all modelling details. However, some kind of proof is highly desirable to ensure reasonable behaviour of the model to use modelling results for practical purposes.
We will demonstrate a simple approach to the interpretation of particular modeling results in the next section, and show that such explanatory approaches may give some insights for improving the model structure.

3.1.4. Reporting and sharing of results. Sharing of the modelling results is perfectly consistent both with the open source paradigm and the reproducibility of the research. The modern data-science tools propose a number of sharing concepts including package maintaining [15], code and data repositories, development of web-applications [16], interactive frameworks [17], and collaborative platforms [18].

3.2. Problem being considered
The formulated workflow was implemented to automate the recognition of the global-scale atmospheric circulation regimes following, as much as possible, the code reuse principles. The main motivation of the problem was to justify the use of the classical subjective atmospheric classification as an analysis tool, which is particularly suitable for the interpretation of the output of global climate models.

One of the greatest challenges of the modern climate science is to understand the physical mechanisms behind the regional manifestation of the global climate processes. Even the most widely studied teleconnection mechanisms seem to be not quite understood yet and may be explained by different research groups in not a fully consistent way. Particularly, the long-term trends of the regional atmospheric circulation may differ dramatically when different analysis approaches are used [19].

A decisive advantage of the subjective classification schemes towards the global atmospheric circulation is a clear physical meaning of each particular regime including linking of the regional and global climate processes. It may be even more important that the global circulation is, in fact, a manifestation of the large-scale turbulence structures, which are known to be much more stable compared to the regional-scale structures usually assessed by the explanatory regional climate analysis.

4. Methods and implementation
The problem was addressed as an image recognition task. A supervised classification approach was used to emulate classification decisions normally made by an expert. This problem definition is very much like image classifications, which is quite a classical computer vision problem with a wide range of the existing development tools. Dzerdzevskii’s classification scheme was used [20] as the most well-developed subjective circulation scheme available up to now.

4.1. Data management
The sea-level atmospheric pressure was utilized as a classification base following the principles of manual classification. NOAA Twenty Century reanalysis [21] was used as the pressure input data, being the longest available homogeneous dataset on the global atmospheric circulation. The raw data netCDF archives were organized as the PostgreSQL database (Figure 1). The main motivation for this design was an option to reuse these data as a part of collaboration process.

4.2. Data preprocessing
The database was labeled according to the circulation calendar created by N.K. Kononova and N.N. Cherenkova according to the methodology proposed by B.L. Dzerdzevskii [20]. The dataset was shuffled and split into a train part and a validation part. Four cumulative circulation classes (zonal, disturbed zonal, meridional northern, and meridional southern ones) were considered as far neglecting with finer classification types to ensure that the learning dataset is balanced. The improvement of the detailedness level will be assessed at the next iteration of the developed model. It should be noted that even the assumed quite rough cumulative classification approach may give meaningful results [22].
The specific preprocessing procedures were essential for ensuring stable operation of the model. The seasonal pressure patterns in raw data were found to overshadow the effect of the circulation regime. Thus, calculation of the seasonal anomalies was applied to the input pressure fields. The final data preparation step was standard hypercube transformation.

### 4.3. Model architecture and implementation

The classical architecture for the image classification tasks is based on different modifications of the convolution neural network (CNN). This is one of the leading approaches of computer vision research, which allows using a wide range of available programming approaches to implement our model. We decided to use the above Keras library [10], which makes possible a seamless integration of the data preprocessing and model tuning into a Python code single block. A TensorFlow machine learning library was used as a computational backend of the model.

Parallel computation was implemented with in-built Keras parallel computing options which use the CUDA programming model via graphical processing units (GPUs). Using a GPU for parallelizing allows one to accelerate the model learning up to 20–40 times as compared with the central processing unit, making it possible to run the model on a personal computer with reasonable computational speed, which is few hours on a modern high-performance video card for a single model run.

