Physics – Uspekhi **68** (4) 374 – 392 (2025)

PHYSICS OF OUR DAYS

PACS numbers: 07.05.Mh, **84.35.+i**, **96.60.–j**

ON THE 270th ANNIVERSARY OF M.V. LOMONOSOV MOSCOW STATE UNIVERSITY (MSU)

Machine learning methods in solar physics

E.A. Illarionov

DOI: https://doi.org/10.3367/UFNe.2025.02.039872

Contents

Abstract. The development of machine learning methods and their success in a wide range of problems have had a significant impact on the design and implementation of solar physics research. Large data sets have emerged as an intrinsic value in which the efforts of experts and significant technological resources are invested. The research itself has acquired an interdisciplinary nature and is concentrated around advanced computing centers. Large-scale problems can now be posed whose mathematical formulation was unclear yesterday. In this review, we present the main ideas underlying modern machine learning models, the databases prepared for machine learning tasks, and data processing tools. A major part of this review is devoted to a discussion of models proposed in the context of specific solar physics problems and their extension to other applications.

Keywords: solar physics, solar activity, machine learning, databases

1. Introduction

Machine learning methods are becoming an increasingly popular and effective tool in data processing. Perhaps one of the most notable results is provided by natural language processing models, used, for example, in machine translation. Interestingly, natural language models are currently one of the main driving forces in machine learning. Many modern models for processing images, videos, and other data formats were first proposed for text processing. A similar situation can also be seen in the past. As is known, the first striking application of Markov chains was the statistical analysis of a literary text [1], but then this model turned out to be in demand in a much wider range of applications. A modern example is the architecture of transformer-type neural networks (Transformer) [2]. The model was proposed in the context of machine translation and formed the basis of language models that have become the most popular and advanced: BERT (Bidirectional Encoder Representations from Transformers) [3], GPT (Generative Pretrained Transformer) [4], and a number of others. Recently, adaptations have been proposed for working with images (VisualTransformer) [5] and video (VideoGPT) [6], as well as other tasks (see review [7]). The above language models are closely related to another important concept that has been shaped mainly in the context of natural language processing models: the concept of foundation models of machine learning. The essence is that, instead of training individual models for each specific task, a single and, in a sense, universal model is trained for many tasks at once. Today, this approach is actively being developed in a variety of applications (see review [8]).

Without a doubt, the results that machine learning methods show in mundane tasks inspire the quest for applications in more specialized fields. The purpose of this paper is to review applications for problems related to solar activity studies. The emphasis is on demonstrating the diversity of problems for which machine learning models can be useful and on providing examples of solutions to those problems. A detailed discussion of differences among different solutions to the same problem and a comparison of the quality of approaches is beyond the scope of this paper, because it would require a more in-depth analysis of subtleties of the nature of data, the structure of the models, and the quality assessment criteria.

E.A. Illarionov
Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Leninskie gory 1, 119991 Moscow, Russian Federation
E-mail: egor.illarionov@math.msu.ru

Received 5 August 2024
Uspekhi Fizicheskikh Nauk **195** (4) 395 – 415 (2025)
Translated by S. Alekseev

The idea to use machine learning methods in solar physics is not unreasonable. One of the most important premises suggesting that the resulting model would be viable is the availability of a large array of data containing examples of the desired solution to the problem. In this sense, solar physics is a rich source of both diverse problems and a wealth of observational data. The appearance of the first systematic data on solar activity is usually associated with a series of records and sketches of the solar disk and positions of sunspots made by Galileo Galilei in 1612, although several earlier series of sketches by Thomas Harriot in late 1610 to early 1613 have recently been reconstructed [9]. In any case, the historical corpus of data is currently more than 400 years old, which is a unique duration for a scientific measurement streak in general. Of course, historical data are not without their own uncertainties and irregularities, which complicates their analysis, but they help us to understand the long-term features of the dynamics of solar activity. Modern ground-based and space telescopes together provide virtually continuous monitoring of the solar disk with high spatial resolution (up to several tens of kilometers for the DKIST telescope [10]) and in different spectral ranges (white light, ultraviolet (UV) and X-ray ranges, the radio range, and magnetograms). All this allows the formation and development of active regions on the Sun to be comprehensively studied.

The large time span and high level of space–time details achieved to date invite various statistical studies. In practice, however, the analysis of observational data is significantly complicated by a number of factors. One of them is the heterogeneity of the data. Over the 400 years, observers have changed, technology has improved, and ideas about what and how to observe have evolved. As a result, the question of reconciling earlier and later observations arises, entailing various discussions on this matter. For example, the Wolf number series, compiled from individual series of observations of sunspot groups, has been revised more than once, resulting in significant modifications (see, e.g., [11]).

Another major problem is the availability and completeness of data. A hefty stratum of historical data is still in paper archives, and much of what has been scanned into electronic form awaits digitization of handwritten text and sketches. Many details of how observations were conducted in the distant past are unknown to us, and hence the digitization stage is often also a stage of reconstructing the observation methods.

Modern observational data, on the one hand, are free of many problems that arise when working with historical data: they are immediately presented in digital form, are much more homogeneous (the known problem of degradation of measuring instruments notwithstanding), and are taken regularly. Still, we cannot say that they have been studied systematically. The main stumbling block is the large physical volume of data, replenished daily with streams of new data. As an example, we can recall that just one SDO (Solar Dynamics Observatory [12]) satellite, which is operational since 2010, observes the solar disk every 12 seconds.

We can conclude that extracting useful information from large amounts of data is a challenge for modern science and computing technologies. Machine learning methods, as practice shows, constitute one effective way to achieve this goal.

In what follows, we recall the main ideas of machine learning methods and then proceed to examples of their use in solar physics.

2. Main ideas of machine learning

2.1 Data model

A typical example of the emergence of a machine learning task is the problem of describing some observed dependence in functional form (building a data model). If it is also desirable to work out a general approach to solving a whole class of similar problems, then the nature and details of the observed dependence at a particular moment must be unimportant. Instead, a parametric family of functions (often also called a model) must be chosen so as to be broad enough to cover any possible dependence. An example of such a family is given by polynomials, which, as is known, allow approximating continuous functions on an interval with any accuracy.

While mathematically perfect, many constructions become computationally complex when working in high-dimensional spaces (for example, the resulting products of a large number of polynomials can quickly overcome the limits set by machine precision). We note that high-dimensional spaces are a typical, rather than a special, case in modern data processing tasks. An example is given by the problem of classifying images, which are essentially numerical matrices and are interpreted as objects in a space with a dimension equal to the number of matrix entries. For high-resolution images, this number can be several million. Thus, from a computational standpoint, the question arises of finding a more convenient and universal method for approximating functions.

An important concept of functions in arbitrary-dimension spaces was worked out by Kolmogorov [13]. He showed that any continuous function in the interior of a unit n-dimensional cube can be represented by a finite composition of functions of one argument,

$$f(x_1, x_2, \ldots, x_n) = \sum_{k=1}^{2n+1} \varphi_k \left(\sum_{i=1}^{n} \psi_{i,k}(x_i) \right), \tag{1}$$

where the functions $\psi_{i,k}$ are independent of f and depend only on the dimension n. We can say that they form a universal basis. The functions φ_k, of course, do depend on f.

In essence, functions of n arguments can be represented as a relatively simple computational graph, as shown in Fig. 1, where the number of nodes depends on the space dimension only linearly. Thus, relying on this theorem, we can hope to

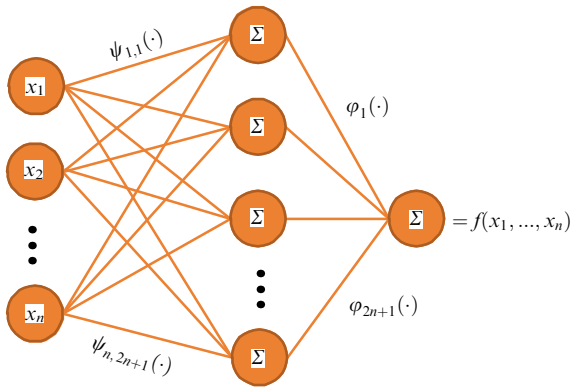

Figure 1. Representation of a function by a superposition of functions of one variable, in accordance with Kolmogorov's theorem [13].

find an approximation of a function using a finite (and quite reasonable) number of simple functions (simple in the sense that they depend on one argument). A practical implementation of this theorem was proposed quite recently in [14], but it is still not without its shortcomings, and it is therefore too early to speak about its widespread use.

From a practical standpoint, a somewhat later result in Cybenko's theorem [15] was of great importance (it is worth noting that, in the same year, the same result was independently obtained in [16, 17]). Cybenko's theorem states that functions that are continuous inside a unit n-dimensional cube can be approximated with any predefined accuracy using a composition of sigmoid functions,

$$f(x_1, x_2, \ldots, x_n) \approx \sum_k \alpha_k \sigma\left(\sum_{i=1}^{n} \gamma_{i,k} x_i + \beta_k\right), \tag{2}$$

where $\gamma_{i,k}$, α_k, and β_k are constants depending on f, and σ is some predefined continuous nondecreasing bounded function on the entire line. Such a function is often taken in the form

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \tag{3}$$

known as the logistic function or sigmoid. The corresponding computational graph is shown in Fig. 2. Superficially, it is very similar to the one in Fig. 1, but an essential difference is that the number of nodes in the second case is not specified in advance and can be large enough for an accurate approximation, but unknown functions are replaced with unknown scalar parameters, which are much easier to work with. Today, this computational graph is usually called the fully connected neural network model with one hidden layer and the activation function σ. Accordingly, Cybenko's theorem is referred to as the theorem of universal approximation by neural networks. Graph nodes are called neurons, and the specific structure of the graph (the number of neurons, layers, and connections) is called the neural network architecture. Somewhat anticipating the subsequent discussion, we note that modern architectures have a large number of layers and a sophisticated system of connections. We refer the reader to [18, 19] for further discussion of theoretical issues.

In this review, we limit ourselves to considering applications of machine learning models based on neural networks, although they by no means exhaust the toolbox of machine learning methods. A systematic review can be found, for example, in books [20–22]. We only note that the general principles remain the same.

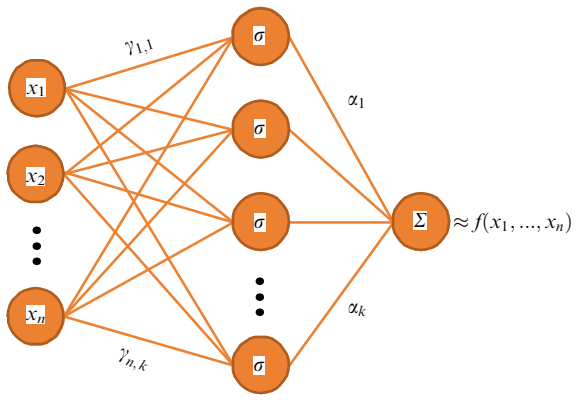

Fig. 2. Approximation of a function by a superposition of sigmoid functions, in accordance with Cybenko's theorem [15].

2.2 Training problem and loss function

A data model based on a neural network defines a whole family of functions. The choice of a particular function is determined by the choice of specific model parameter values. To select such values, we must formulate the criteria that the desired function must satisfy. Mathematically, this is implemented by defining a functional (often called the loss function or the risk function) whose minimum is achieved on the desired function, and the optimization problem is then solved. The functional in question can include observational data, physical relations (for example, differential equations), various types of constraints on the function values, and other terms. We consider some of the most important examples.

2.2.1 Classification and regression. A typical problem is as follows: given the data in the form of pairs of observations (x_i, y_i), where the subscript i is the observation number, find a function $f(x)$ that approximates the observed values of the second variable y. Such a formulation is possible for the binary classification problem (where y takes only two values, for example, 0 and 1), a multiclass classification problem (with y having a discrete nature and more than two possible values), or regression (where y is continuous). We let $f(x, \theta)$ denote the data model, with θ being the set of model parameters. We can then propose the loss function as the average over all observations of

- the squared difference $l(\theta) = (y_i - f(x_i, \theta))^2$ for the regression problem;
- the binary cross-entropy $l(\theta) = -y_i \log f(x_i, \theta) - (1 - y_i) \log(1 - f(x_i, \theta))$ for the binary classification problem (where the values of $f(x_i, \theta)$ are assumed to be nonnegative and not exceeding 1);
- the cross-entropy $l(\theta) = -\sum_{k=1}^{K} y_{i,k} \log f_k(x_i, \theta)$ for the K-class classification problem, where $y_{i,k} = 1$ if $y_i = k$ and 0 if otherwise (and the $f_k(x_i, \theta)$ are also assumed to be nonnegative numbers such that their sum over k is 1).

The above functions are by no means the only possible ones. They can be called canonical in the sense that they arise from maximum likelihood estimates. For example, the mean-square loss function arises under the assumption that the observations y_i are independent and distributed in accordance with the normal (Gaussian) law, in which the mean (mathematical expectation) is a function of x_i and the parameters θ. It is also useful to recall that all three cases (regression, binary, and multiclass classification), despite their superficial differences, can be combined into one model and studied in the framework of a single theory of a generalized linear model using the tools of probability theory and mathematical statistics ('linearity' in no way limits the consideration of nonlinear dependences). For more information on the probabilistic context of machine learning models, we refer the reader to books [20, 22].

It is worth noting that, in a number of problems formally related to classification or regression, the choice of the functions considered above may not fully correspond to the nature of the problem. As an example, we consider the situation where y_i are images (say, any photograph). If we mentally shift the entire image one pixel to the right, then both images would be virtually indistinguishable from the standpoint of perception, but the mean square distance between

them can be large (for example, larger than between two images that are visually dissimilar). In such cases, more specific metrics have to be used. For example, in the context of images, they can be metrics calculated not directly between pixel values but between some more complex characteristics of two images (for example, perceptual loss [23]).

2.2.2 Physically informed neural networks. Classification and regression problems are basic in the sense that other problems, whose original formulation can differ greatly from the problem of predicting y from x values, can be reduced to them in one way or another. As an example, we consider the problem of numerically solving differential equations. For definiteness, we take an ordinary differential equation $F(x, y, y', \ldots, y^{(n)}) = 0$ for a function $y(x)$ with the initial condition $y(x_0) = y_0$. This problem can be solved using the finite-difference method and its variations [24]. However, it can also be approached from the machine learning standpoint. Because neural networks are a universal approximator, we can define the desired function $y(x)$ in the form of a neural network and choose its parameters such that it satisfies the chosen differential equation as accurately as possible. It is useful to note here that a function represented by a neural network is by construction differentiable with respect to both its parameters and the input variables. Moreover, all derivatives are calculated using automatic differentiation methods (see below). Thus, if we define $y(x)$ by a neural network $y(x, \theta)$, then all derivatives $y^{(n)}$ are also defined. Next, to compile the loss function, one usually takes a set of random points x_i from the domain of $y(x)$ and calculates the mean square of the discrepancy between the left- and right-hand sides of the equation at these points: $l(\theta) = F^2(x_i, y(x_i, \theta), y'(x_i, \theta), \ldots, y^{(n)}(x_i, \theta))$ (recall that there is zero on the right-hand side of the equation, hence the square of F). To take the initial condition into account, a term of the form $(y(x_0, \theta) - y_0)^2$ is added to the error function with a coefficient that controls the contribution of this term.

It is easy to see that solving a differential equation is thus reduced to a regression problem. However, a more interesting conclusion is obtained if we try to combine the problems of regression on observed pairs (x_i, y_i) and of solving a differential equation. In other words, we assume that the observed dependence of y on x satisfies a given differential equation with some accuracy. Accordingly, the dependence $y(x)$ can be found by minimizing the composite loss function

$$l(\theta) = \frac{1}{n}\sum_{i=1}^{n}\big(y(x_i) - y_i\big)^2 + \frac{\alpha}{m}\sum_{j=1}^{m} F^2\big(x_j, y(x_j), y'(x_j), \ldots, y^{(n)}(x_j)\big)\,. \tag{4}$$

Here, the θ argument is omitted for simplicity, n is the sample size of observational data, m is the number of points at which the accuracy of the solution of the differential equation is estimated, and the α coefficient is responsible for the balance of the two terms. Such a construction is known as a physics-informed neural network (PINN) [25]. We note that the a priori physical model of the data in the form of a differential equation acts as an extra constraint (often called a regularizer) on the spectrum of possible dependences $y(x)$ underlying the observed data. In particular, this approach can help overcome the model overfitting problem (when the training data error is significantly less than the test data error) and increase the stability of prediction for new data.

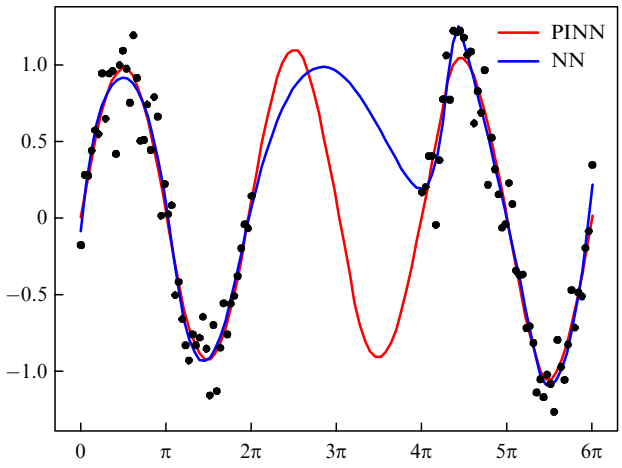

Figure 3. Regression on training set (black dots) using PINN (red line) and regular neural network (NN, blue line) models. Model architecture is the same in both cases. PINN is trained using equation $y'(x) = \cos x$.

As an example, we consider a problem where the training sample is obtained from a sinusoid superimposed by random Gaussian noise (Fig. 3). We take two identical fully connected neural network models and train the first one with the loss function given by the sum of the mean-square error in the training sample and the error from solving the differential equation $y'(x) = \cos x$; the second model is trained based only on the mean-square error in the training sample. Figure 3 shows that the second model fits the observational data better, while the first model balances between the physical model and the observational data. The difference is pronounced in the region where no observations are available.

Other methods of model regularization also exist. For example, additional terms characterizing the amplitude of the model parameters can be added to the loss function: the sum of the squares of the model parameters (the so-called L_2, or ridge, or Tikhonov regularization) or the sum of their moduli (the L_1, or LASSO regularization). More complex constructions are described in [26].

2.2.3 Neural-differential equations. From the problem of solving differential equations, we proceed to the inverse problem: given observations of the process at certain time instants, reconstruct the differential equation that models this process. The mathematical formulation of the problem again reduces to finding the loss function. We assume that the observed process satisfies the equation

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = g\big(f(t), t\big) \tag{5}$$

with an unknown function $g(\cdot, \cdot)$, which we define by a neural network $g(\cdot, \cdot, \theta)$. We then solve the equation using standard numerical schemes and vary the neural network parameters such that the solution result agrees with the observed data. To construct the loss function, we let $\mathrm{ODESolve}\,(g, t_0, f_0, t)$ denote the numerical scheme whose input is given by the function on the right-hand side of (5), the initial data (t_0, f_0), and the time instant t for which the solution is sought. Let $\{(t_i, f_i)\}_{i=1}^{n}$ be a set of observed values of the process. Then, the mean-square error can again be used as the loss

function:

$$l(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathrm{ODESolve}\,(g, t_0, f_0, t_i) - f_i\right)^2 . \tag{6}$$

This approach is known as neural-differential equations [27]. We note that the key is the procedure for calculating the gradient of the loss function $l(\theta)$ with respect to the parameters θ without having to directly differentiate the ODESolve function. For this, an auxiliary (adjoint [28]) differential equation is constructed, whose solution yields the value of $\mathrm{d}l/\mathrm{d}\theta$.

2.2.4 Autoencoders. Many types of observational data, for example, images and spectral line profiles, are objects in high-dimensional spaces (images are specified by numerical matrices, and lines, by a set of values at each wavelength). To analyze such objects, their description in a more compact form is desirable, using a smaller number of parameters but capturing the most significant features. Mathematically, such a problem is akin to that of reducing the dimensionality of data. To make this problem meaningful, one can require that the map of an object to a lower-dimension space be almost invertible, i.e., that the reconstruction of the original image be possible. Then, one can assert that the lower-dimensional representation preserves information about the original object. A common standard method for solving this problem is the principal component analysis (PCA) method proposed by Pearson in 1901 [29]. However, this method is linear, and we can therefore assume its efficiency to be insufficient with complex data (a discussion of the features of the PCA method can be found in [30]). Finding nonlinear methods then becomes of interest, the autoencoder model being a possible solution. The model consists of two functions defined by neural networks. One of them, denoted by $\mathrm{Enc}\,(x, \theta)$, is called the encoder and maps x to a lower-dimensional vector $z = \mathrm{Enc}\,(x)$. The vector z is called the hidden or latent representation of x. The second function, the decoder $\mathrm{Dec}\,(z, \tilde{\theta})$, maps vectors from the latent space to the original one: $\tilde{x} = \mathrm{Dec}\,(z)$. In the simplest implementation, model training requires that the composition of the decoder and encoder $\mathrm{Dec}(\mathrm{Enc}\,(x))$ be as close to the identity transformation as possible. Accordingly, the loss function arises in the form

$$l(\theta, \tilde{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathrm{Dec}\big(\mathrm{Enc}\,(x_i)\big) - x_i\right)^2 , \tag{7}$$

where $\{x_i\}_{i=1}^{n}$ is the training dataset. There are various modifications of the autoencoder model, which, for example, impose additional constraints on the distribution of vectors in the latent space. As an example, let us recall the variational autoencoder proposed in [31]. It involves an additional term in the loss function, which is responsible for the closeness of the distribution of latent vectors to the standard multivariate Gaussian distribution. An extended review can be found in [32].

In the context of variational autoencoder-type models, the models called normalizing flows are also worth mentioning [33]. Their main purpose is to select a rigorously invertible transformation that converts the distribution generated by the sample into a standard Gaussian one. Accordingly, the dimensions of the hidden and original spaces are set equal, and the neural network is constructed such that the inverse transformation can easily be written. For a more detailed review, we refer the reader to [34]. One of the applications of normalizing flows is the generation of synthetic data. For this, a sample from a Gaussian distribution is modeled, which is then mapped into the original space by the inverse of the normalizing flow, thereby creating a new synthetic sample. The same technique allows modeling synthetic data using the decoder of a pretrained variational autoencoder.