Usually CNNs are used to process pictures consisting of three layers, namely, red-, green-, and blue- color channels. Our two-dimensional pressure fields consist of a single layer corresponding to the pressure value distribution. The inlet channel of the CNN is a matrix presented as a multidimensional tensor. Each convolutional layer transforms the input tensor, and the value of channels may be different for each next layer. The number of the processing channels equals the number of feature maps extracted by the layer and should be set separately for each layer. The feature maps are the calculated response of the convolution artificial neural network to input vectors.

The convolution artificial neural network is based on the convolution operation. The convolution is a linear transformation of the input data with the following properties:

- keeping the structure of the input data due to the application of input data to each image part separately, namely, the order of elements in the one-dimensional case, mutual arrangement of pixels for a two-dimensional image;
- the property of a sparse layer, because the value of each neuron of the next layer depends only on a small fraction of the input neurons, opposite to a fully-connected network, where each neuron depends on all neurons of the previous layer;
- reuse of the same neuron weights many times, because they are applied repeatedly to different input areas.

The convolution layers are usually followed by a transformation with an activation function. The deep neural networks a non-linear rectified linear activation function (ReLU).
We have implemented the CNN model using the modern VGG16 architecture [23], which is known to ensure high classification accuracy for the image recognition problems. The model structure selected after some trials and errors is presented in Table 1. The model hyperparameter tuning could be automated with the Keras-compatible tools. However, keeping in mind that it would require to put aside a separate dataset purely for tuning purposes, we decided to tune the CNN parameters manually, which gives a chance to study the interaction between the model and the problems being considered more carefully.

The CNN model is based on combinations of the convolution and MaxPoling pairs. Two such blocks were found to be sufficient for our problem. The input layer is the ZeroPadding layer with a resolution of 91x180 equal to the resolution of the original pressure fields. The ZeroPadding layer forms a one-pixel wide empty frame across the input image to guarantee that the entire original image area will be covered by the convolution procedure.

The CNN model was found to work much more stably when the second convolution layer is followed by a MaxPooling layer with a 7x7 filter, instead of 3x3, allowing one to resolve cyclone-like structures in the pressure field. This idea of model design was the result of an iterative model improvement, and was proposed based on examination of the extracted feature maps.

The final convolution block is followed by a dropout regularization layer with a probability value of 0.25. The dropout layer is intended to increase the generalization abilities of the CNN model [24]. The essence of the dropout approach is including a random neuron shutdown with a given probability. In addition, the dropout procedure increases the stability of the networks to flaws contained in the input data. The concluding layer of the convolution blocks is the Flatten layer. The layer performs a conditional operation pulling the convolution results into the vector whose values are fed into the fully-coupled perceptron, that is, a dense layer with the neurons number 512. This is followed by one more dropout regularization layer with the same probability of 0.25. The output of the CNN-model is a full-connected dense layer with the number of outputs equal to the number of the classified atmospheric circulation types, which is four. The output of the model is confidence for each of the possible types calculated for a particular input pressure field (Figure 2).

| Layer               | Output Shape | Parameter |
|---------------------|--------------|-----------|
| ZeroPadding2D       | (93, 182, 1) | 0         |
| Conv2D              | (93, 182, 128) | 1280     |
| MaxPooling2D        | (31, 60, 128) | 0         |
| Conv2D              | (29, 58, 64)  | 73792     |
| MaxPooling2D        | (4, 8, 64)    | 0         |
| Conv2D              | (2, 6, 32)    | 18464     |
| Dropout             | (2, 6, 32)    | 0         |
| Flatten             | (384)         | 0         |
| Dense               | (512)         | 197120    |
| Dropout             | (512)         | 0         |
| Dense               | (4)           | 2052      |

The input layer is followed by the Conv2D convolution layer, which produces 128 feature maps applying a 3x3 convolution filter followed by the ReLU activation function. Then the feature maps are transferred to a MaxPooling layer, which extracts the maximum element from each 3x3 square formed by the convolutional filters. The MaxPooling layer is not learned and serves to sub-sample the most important features extracted by the convolution layer. Besides, it serves as a compression procedure decreasing the size of the model.