2.2.5 Generative models. Let us discuss generative models in more detail. Their purpose is to create new data samples that are indistinguishable in properties from a real data sample, but are not literally the same. Mathematically, this means that the sample obtained from the model has the same distribution as the actual data sample. The resulting synthetic samples can be useful, for example, as initial data in simulations of dynamic systems.

A generative model (generative-adversarial network, GAN) usually consists of two parts implemented in terms of neural networks. One part is called the generator $G(z, \theta)$: it maps the space in which the standard Gaussian sample z_i is modeled to the space in which the real sample x_i is defined. The second model is called the discriminator $D(x, \tilde{\theta})$: it is a binary classifier that must distinguish a synthetic example from the real one. The generator and the discriminator have competing tasks. The generator must learn to produce examples that are classified as real from the discriminator's standpoint, while the discriminator, on the contrary, must learn to discriminate between real and synthetic data as well as possible. This game gives rise to a loss function for the generator

$$l(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\log\left[1 - D\big(G(z_i)\big)\right] , \tag{8}$$

which becomes smaller (noting the minus sign before the expression) the more often the discriminator erroneously classifies synthetic data as real (it is assumed that real data correspond to class zero, and synthetic data, to class one). The loss function for the discriminator is

$$l(\tilde{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\log D(x_i) + \log\left[1 - D\big(G(z_i)\big)\right] . \tag{9}$$

The first term becomes smaller the more accurately the discriminator classifies real data as real, and the second term becomes smaller the more accurately it classifies synthetic examples as synthetic. This loss function was proposed in [35]; modern implementations use more complex constructions (see, e.g., [36]).

The above examples of problems and corresponding loss functions by no means exhaust the range of possibilities, but they often serve as a starting point for discussions of more complex constructions (as an example, we note the problem of approximating operators in infinite-dimensional spaces by neural networks [37]). Another motivation for discussing these examples was to demonstrate the diversity of problem statements for machine learning methods. We now proceed to the next stage: numerical solution of the loss function minimization problem.

2.3 Minimizing loss function

On the one hand, the problem of finding a minimum point of a function is classical in numerical methods, but a number of

features that arise in the context of machine learning give rise to new challenges for methods as well as computing devices. The first feature is associated with the high dimension of the space in which the optimization problem is defined. Modern neural networks have about 10^6–10^8 parameters and are trained on samples of comparable size. Accordingly, the question arises about efficient parallelization of computations and the use of processors that support such technologies. In particular, graphic processing units (GPUs) and tensor processors (TPUs) [38], designed especially for machine learning tasks, are used.

Another important feature of the problem is that the loss function is almost always nonconvex and has many local minima. Accordingly, the classical gradient descent method is ineffective (it directs to the nearest local minimum, which can be much worse than the global one) and, moreover, the result is strongly dependent on the choice of the starting point (in the context of neural networks, we are talking about the initialization of the model parameters, which are usually set randomly). Machine learning problems have stimulated a new round of development of numerical optimization methods, as a result of which a number of new algorithms have appeared that use different ideas on how to ensure convergence to a more optimal local minimum. A review of these methods is beyond the scope of this paper, and we only mention some algorithms that have appeared over the past decade and have entered the standard set of libraries for working with neural networks:

- adaptive learning rate method (Adadelta) [39];
- adaptive moment estimation (Adam) [40];
- adaptive subgradient method (Adagrad) [41];
- evolved sign momentum (Lion) [42];
- follow the regularized leader (FTRL) [43].

More classical algorithms, also included in standard libraries, are

- stochastic gradient descent (SGD) [44];
- averaged stochastic gradient descent (ASGD) [45];
- limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [46];
- resilient backpropagation (RPROP) [47].

As an example, Fig. 4 shows the trajectories traced in searching for a minimum by some algorithms for the same function and starting point.

The above optimization methods use gradient descent in one form or another. This raises the question of how to calculate the derivatives (gradient) of the loss function with respect to the model parameters. Standard approaches — numerical or symbolic (computer algebra) differentiation — are not entirely suitable for such purposes for a number of reasons. Symbolic differentiation is slow due to implementation features and is ineffective for deep networks with complex architecture. Numerical differentiation, due to its discrete nature, introduces errors that lead to instability and is ineffective for a large number of model parameters. These problems were solved with a method called automatic differentiation [48] (see also contemporary review [49]). We discuss one of the implementations of automatic differentiation, which is called the backpropagation algorithm (see, e.g., [50, 51] and contemporary review [52]) and is currently the main method for training neural networks.

The computational graph of a neural network is built from nodes that represent simple arithmetic operations: addition, multiplication, raising to a power, exponential, logarithm, etc. In each node, besides the operation itself, a

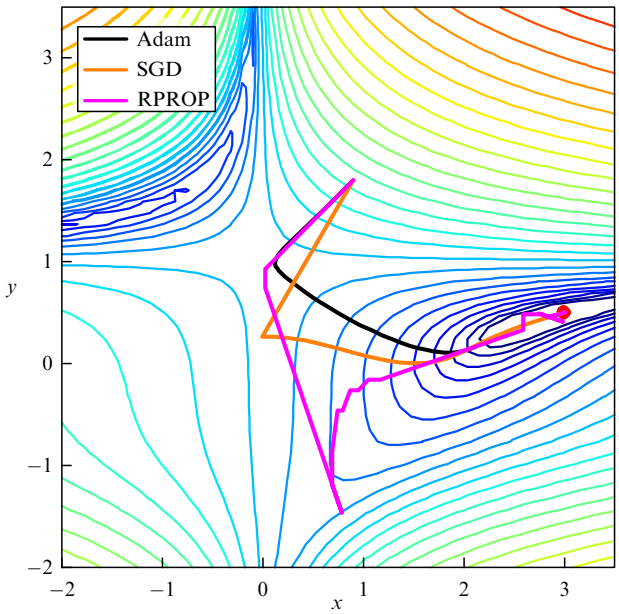

Figure 4. Trajectories converging to a minimum obtained by different optimization algorithms: black line is Adam, orange line is SGD, and purple line is RPROP. Red dot shows location of global minimum of the function.

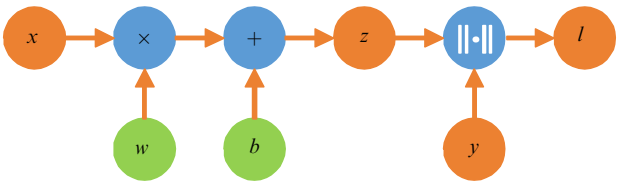

Figure 5. Computational graph illustrating error backpropagation algorithm. Green: trained model parameters, blue: arithmetic operations (multiplication, addition, calculating the norm), orange: other numerical variables.

function is written that implements the rule for calculating the derivative of this operation. With the example of the simplest computational graph shown in Fig. 5 and implementing a neural network with one neuron, we explain the procedure for calculating the gradient based on model parameters w and b.

The procedure consists of two stages. In the first stage, called the forward pass, the values are calculated at all nodes of the graph, from the input to the output, and are stored in memory. The second stage, called the backward pass, consists of applying the chain rule of differentiation of a composite function to sequentially find derivatives when going from the output node to the nodes containing the model parameters:

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial z}\frac{\partial z}{\partial b}, \tag{10}$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial z}\frac{\partial z}{\partial w}. \tag{11}$$

The rules for calculating all derivatives on the right-hand side of (10) and (11) are written in the graph nodes: $\partial l/\partial z = 2(z-y)$, $\partial z/\partial b = 1$, and $\partial z/\partial w = x$. It then remains to substitute the values of x, y, and z saved after the forward pass. We note that the use of dynamical programming principles allows avoiding the duplicating of calculations in

(10) and (11) and provides an efficient procedure for more complex neural network architectures.

Next, the iterative training procedure is launched. At each iteration, a subsample (called a batch in machine learning) is taken from the training data set, the loss function gradient is calculated on this subsample based on the model parameters, and the model parameters are updated in accordance with the optimization algorithm. There are different criteria for stopping the iterations: for example, when the loss function value stops decreasing significantly on the training sample or the error on the validation sample starts increasing (for the mathematical argument, see, e.g., [53]). The last-iteration model parameters are fixed, and a trained model is said to have been obtained.

In the next section, we consider examples of models and their training results in solar physics problems.

3. Data sets for machine learning

The basis of machine learning is the training data set. The quality of a model depends on the properties of the set:

- size (the number of objects);
- variability (the diversity of examples);
- unbiasedness (the statistical properties of the training set must match the properties of the test set);
- annotation accuracy (well-defined classes, object boundaries, etc.).

Yet another important factor is the homogeneity of the data format. For example, if the data are represented by images of different sizes (in pixels), then time-efficient training of the model would most likely require a reduction of all images to the same size. For example, efficient training of neural networks is based on matrix operations, and hence the set of images should be represented by a single multidimensional array.

In practice, collecting and preparing a training dataset is one of the most labor-intensive and time-consuming stages in machine learning, and at the same time one of the most resource-intensive stages, requiring large data warehouses and a high speed of access to data (reading and writing). This can only be fully implemented in large computing centers. As a result, large volumes of data prepared for machine learning are of exceptional value today. In Table 1, we collect examples of solar activity data sources and datasets

Table 1. Example of solar observation catalogues and databases prepared for machine learning tasks.

Title	Brief description	Address	Literature
HEK	Consolidated database from various sources on solar activity events and objects	https://lmsal.com/hek/	[54]
Helioviewer	Visualization of solar observation catalogues from various instruments and solar activity maps	https://www.helioviewer.org/	[55]
Interactive Multi-Instrument Database of Solar Flares	Consolidated catalogue of solar flares based on GOES, RHESSI, SDO/AIA, and a number of other instruments for 2002–2022	https://solarflare.njit.edu/	[56]
Catalogue of SDO/AIA 193 Angstrom synoptic maps and coronal holes	Catalogue of synoptic maps and coronal hole maps based on SDO/AIA observations on the 193-Å line from 2010 to the present	https://sun.njit.edu/coronal_holes	[57]
Large-Scale Dataset of Three-Dimensional Solar Magnetic Fields Extrapolated by Nonlinear Force-Free Method	Set of 73,000 examples of coronal magnetic field reconstructions over active regions from the SHARP catalogue for 2010–2019	https://nlfff.dataset.deepsolar.space/en	[58]
Large-scale Solar Dynamics Observatory image dataset for computer vision applications	Set of 260,000 SDO/AIA images corresponding to 270,000 events from the Heliophysics Event Knowledge (HEK) base	https://dataverse.harvard.edu/dataverse/lsdo	[59]
Machine Learning Data Set for NASA's Solar Dynamics Observatory	SDO/AIA and SDO/HMI images from 2010–2018, calibrated and downsampled to a 512×512 resolution	2010, https://purl.stanford.edu/vk217bh4910 2011, https://purl.stanford.edu/jc488jb7715 2012, https://purl.stanford.edu/dc156hp0190 2013, https://purl.stanford.edu/km388vz4371 2014, https://purl.stanford.edu/sr325xz9271 2015, https://purl.stanford.edu/qw012qy2533 2016, https://purl.stanford.edu/vf806tr8954 2017, https://purl.stanford.edu/kp222tm1554 2018, https://purl.stanford.edu/nk828sc2920	[60]
ObserveTheSun	Catalogue of solar activity maps compiled by the Kislovodsk Mountain Astronomical Station	https://observethesun.ru	[61]
SDOBenchmark	Set of 8336 training and 886 test data on solar flares based on SDO/AIA and SDO/HMI images 12 h, 5 h, 1.5 h, and 10 min before the flare	https://i4ds.github.io/SDOBenchmark/	[62]
SEP[3]	Consolidated catalogue of SPEs, CMEs, and solar flares and their parameters based on GOES, SOHO/LASCO, SDO/HMI, and a number of other instruments	https://sun.njit.edu/SEP3	[63]

Table 1 (continued)

Title	Brief description	Address	Literature
SERPENTINE	Catalogue of solar cosmic ray data based on Solar Orbiter, Parker Solar Probe, and BepiColombo observations	https://serpentine-h2020.eu/	[64]
SHARP	Vector magnetograms of active regions and their parameters based on SDO/HMI observations in 2010–2020	http://jsoc.stanford.edu/doc/data/hmi/sharp/sharp.htm	[65]
SMARP	Magnetograms of active regions based on SOHO/MDI observations in 1996–2010	http://jsoc.stanford.edu/	[66]
SOHO/EIT Flare Catalog	Catalogue of solar flares and their parameters based on SOHO/EIT data in 1997–2010	https://doi.org/10.7910/DVN/C9H34R	[67]
Solar active region magnetogram image dataset for space weather studies	Catalogue of magnetograms of active regions, reduced to a single size of 600×600 pixels (and 224×224 in a reduced version) for 2010–2018	Full resolution version, https://doi.org/10.5061/dryad.dv41ns23n Low resolution version, https://doi.org/10.5061/dryad.jq2bvq898	[68]
SolarMonitor	Visualization of solar observation catalogues from different instruments and solar activity maps	https://www.solarmonitor.org/	[69]
SunInTime	Visualization of solar disk images from the SDO satellite and the catalogue of active HEK events	https://suntoday.lmsal.com/	[54]
Sunspot groups	Catalogue of sunspot group images and their parameters based on data from the Kislovodsk Mountain Astronomical Station for 2010–2022	https://github.com/observethesun/sunspot_groups	[70]
SWAN-SF	Catalogue of solar flares and associated parameters obtained simultaneously from different instruments in 2010–2018	https://doi.org/10.7910/DVN/EBCFKM	[71]

prepared for machine learning tasks. We note that such databases are also relevant for more general problems of studying solar activity.

4. Software for working with data

The use of machine learning methods involves the use of tools for both creating and launching models, for data loading and preprocessing, and for implementing the training procedure. Tools for working with machine learning models are universal and independent of the nature of the data. For example, PyTorch, Keras, and Tensorflow libraries are popular for working with neural networks. Data processing tools, on the contrary, depend significantly on the data storage format and their nature. In most solar activity studies, these steps are quite standard and are repeated from one task to another, and a reasonable strategy is therefore not to create a new program code for each new task but to use unified tools. The idea arises of creating libraries focused on solving standard data processing problems in a certain area of research.

In solar physics, one of the most famous is the SolarSoft library [72], which has existed for more than 30 years. Written in the IDL language, it continues to be developed and is currently supported. Most of its functionality is currently implemented in the SunPy library (Table 2) written in Python, which is the main language for machine learning. At the same time, a number of other libraries are emerging that solve the problems of creating homogeneous datasets from various sources of observational data, preprocessing and analyzing them (see examples in Table 2).

We note that the modern practice is to make the source code publicly available from repositories. This allows independently verifying and reproducing the stages of

Table 2. Examples of software libraries for processing and analyzing solar observations.

Name	Brief description	Address
aiapy	Library for calibration and processing of SDO/AIA data [73]	https://aiapy.readthedocs.io/
FlareNet	Code for preparing data and training a solar flare forecast model [74]	https://github.com/nasa-fdl/flarenet
Mission ML Data Ready	Library for creating datasets from SOHO and SDO catalogues [75]	https://github.com/cshneider/soho-ml-data-ready
observethesun	Collection of machine learning programs and models in solar physics	https://github.com/observethesun
SERPENTINE	Tools for loading and analyzing data on solar cosmic rays [64]	https://serpentine-h2020.eu/
SunPy	Library for loading, processing, and analyzing observational data on solar activity [76]	https://sunpy.org/
SpaceML	Collection of programs and models on machine learning in solar physics	https://spaceml.org/

scientific research and using it as a starting point for further work.

5. Machine learning models in solar physics

5.1 Segmentation of solar disk images

One of the main types of observational data on the Sun and solar activity is solar disk images. These images can be obtained in white light as well as at specific wavelengths. Observers start processing such images by identifying active regions of the Sun. Sunspots are a well-known example, but they do not exhaust solar activity. In particular, there are areas known as coronal holes. They appear as dark areas in X-ray images of the solar disk (the 193- or 195-Å line is usually chosen). The problem is that not only coronal holes but also some other objects, including solar filaments, appear dark in such images. Thus, the problem of identifying (outlining) coronal holes is nontrivial and ambiguous.

On the other hand, the results of many years of processing, accumulated in catalogues of coronal hole maps, allow creating a machine learning model tasked with reproducing the work of expert observers. A recent proposal in [77] was to use a U-Net-type convolutional neural network model [78] for coronal hole segmentation.

The U-Net model consists of a sequence of convolution operations applied to the input image of the solar disk. Perhaps the best-known example of convolution is with a Gaussian kernel, which smoothens the image. The Gaussian kernel contains predetermined values, while kernel values in the convolutional neural network model are trainable parameters. We discuss the advantages of this approach to working with images by comparing it with fully connected models (such as the one shown in Fig. 2).

First, the input to a fully connected model is a vector of values but not a matrix of values. Of course, a matrix can be represented as a vector, but this erases the two-dimensional structure and the concept of locality (nearby pixels in the image being connected to each other). A two-dimensional convolution appears to be a more natural model in this case.

Another issue is related to the number of model parameters. Let us assume that the input matrix (image) is 1000×1000 (a typical resolution of existing solar disk images), and the next layer of the fully connected network has at least 100 neurons. Then, the model already has about 10^8 parameters (the number of parameters is of the order of the number of neurons in the preceding layer times the number of neurons in the next layer). In a convolutional network, a single convolution kernel (which is typically 3×3, 5×5, or 7×7, and rarely bigger) has the number of parameters of the order of 10, and a smaller total number of parameters can even be obtained when using multiple kernels.

In addition, the convolutional model, unlike the fully connected one, is invariant under shifts and to some extent insensitive to the image size. Indeed, the convolution operation is defined on images of any size, and a shift of objects in the input image leads only to a shift in the output image. In a fully connected model, the number of inputs is fixed, and the result of shifting objects in the input image gives a poorly predictable result.

Figure 6 shows some of the images obtained by convolutions with a set of kernels in the internal layers of a trained U-Net model. We note that the model consists of two parts, implementing the principle of an encoder and a decoder. When passing through the encoder branch, the spatial dimensions of the image decrease, but the number of channels increases. It is assumed that, on this path, the features significant for solving the target problem are extracted from the image. On the decoder branch, the features are unfolded into a new image, which in the case of a segmentation problem is a binary matrix with ones corresponding to pixels belonging to the desired objects (for example, coronal holes).

The model in [77] was trained on a long-term record of coronal hole maps of the Kislovodsk Mountain Astronomical Station of the Main (Pulkovo) Astronomical Observatory

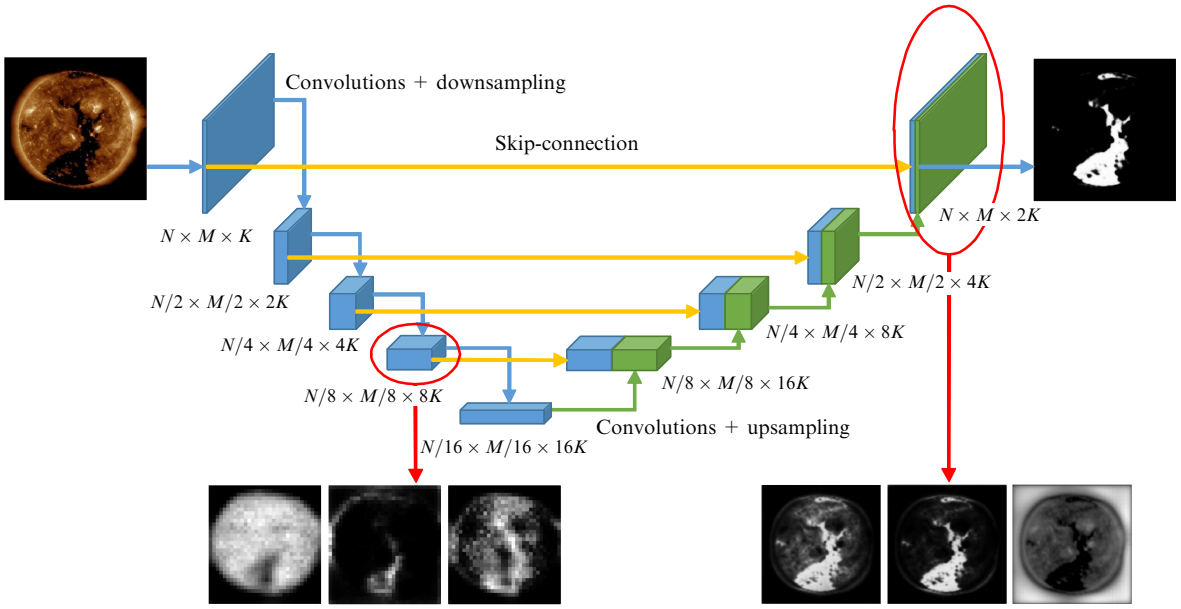

Figure 6. Architecture of a U-Net convolutional neural network. Blue arrows: convolution and downsampling operations. Green arrows: transposed convolution (convolution and upsampling). Orange arrows: array concatenation operations. Block captions indicate array sizes (in terms of images: height, width, and number of channels). Images under diagram show individual slices of the arrays outlined with red ellipses.

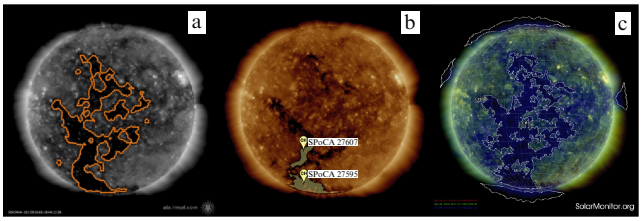

Figure 7. Comparison of coronal hole segmentation results using different algorithms: (a) U-Net model [77], (b) SPoCA algorithm [81], and (c) HIMERA algorithm [82]. (Figure from [77].)