The CNN model was found to work much more stably when the second convolution layer is followed by a MaxPooling layer with a 7x7 filter, instead of 3x3, allowing one to resolve cyclone-like structures in the pressure field. This idea of model design was the result of an iterative model improvement, and was proposed based on examination of the extracted feature maps.

The final convolution block is followed by a dropout regularization layer with a probability value of 0.25. The dropout layer is intended to increase the generalization abilities of the CNN model [24]. The essence of the dropout approach is including a random neuron shutdown with a given probability. In addition, the dropout procedure increases the stability of the networks to flaws contained in the input data. The concluding layer of the convolution blocks is the Flatten layer. The layer performs a conditional operation pulling the convolution results into the vector whose values are fed into the fully-coupled perceptron, that is, a dense layer with the neurons number 512. This is followed by one more dropout regularization layer with the same probability of 0.25. The output of the CNN-model is a full-connected dense layer with the number of outputs equal to the number of the classified atmospheric circulation types, which is four. The output of the model is confidence for each of the possible types calculated for a particular input pressure field (Figure 2).
4.4. Model learning and validation

The model neuron weights were set up during the training process using the algorithm of error backward propagation. The batch size parameter governed by the learning process was optimized (Figure 3).

This parameter determines the elements number in a training sample which is used to calculate the weights during a single iteration of the training process. Tuning of the bath size allowed us to balance the computational cost and the training error. The batch size of 64 samples was set for further calculations based on the optimization performed.

4.5. Interpretation of the results

The model was found to work on the global data better than on hemispherical ones, which is quite a surprising finding, since the used classification scheme was initially developed for the Northern Hemisphere only. Indeed, the processes in both Earth’s hemispheres are tightly interconnected. However, a kind of assessment of the calculation processes inside the model was necessary to ensure that the model results are meaningful.

Extraction of the feature maps was taken as an explanatory approach. The inlet convolution layers extract the general structure of the classified images, while the deeper layers are responsible for the extraction of finer details. An example of a feature map constructed for the meridional north type is presented in Figure 4 (cyclone-like structures, as it could be expected; 7x7 filters).
5. Discussion and summary
We have demonstrated an application of a deep neural network to Dzerdzeevskii’s subjective classification of atmospheric circulation regimes. The above developed convolutional neural network model was found to reproduce the classification group with a confidence higher than 90%. This means that the subjective classification is perfectly reproducible and can be used as an explanatory tool in analyzing the global climate processes, including the interpretation of the results of global climate models. The recognition of the finer circulation types should be the next step in the model development.

Figure 4. Example of the feature maps extracted by different layers of the model for the meridional circulation north type: a – the first convolutional layer, b – the deepest convolutional layer.

Despite the fact that the use of standard libraries has been found to work properly for the considered problem, we have also found that proper accounting for the specific problem features is essential for the successful operation of the model. This may seem somewhat contradictory to the common approach of the most AI-related works, which rarely refer to the nature of the modeled problem and usually are restricted to the use of some standard datasets to test the proposed computational methods.

The key approaches used to fit the modeling workflow to our problem are related to data preprocessing and model tuning. Particularly, we have demonstrated that the calculation of seasonal anomalies works well to separate the regime-related pressure patterns. The adjustment of the model convolution filters is another approach to ensure the proper recognition of details of the global atmospheric circulation. The above-presented model was constructed with quite reasonable efforts thanks to the incorporation of existing computational toolsets according to the described step-by-step workflow. Further work will include the sharing of the model as a web-application. So far, the source code of the project, the results of the model tuning, and the modeling results have been made freely available under GPL v3.0 license in the repository https://gitlab.com/luferov/ml_atm.

Acknowledgments
The authors highly acknowledge the insightful discussions of meteorological regimes with N.K. Kononova.

This work was supported by the Russian Science Foundation (project No. 18-79-10255).