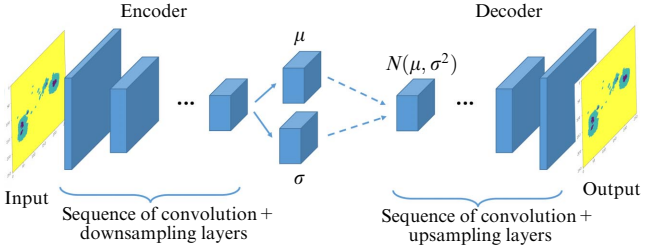

Figure 8. Variational autoencoder architecture. Input image is passed through a sequence of convolutional layers (encoder), resulting in two arrays that are interpreted as parameters of a normal distribution. Random vector from resulting distribution is passed through a second sequence of convolutional layers (decoder), resulting in reconstructed image. (Figure from [70].)

of the Russian Academy of Sciences. During training, a standard metric was used for the segmentation task in the form of binary cross-entropy (recall that the same metric is standardly used in classification problems; the segmentation problem is considered as a pixel-by-pixel classification problem).

Figure 7 shows an example of segmentation using a trained U-Net model in comparison with some other algorithms. The observed differences, sometimes significant, are the subject of ongoing discussions (see, e.g., [79, 80]), which will apparently start coming to definitive conclusions as experience with the use of the obtained data accumulates. In this regard, one of the fundamental machine learning problems can be noted once again: to master the methodology of expert data processing and reproduce it on new (or, conversely, historical) data. This will create a long-term homogeneous data series, which is relevant for many problems in solar physics research.

Interestingly, the feature of convolutional models mentioned above — independence from the input image size — was confirmed in [57], where a U-Net model trained on solar disk images showed a stable result in segmenting coronal holes on synoptic maps (recall that a synoptic map represents the entire surface of the Sun obtained by gluing together daily solar disk observations).

The idea of using a U-Net-type model for segmenting solar disk images was developed further: for segmenting filaments [83], active regions on vector magnetograms [84], regions in the solar corona [85], and granules on the photosphere [86].

The study of various architectures continues: a convolutional model used in [87] had the input given by a set of images in different spectral lines and a magnetogram, and solved the coronal hole segmentation problem; the authors of [88] considered the detection problem (drawing a bounding box around an object) for identification of coronal holes, sunspots, and prominences; in [89–91], a more complex segmentation problem was considered, where, in addition to classifying pixels depending on whether they belong to a desired class of objects, individual instances within the class also had to be distinguished (the so-called instance segmentation problem, in contrast to the simpler semantic segmentation problem mentioned above).

5.2 Parametric description of data

The next step after identifying active regions is usually to measure their characteristics. For example, for a group of sunspots, the area, the extent in latitude and longitude, and the angle of inclination to the equator are determined and a certain morphological class is assigned, following, for example, the Zurich classification system [92]. More complex topological characteristics are sometimes considered (see, e.g., [93]). A similar analysis is carried out for other types of regions. Mathematically, this is about assigning certain quantitative characteristics to objects. The question of which characteristics can be useful, for example, when studying solar activity or predicting space weather factors, is open; a more general question regarding new methods for quantitatively describing objects can be considered.

In [70], a neural network model of a variational autoencoder type was proposed to describe the structure of sunspot groups. The general architecture of the model, shown in Fig. 8, consists of two parts: an encoder and a decoder. It is worth noting that this architecture has much in common with the U-Net model architecture in Fig. 6: both models use the concept of latent space and are based on the idea of compression and restoration.

A set of sunspot group images from the catalogue of the Kislovodsk Mountain Astronomical Station (www.observethesun.ru) was used for training. The task of the model at the training stage was to construct encoder and decoder functions such that their composition (i.e., compression followed by restoration) would reproduce the original image. In addition, the PCA method was used to rank the parameters obtained at the output of the encoder. As a result, a model was constructed that transforms sunspot group images 256×256 pixels in size into 283 numerical parameters. Next, the obtained parameters were analyzed and thereby given a physical interpretation. For this, one or more parameters were varied with the others fixed, and the restored images were studied. An example of the obtained images is shown in Fig. 9, whence we can conclude that one of the parameters characterizes the general bipolar or unipolar structure of the group and the other is responsible for the tilt angle of the group. As an application based on the obtained parameters, a model for assessing the complexity and for automatic classification of sunspot groups was proposed.

The idea of studying and using features extracted by machine learning methods also appears in a number of other studies. For example, in [94], an autoencoder was used to describe spectral line profiles; it was also shown that the obtained parameters allow a physical interpretation: in particular, they characterize the spectral line's asymmetry and width. In [95], an autoencoder model was used to identify anomalous images and localize active regions in a series of solar disk images taken by the SDO satellite. In [96], also based on parameters extracted from solar disk images, a model was proposed for generating synthetic images and predicting the radio emission flux. It was shown recently in

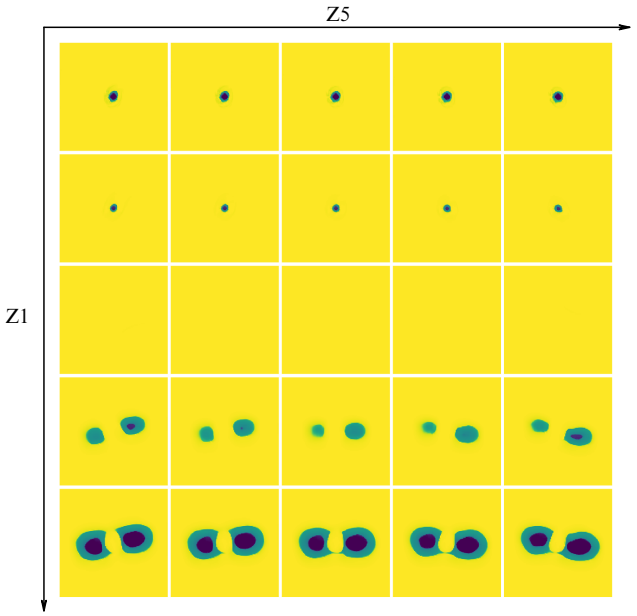

Figure 9. Reconstruction of images of sunspot groups by varying two parameters (denoted as Z1 and Z5) that encode the image. Interpretation follows from the figure: parameter Z1 encodes the unipolar or bipolar structure of the group, and Z5 defines its tilt. (Figure from [70].)

[97] that the parameters of magnetic active regions identified by an autoencoder can serve as precursors for solar flares and can be used to improve the accuracy and lead time of forecasts.

The above examples from solar physics, as well as a number of similar studies in other areas, show that machine learning methods open up new possibilities for describing and analyzing data. It is noteworthy that some of the parameters identified by the models coincide with those on which experts working with data primarily focus. And while an expert comes to a conclusion about the importance of some parameters based on many years of experience and knowledge of the subject area, a machine learning model does this only by collecting statistics. We also note that elements of this approach can often be found in classification, regression, and forecasting models, for example, as additional parameters on which the model is based. In the next section, we consider these constructions in more detail.

5.3 Solar activity forecasts

The task of forecasting solar activity parameters arguably accumulates all the main theoretical models and observational data. At the same time, many mechanisms remain not entirely clear and are in fact replaced by empirically (statistically) established connections. The use of machine learning methods is of interest because they allow revealing connections in a much wider data space and, as a result, obtaining more accurate models on the one hand and new ideas for theoretical models on the other.

The forecast tasks fall into two large groups: the forecast of 'rare' events and the forecast of indices. We discuss their properties.

5.3.1 Forecasting rare events. This problem includes forecasting events that can be called rare against the background of long-term solar activity: solar flares (the forecast of powerful flares of classes M and X being of practical interest, first and foremost), coronal mass ejections (CMEs), solar proton events (SPEs), etc. The standard technique is to discretize time (for example, by days) and reduce the problem to a binary classification one: whether an event occurs in a given time interval. Due to the rare nature of events, the number of examples when an event occurred is much less than when it did not, which makes the sample highly unbalanced (for example, for one region with an X-flare, there are 800 regions without flares [98]). When attempting to construct standard forecast models based on machine learning (or another statistical method), it turns out that the models tend to ignore rare events and collapse to a trivial solution: predicting the predominant class. This requires the use of techniques that increase the significance of error for rare events, for example, by introducing weight coefficients into the loss functions, artificially balancing the sample by selecting an equal number of positive and negative examples, or simulating new positive examples.

Another feature resulting from the discretization of time in forecast models is that it is not entirely clear how the accuracy of the model can be assessed. For example, according to a forecast, an event should occur within 24 hours, but it actually occurs after 24 hours and 1 minute. During standard training and testing of the model, such a case is classified as erroneous, although from an intuitive standpoint, the model was close to the correct answer. Similar problems arise with the very definition of what is considered an event and what is not. In fact, most events are by definition associated with some conventional threshold (for example, the particle energy being greater than 10 MeV for SPE-type events). In this case, a false positive forecast may also be associated not with the global inaccuracy of the model but with the proximity to the threshold value. It may seem that such nuances are somehow leveled out against the majority background, but it is worth remembering that the total number of rare events is small, and an error in a single event can actually affect the entire model. Therefore, the practice is to use not one metric (for example, the fraction of correct predictions) but a set of metrics that characterize the predictive power of the model from different sides. And it is by no means guaranteed that one model that is better than another in terms of one metric would be better in terms of all metrics simultaneously. As an example, in Table 3, we show a number of metrics often encountered in papers on the forecasting of solar activity events. Because most binary classification metrics are calculated from the error matrix, the following notation is used: TP (true positive) is the number of correctly predicted positive classes (an event did occur), FP (false positive) is the number of false positive responses, TN (true negative) is the number of correctly predicted negative classes (no event), FN (false negative) is the number of false negative responses, $P = TP + FN$ is the number of positive classes in a sample, and $N = TN + FP$ is the number of negative classes in a sample.

We note yet another aspect encountered in assessing the model accuracy. The concept of a positive or negative forecast for logistic regression models is of a threshold nature: the model forecast is a number in the range from 0 to 1 (interpreted as the probability of a given object belonging to the positive class), which is then compared with the standard decision threshold 0.5. When the threshold changes, the error matrix and hence the metrics derived from it change. For example, when the threshold increases, the type-I error decreases but the type-II error increases. For an integral

Table 3. Binary classification metrics.

Name	Definition	Interpretation
Precision	$\frac{TP}{TP+FP}$	Fraction of correct answers in positive forecasts
Accuracy	$\frac{TP+TN}{N}$	Fraction of correct answers
False positive rate (FPR), type-I error	$\frac{FP}{N}$	False positive prediction probability
False negative rate (FNR), type-II error	$\frac{FN}{P}$	False negative prediction probability
Sensitivity, recall, true positive rate (TPR)	$\frac{TP}{P}$	True positive prediction probability, 1 – type-II error
Specificity, selectivity, true negative rate (TNR)	$\frac{TN}{N}$	True negative prediction probability, 1 – type-I error
F_1 score	$2\frac{\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$	Harmonic mean of precision and recall
True Skill Statistic (TSS)	$\frac{TP}{P}-\frac{FP}{N}$	1 – type-I error – type-II error

assessment of the effect of the threshold, the sensitivity is plotted against 1 – specificity by varying the threshold from 0 to 1. The resulting graph is called the receiver operating characteristic (ROC) curve. By definition, the graph is nondecreasing and connects the origin with the point $(1, 1)$. The area under the graph is called the ROC AUC (area under curve) metric; it is equal to 1 for an ideal classifier (Fig. 10). The ROC AUC metric has a simple probabilistic interpretation: it is the probability that the response of the model to a randomly selected example from the positive class will be higher than the response to a randomly selected example from the negative class.

Models for the prediction of active events are usually constructed for an active region that is a potential source of solar flares, CMEs, SPEs, and other events. Accordingly, the problem of identifying active regions arises, and the accuracy of the forecast depends on its solution. In machine learning, it is important that the models be based on the same sample, otherwise they are difficult to compare. A major contribution to solving this problem made in [63] was to propose an identification algorithm and a dataset of SHARP (Space-weather HMI Active Region Patches) active regions and their parameters on vector magnetograms. In a subsequent paper [99], the first model for predicting solar flares was presented using the SHARP catalogue and a machine learning model. The diversity of subsequent models and comparison of the results is a subject of a separate large review. In particular, a review of solar flare forecasting models is given in a series of studies [100–103] and in [104]. In a recent review of SPE forecasting models [105], more than 30 SPE forecasting models were presented. We discuss one of them based on a neural network model, following the original study [106].

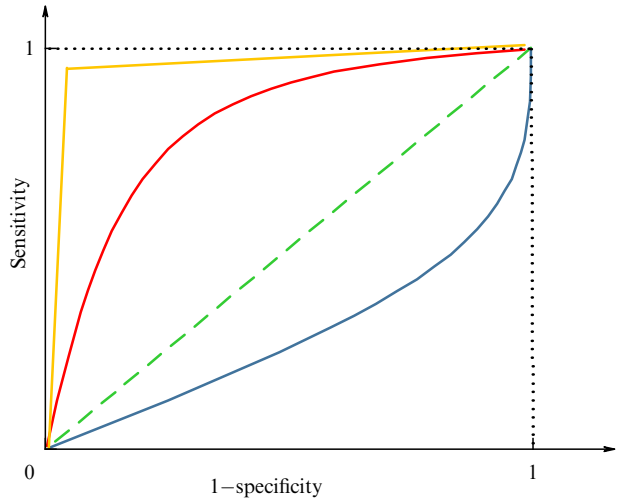

Figure 10. Examples of ROC curves for different classifiers. Each point on a curve corresponds to model accuracy metrics (sensitivity and 1 – specificity) for a certain classification threshold; to plot the curve, threshold is varied from zero to one. Green dashed line is a purely random forecast model. Red curve is a model that works better than random. Orange curve is a model that is close to an error-free one. Blue curve is a model that gives inverse predictions (meaning that interpreting its predictions exactly to the contrary would give a good classifier). Area under ROC curve is called ROC AUC metric.

The SPE forecasting model in [106] aggregates the parameters of magnetic active regions from the SHARP dataset observed during the day and forms a forecast for the SPE occurrence on the next day. The parameters of the active region pass through a fully connected neural network, which encodes them into a smaller number of variables. The parameters obtained from different active regions are combined into one vector, which is passed through a second fully connected neural network with a logistic activation function (sigmoid) on the output neuron. The model architecture is shown in Fig. 11. When training the network, the sample imbalance (1 to 34) was taken into account by duplicating positive examples many times. The proposed encoding–aggregation–forecast scheme is quite general and allows numerous variations. For example, it is possible to encode not the parameters of the active region but the image of the active region itself, use additional parameters from other telescopes during aggregation, and replace the neural network model with other machine learning models (see, e.g., [107]).

5.3.2 Forecasting solar activity indices. Unlike the problem of forecasting rare events, which in the standard setting is a classification problem, the forecast of solar activity indices (for example, the solar cycle, X-ray fluxes, and the K_p index of geomagnetic activity) is based on time series. The difference between the two standpoints is that classification is an

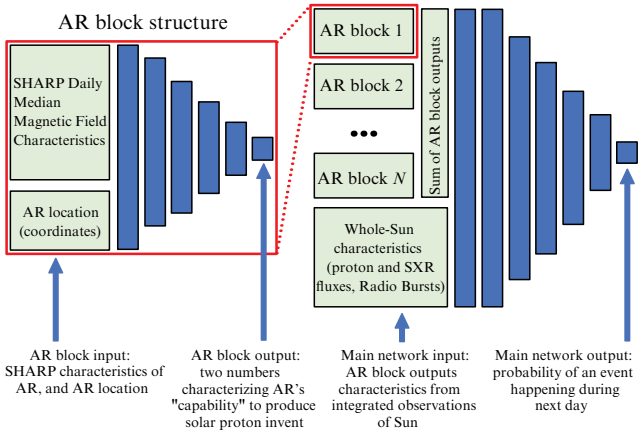

Figure 11. Neural network architecture for predicting SPEs. First neural network (AR block) encodes parameters of active region. Obtained parameters from different active regions are fed to input of second neural network, which outputs forecast. (Figure from [106].)

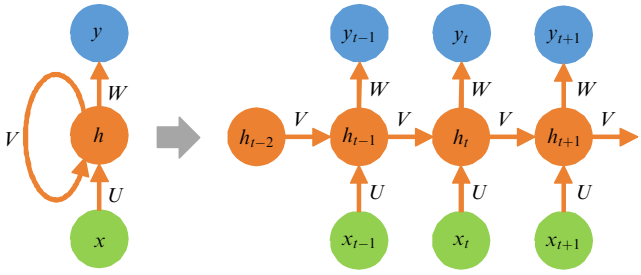

Figure 12. Structure of simplest recurrent neural network. Left of gray arrow: compact view. New internal state h_{t+1} is calculated based on current h_t and observed x_t values; output value y_t is a function of internal state h_t; U, V, and W are trainable parameters of the model. Right of gray arrow: expanded view of calculation of sequence of internal states and output values.

interpolation problem, while time series forecasting is an extrapolation problem. This difference also propagates into the structure of the models. As regards neural networks, fully connected and convolutional networks are more common in the first case, and recurrent neural networks (RNNs), in the second case. A feature of recurrent models is the memory mechanism, which allows accumulating information when passing through a sequence of unlimited length. Figure 12 shows the simplest example of an RNN, in which, for a current observed value x_t, the internal state h_t is updated by the rule

$$h_t = f(Ux_t + Vh_{t-1}), \tag{12}$$

and the output (predicted) value y_t is calculated as a function of the internal state,

$$y_t = g(Wh_t), \tag{13}$$

where f and g are activation functions, and U, V, and W are trainable parameters of the model.

In practice, more complex architectures are used that involve the concept of long short-term memory, analyze the sequence bidirectionally, and contain several levels of recurrent models (see reviews [108, 109]).

In 2021 (the beginning of the growth of the 25th solar cycle), a comparative graph of 34 forecasts of the amplitude of the 25th solar cycle was presented in [110], demonstrating a large uncertainty in the forecast for the future, even though all methods successfully predicted past cycles (Fig. 13).

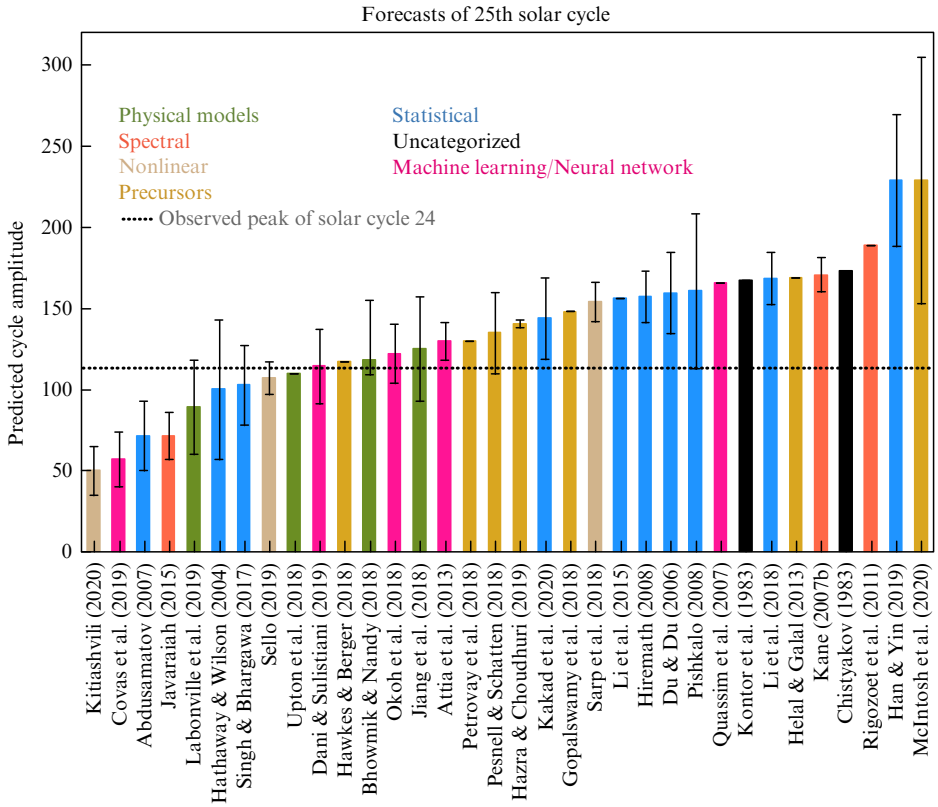

Figure 13. Forecasts of amplitude of 25th solar cycle. (Figure from [110].)

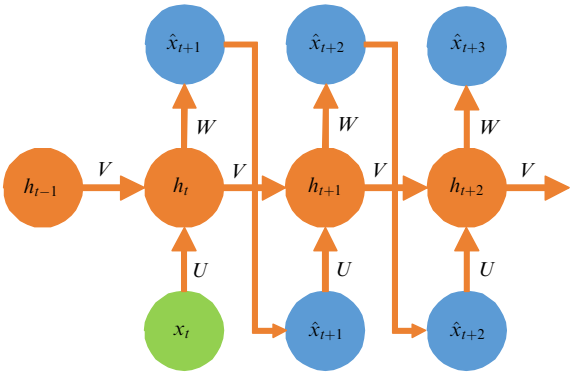

Figure 14. Forecast based on recurrent model: x_t is last observed value of time series. Forward forecast is built on the basis of preceding forecast values $\hat{x}_{t+k}$.

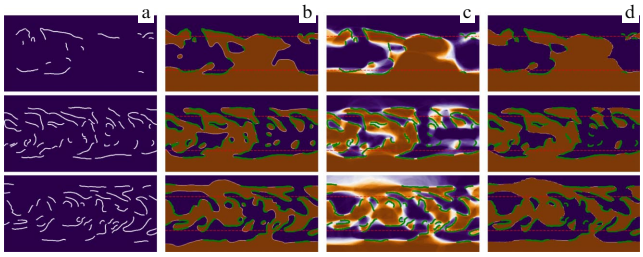

Figure 15. Reconstruction examples of a magnetic field polarity map for three synoptic rotations (by rows). (a) Synoptic map specifying positions of solar filaments. (b) Polarity map constructed by an expert [127]. (c) Averaging over 100 realizations of neural network reconstruction model. (d) Binarized version of c. (Figure from [130].)