References
[1] Fogel K 2006 Producing Open Source Software How to Run a Successful Free Software Project (Boston: O'Reily)
[2] Neteler M, Bowman M H, Landa M, Metz M 2012 GRASS GIS: A multi-purpose open source GIS Environmental Modelling and Software 31 124-30
[3] Bivand R 2014 GeoComputation and Open-Source Software: Components and Software
Component Stacks (preprint SAM 23 2011, Norwegian School of Economics)

[4] Curran K, Crumlish J, Fisher G OpenStreetMap 2012 *International Journal of Interactive Communication Systems and Technologies* 2 1

[5] Ketkar N Introduction to Theano 35-61 in Ketkar N 2017 Deep Learning with Python (New York: Apress)

[6] Abadi M, Isard M, Murray D G 2017 A computational model for TensorFlow: an introduction Proceedings of the 1st ACM SIGPLAN International Workshop on Machine Learning and Programming Languages 1-7 (New York: ACM)

[7] Ketkar N 2017 Introduction to PyTorch 195-208 in Ketkar N 2017 Deep Learning with Python (New York: Apress)

[8] Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T 2014 Caffe: Convolutional Architecture for Fast Feature Embedding 2014 MM '14 Proceedings of the 22nd ACM international conference (Orlando, Florida, USA) 675-678 (New York: ACM)

[9] Laundry M 2018 Machine Learning with R and H2O (Mountain View, CA: H2O.ai, Inc)

[10] Chollet F 2018 Deep Learning with Python (Shelter Island: Manning)

[11] Wolpert D H, Macready W G 2005 Coevolutionary free lunches *IEEE Transactions on Evolutionary Computation* 9 6

[12] Landset S, Khoshgoftaar T M, Richter A N, Hasanin T 2015 A survey of open source tools for machine learning with big data in the Hadoop ecosystem *Journal of Big Data* 2 24

[13] Shallue C J, Lee J, Antognini J, Sohl-Dickstein J, Frostig R, Dahl G E 2019 Measuring the Effects of Data Parallelism on Neural Network Training *Journal of Machine Learning Research* 20 1-49

[14] Ribeiro M, Singh S, Guestrin C 2016 Why Should I Trust You?: Explaining the Predictions of Any Classifier *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations* (San Diego, California: Association for Computational Linguistics)

[15] Lovelace R, Nowosad J, Muenchow J 2019 Geocomputation with R (Indianapolis: Chapman and Hall/CRC)

[16] Moraga P 2017 SpatialEpiApp: A Shiny web application for the analysis of spatial and spatio-temporal disease data *Spatial and Spatio-temporal Epidemiology* 23 47-57

[17] Kluyver T *et al* 2016 Jupyter Notebooks—a publishing format for reproducible computational workflows *Positioning and Power in Academic Publishing: Players, Agents and Agendas. Proceedings of the 20th International Conference on Electronic Publishing* (Tokyo: IOS Press)

[18] Lokhmotov A, Fursin G 2016 Benchmarking, autotuning and crowdtuning OpenCL programs using the Collective Knowledge framework 2016 *Proceedings of the 4th International Workshop on OpenCL* (New York: ACM)

[19] Cahynova M, Huth R 2016 Atmospheric circulation influence on climatic trends in Europe: An analysis of circulation type classifications from the COST733 catalogue *International Journal of Climatology* 36 2743–60

[20] Kononova N K 2018 Types of the global atmospheric circulation: monitoring and retrospective assessment results of 1899-2017 *Fundamental and Applied Climatology* 3 108-23

[21] Compo G P *et al* 2011 The Twentieth Century Reanalysis Project Quarterly Journal of the Royal Meteorological Society 137 654 1-28

[22] Kononova N K, Pimankina N V, Yeriskovskaya L A, Jing L, Weijia B, Shiyin L 2015 Effects of atmospheric circulation on summertime precipitation variability and glacier mass balance over the tuyuksu glacier in Tianshan mountains, Kazakhstan *Journal of Arid Land* 7 5 687-95

[23] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A 2015 Going deeper with convolutions Preprint arXiv:1602.07261
[24] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R 2014 Dropout: A simple way to prevent neural networks from overfitting *Journal of Machine Learning Research* **15** 1929-58