In the machine learning framework, this situation can be explained as follows. A forecast of the next cycle maximum can be obtained either by directly predicting the maximum or by obtaining a forecast of the entire cycle, for example, using a recurrent model (Fig. 14), and then finding its maximum. In the first case, the size of the training sample is equal not to the number of years of observations of solar cycles but to the number of known maxima. Due to the large uncertainty of historical data, this number does not exceed several dozen. In the second case, a well-known problem arises: how to ensure the stability of a model that uses previous forecast values when constructing a forecast. For this, the model can be trained on a long forecast (of the order of the cycle length), but this again narrows the effective volume of the training sample to a size of the order of the number of solar cycles, as in the first case. Thus, a highly underdetermined problem arises (the volume of the training sample is several dozen, and a typical neural network model has orders of magnitude more free parameters), resulting in a large uncertainty in the forecast.

Nevertheless, studies of the possibilities of solar cycle forecasting using machine learning methods are ongoing (see, e.g., recent studies [111–113]), contributing to the study of fundamental problems in the nature of solar cyclicity [114].

The forecast of geomagnetic activity indices using machine learning methods is the subject of [115–120] and some others studies; the forecast of solar wind is the subject of [121–123] and some other studies (see review [124] for the problem of solar wind forecasting). We note that these studies address the problem of short-term forecasts (several days in advance), which significantly increases the volume and variability of the training sample and allows expecting a more stable forecast.

5.4 Reconstruction of observational data

A close relation exists between the different solar activity phenomena underlain by solar magnetic fields [125]. The presence of this intrinsic relation allows posing the machine learning problem to transform observational data of one type into others. As a first example, we consider the problem of reconstructing polarity maps of a large-scale magnetic field based on observations of solar filaments.

It is assumed that solar filaments (prominences) form along the neutral lines of the solar magnetic field. Knowing the positions of the filaments, we can estimate the position of the neutral line and then arrange the polarity signs such that different signs are on different sides of the neutral line. The finer details of this process were described in [126], but even taking them into account leaves the result of polarity map reconstruction ambiguous, and the main catalogues of polarity maps were created manually over a long period of time [127–129]. The idea of an automated approach, proposed recently in [130], is based on posing an optimization problem and solving it by a function defined by a fully connected neural network. A polarity map filled with plus and minus signs is regarded as the surface of a smooth function depending on the heliographic coordinates, where the plus sign corresponds to the function value $+1$ and the minus sign, to -1, the function value along the neutral line being 0. The optimization problem is constructed based on the idea that the filament coordinates define the coordinates at which the function is equal to zero, and on a number of physical constraints that prevent the convergence to the trivial (zero) solution. It turns out that the solution to such an optimization problem is close to a polarity map constructed manually by an expert (Fig. 15).

A finer structure of the magnetic field (e.g., coronal magnetic fields) can be reconstructed from photospheric observations. Typically, this is done using a force-free field model (system of equations),

$$\begin{aligned} &\nabla \cdot \mathbf{B} = 0\,, \\ &\big\|(\nabla \times \mathbf{B}) \times \mathbf{B}\big\| = 0\,, \end{aligned} \tag{14}$$

with photospheric magnetograms $\mathbf{B}_0(x, y)$ taken as the boundary conditions:

$$\mathbf{B}(x, y, 0) = \mathbf{B}_0(x, y)\,. \tag{15}$$

The numerical solution to this problem is greatly complicated by the fact that the observational data do not fully agree with the force-free field model [131], and a possible solution must violate one condition or the other. The situation that has arisen contradicts standard numerical methods for solving differential equations, but can be implemented in the framework of the PINN model. Let us recall that the idea is that the neural network defines a function of spatial coordinates, which is then optimized to satisfy the differential equation and boundary conditions as accurately as possible. In the context of reconstructing the coronal magnetic field, such an idea was demonstrated in [132, 133] (Fig. 16). In [134], the PINN model was used to simulate the propagation of shock waves in the solar atmosphere.

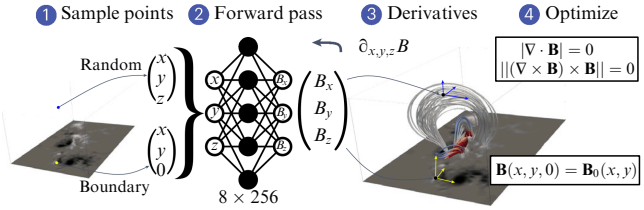

Figure 16. Coronal magnetic field reconstruction above an active region using PINN model. (Figure from [132].)

Another example of a reconstruction problem is the translation of solar disk images obtained in one spectral region into images from another spectral region (image-to-image translation). In [60], a convolutional neural network model was proposed for reconstructing SDO/AIA images in the UV range using SDO/HMI magnetograms. A solution to the inverse problem was presented in [135] using a generative cGAN model (see the example in Fig. 17). As an application, the authors of [135] presented magnetograms of the far side of the Sun based on STEREO/EUVI observations. Further development of translation methods and new combinations of images were studied, for instance, in [136–139].

The task of increasing the spatial resolution of images and reducing noise can also be assigned to this group of tasks. One of the first attempts to use a convolutional neural network in this context was made in [140]. The model was trained on pairs of images with different spatial resolutions obtained by magnetohydrodynamic (MHD) modeling, and testing was carried out on real SDO/HMI magnetograms (Fig. 18). For further studies of models based on machine learning, see, e.g., [141–145].

The problem of the reliability of the resulting reconstructions was discussed in [146]. The authors conclude that, although the results are statistically quite plausible, the reconstruction quality is significantly reduced for rare and powerful events.

The reconstruction problem is especially relevant when working with historical observational data stored in the form of handwritten notes and sketches. Using such data for systematic analysis requires digitization, and machine learning can help here when working with large catalogues. For example, in [147], the problem of digitizing more than 10,000 pages of handwritten tables specifying the positions of spots, faculae, and prominences from the Zurich Observatory archives was solved. The main difficulty was the large variability of handwriting, and therefore the model trained on a limited (albeit large) subsample of examples had difficulty recognizing text from a new handwriting. The solution was an adaptive approach: the model (a convolutional recurrent neural network, CRNN) was automatically retrained on each new page, using only the fact that the numbers in the tables with coordinates were not completely arbitrary but manifested some dependence. An example of the digitization result is shown in Fig. 19.

We note that the adaptive approach significantly extends the capabilities of machine learning models. In the classical approach, a model is trained on a sample consisting of pairs of independent and dependent variables (x_i, y_i), and is then used unaltered on new data (supervised learning). In practice, however, significant statistical changes can occur in the data over time, which causes the quality of the model to decline. Again, the standard solution is to prepare a new training sample and retrain the model. However, this is not always possible: changes may occur frequently, and the process of preparing training data usually takes a long time. Moreover, the fact that there is a decrease in the model quality is usually apparent not immediately but only after some time. Therefore, approaches are being developed in which models are automatically and continuously adjusted using self-training

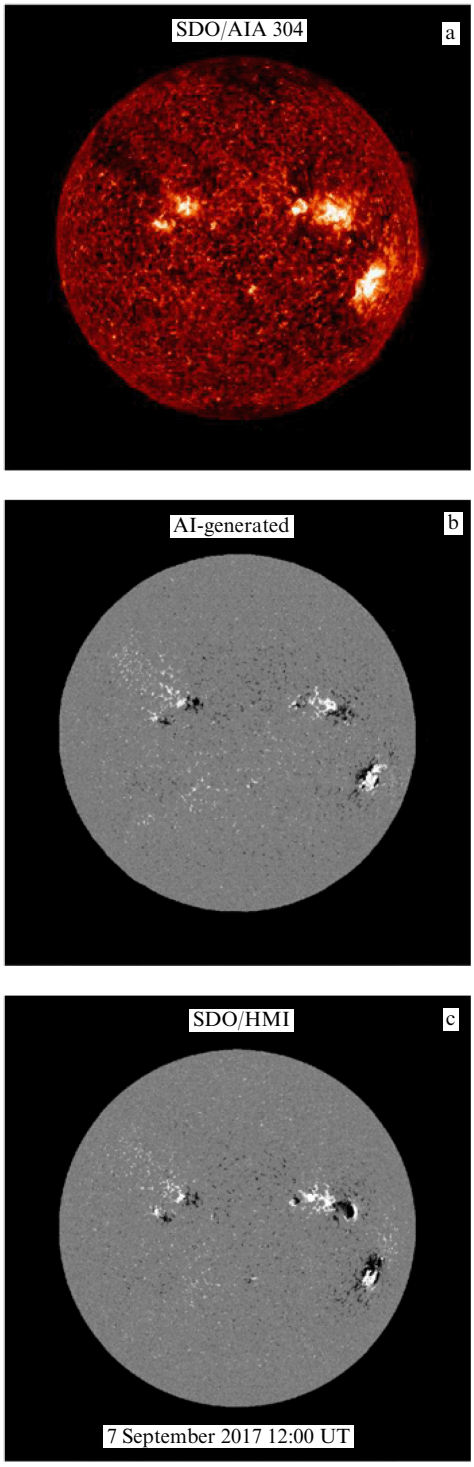

Figure 17. Example of operation of SDO/HMI magnetogram reconstruction model based on SDO/AIA image of solar disk in 304-Å line. (a) Original SDO/AIA image, (b) reconstruction result, and (c) original SDO/HMI image. (Figure from [135].)

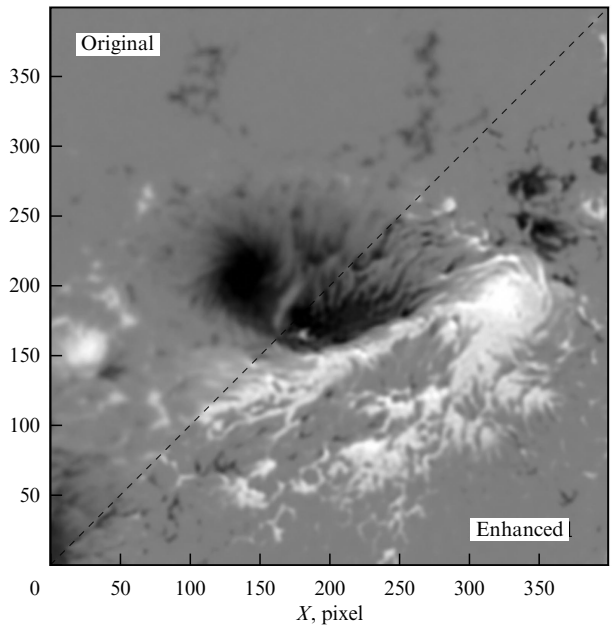

Figure 18. Example of operation of spatial resolution enhancement model for an SDO/HMI magnetogram. Upper left: original SDO/HMI image, lower right: image processed by the model. (Figure from [140].)

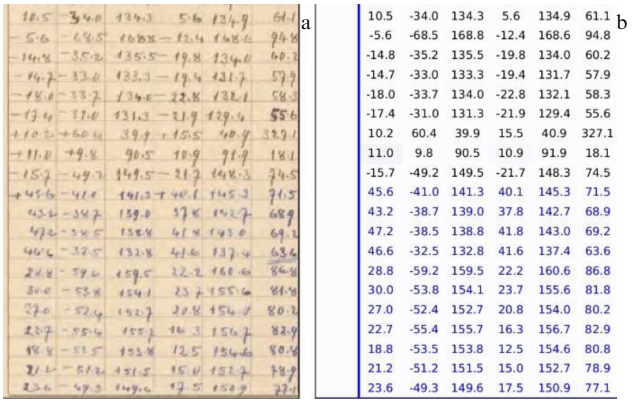

Figure 19. (a) Part of a page from Zurich Observatory catalogue and (b) result of its digitization using a neural network from [147].

methods (self-supervised learning). We consider just a few ideas on how training pairs (x_i, y_i) can be artificially created from an unlabeled sample (for which there are no target values y_i).

A simple example is provided by a text with one word removed from it, the model's task being to fill the gap. The removed word becomes the target value. The opposite example is to restore the context for a given word in a sentence. In this case, the word environment is declared to be the target value. A similar example can be given for images: a certain detail is painted over in the image, and the model's task is to restore the missing part based on the image with a gap. The tasks used for self-supervised learning also include a black-and-white conversion of a color image and the inverse task of coloring and rotating an image by a given angle and determining the rotation angle, and cutting an image into pieces and reassembling it from the pieces.

By construction, a self-supervised model does not directly solve the original problem (which could be, for example, an image classification problem), but solves some artificial problem. The rationale for such a substitution is to train the model to identify meaningful internal properties of objects or, in other words, to build a map sending them into a new feature space. In practice, it is often the case that the original problem (for example, classification) is solved much better in a new feature space than in the original one. We note that modern language models are built this principle: first, the model learns some general principles with a large sample, and then the resulting model is used as a basis for solving specific problems. For a more detailed review of self-supervised training methods and applications, see, e.g., [148–151].

5.5 Modeling dynamical processes

The most detailed picture of the stellar (solar) dynamo [152] is currently obtained by direct numerical modeling of magnetohydrodynamics equations. Due to the specific features of the problem, this approach requires an extremely large computational grid, but even the most advanced computing systems are capable of providing calculations on a grid whose size is many orders of magnitude smaller than the one following from theoretical estimates. In view of this and a number of other problems, direct modeling remains one of the most difficult problems. The key results in this field are reviewed in [153].

Another approach in the dynamo theory is associated with identifying and studying mean fields. Using it in numerical modeling methods has allowed reproducing the most significant regularities of large-scale magnetic fields of the Sun, such as, the magnetic cycle (Hale's polarity rule), butterfly diagrams, and more subtle effects. A detailed overview of the achievements and limitations of the mean-field dynamo theory is given in review papers [154, 155].

Machine learning methods are only beginning to be used in numerical modeling problems, but some areas are already shaping up. One is related to the acceleration of calculations in classical numerical schemes. For example, a method for estimating the initial approximation in Newton's tangent method for finding the root of a function was recently proposed in [156], which allowed reducing the number of iterations of the nonlinear solver. The idea is that the machine learning algorithm uses the results of the preceding calculation steps to estimate the solution at the current step, and this estimate is used as an initial approximation (Fig. 20). A special feature of the method is that it does not use pretrained models but is optimized directly during the calculation process. The results show that, in some examples, the proposed method allows reducing the total calculation time by 20%. Although testing was carried out on a hydrodynamics problem, it may well be assumed that the effect of accelerating the calculation can be achieved in MHD problems as well.

Another example of using machine learning is the closure of a system of equations for the joint description of large and small (subgrid) scales in turbulence modeling. In [157], an approach based on neural differential equations was used for this purpose. Recall that the essence of the approach is that the unknown function on the right-hand side of a differential equation is parameterized using a neural network, the calculation of the dynamics is launched using a standard numerical scheme, and the neural network is then optimized such that the resultant solution has the desired properties (for example, coincides with the dynamics available from observations or modeled by other methods).

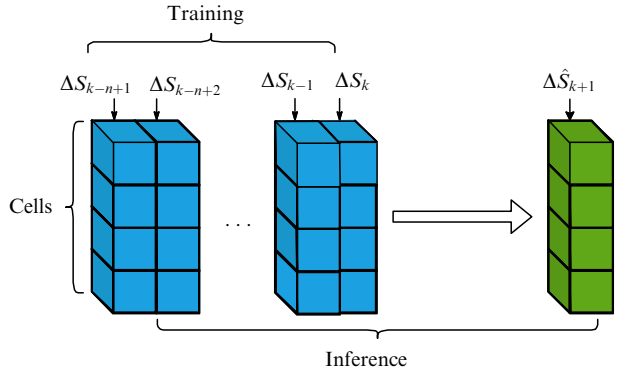

Figure 20. Scheme for estimating initial approximation for Newton's method. Columns represent computational grid at successive time instants. Solutions at preceding time steps and solution at current time step (blue bars) are used to train the model, with current solution being the target value. Solutions at previous time steps, together with solution at current time step, are then fed to input of the model to predict solution for next step (green column). Predicted value is used as initial approximation for Newton's method. (Figure from [156].)

The problem of completely replacing the 'classical' numerical solver with a machine learning model in modeling turbulence was studied, for example, in [158, 159]. In particular, the authors used the neural operator [160] and Fourier neural operator [161] models, the essence of which is as follows. In a domain $D \subset R^d$, we consider the equation defined by a differential operator $\mathcal{L}$,

$$\begin{aligned} &(\mathcal{L}_a u)(x) = f(x)\,, \quad x \in D\,, \\ &u(x) = 0\,, \quad x \in \partial D\,, \end{aligned} \tag{16}$$

where a are the coefficients of the equation (functions of x, including scalars), and the right-hand side f is a fixed function. The classical approach to solving the problem is based on dividing the space into cells, using difference schemes to approximate the equations, and solving the resulting linear system. The idea of the neural operator is to find the kernel k of the integral operator that maps the (a, f) pair to the solution u of the system:

$$u(x) = \int_D k\big(x, y, a(x), a(y)\big) f(y)\,\mathrm{d}y\,. \tag{17}$$

We note that the desired kernel is the Green's function, i.e., the inverse operator of $\mathcal{L}_a$. It can be defined (approximated), for example, using a fully connected neural network. To optimize the neural network, a sample $\{a_i, u_i\}_{i=1}^n$ of solutions of problem (16) for different coefficients a is used. The target solutions u_i can be obtained by a classical solver or known from observational (experimental) data. The constructed operator is then used to evaluate the solution of problem (16) for new values of a. The main advantage is the speed of solving the problem, which can be significantly higher than when solving using classical schemes.

We compare the classical numerical approach and the solution based on machine learning. The advantage of the first approach is that it allows obtaining a solution for any admissible values of the control parameters, has accuracy guarantees, and gives rise to a physically consistent solution. The main disadvantage is the speed of the solution when using a fine grid. A solution based on machine learning (neural network operator), on the one hand, is orders of magnitude faster, but the model requires pretraining, which takes significant time (primarily for creating a training sample). In addition, the obtained model can be applied reasonably only in the range of equation parameters from which the training examples were taken. The accuracy of the obtained solution is difficult to control in advance (theory only deals with accuracy in the limit of an infinitely growing sample), and there are no guarantees of the physical consistency of the solution. Despite the apparently impressive list of 'shortcomings,' speed *is* sometimes a decisive factor from a practical standpoint. For example, this tool can be useful when it is necessary to simulate a large number of scenarios for a preliminary assessment of statistical properties or to select some scenarios useful for further work. In such situations, a fast but not entirely accurate solution can be more practical than a slow but accurate one.

A more detailed review and implementation options for neural network operators are presented in reviews [162, 163], and a review of applications and methods of machine learning in a wider range of plasma physics problems is given in [164].

6. Conclusions

Machine learning methods have shaped a new direction in solar physics research. Its key components are:

- large data sets,
- data processing tools,
- machine learning models.

One of the first important effects of this new direction, which had an impact on the entire field of research as a whole, was the emergence of large and homogeneous data sets prepared for statistical research. This has allowed redirecting the time and effort that was previously spent on preparing data to their direct analysis, and resulted in an increased number of publications. This also made the results of independent studies comparable and provided a basis for their reproduction, which are the most important features of scientific research. Big data have become an independent value in solar physics in particular and in science in general.

The second significant effect was the development of basic tools for working with data and their publication in the public domain. This gives rise to the unification of data preparation procedures and hence to the reproducibility of studies.

Third, machine learning models have allowed mathematically formulating and studying problems that previously remained poorly formalized. For example, over the 400 years of observations, the problem of identifying the boundary of a sunspot was to be solved by each observer individually. Machine learning methods allow posing the problem of reproducing a specific data processing technique and extending it to longer data series.

The research design in solar physics has changed in many ways. Research is becoming interdisciplinary, at the intersection of physical science and computing technologies. Theoretical physicists, machine learning specialists, and computing infrastructure specialists are taking part in the work. Large computing centers are becoming the venues for conducting research.

Applications of machine learning methods are currently being studied in almost all areas of solar physics. Specialized conferences are devoted to the discussion of the results, leading journals on solar physics aggregate papers on machine learning methods into thematic issues, and reviews (e.g., [165, 166]) and books are published; for example:

• E. Camporeale et al., *Machine Learning Techniques for Space Weather* [167],

• M. Bobra and J. Mason, *Machine Learning, Statistics, and Data Mining for Heliophysics* [168],

• L. Xu et al., *Deep Learning in Solar Astronomy* [169].

Machine learning is a new research tool, but the role of the researcher remains paramount. Machine learning-based models look promising in applied problems that require high-speed processing of large and/or multidimensional data sets and support for a given methodology. As part of the development of the general theory, machine learning models can be used to describe relations not yet covered by theory and for subsequent mathematical modeling.

Current and future tasks for machine learning are characterized by the scale of the data involved and the complexity of the models and training procedures. As in any science, reaching such a level does not happen overnight, but requires many years of practice, experience, and a modern technological base. Our country has all the components for such research, and it is only necessary to keep working systematically to stay at the forefront of solar physics.

References

1. Markov A A "Primer statisticheskogo issledovaniya nad tekstom "Evgeniya Onegina", illyustriruyushchii svyaz' ispytanii v tsep"' ("An example of a statistical study of the text of "Eugene Onegin", illustrating the link of trials in a chain") *Izv. Imperatorskoi Akad. Nauk* **7** (3) 153 (1913)
2. Vaswani A et al., arXiv:1706.03762
3. Devlin J et al., arXiv:1810.04805
4. Brown T B et al., arXiv:2005.14165
5. Dosovitskiy A et al., arXiv:2010.11929
6. Yan W et al., arXiv:2104.10157
7. Lin T et al., arXiv:2106.04554
8. Bommasani R et al., arXiv:2108.07258
9. Vokhmyanin M, Arlt R, Zolotova N *Solar Phys.* **295** (3) 39 (2020)
10. Rimmele T R et al. *Solar Phys.* **295** (12) 172 (2020)
11. Clette F et al. *Solar Phys.* **298** (3) 44 (2023)
12. Lemen J R et al. *Solar Phys.* **275** (1–2) 17 (2012)
13. Kolmogorov A N *Dokl. Akad. Nauk SSSR* **108** 179 (1956)
14. Liu Z et al., arXiv:2404.19756
15. Cybenko G *Math. Control Signals Syst.* **2** 303 (1989)
16. Hornik K, Stinchcombe M, White H *Neural Networks* **2** (5) 359 (1989)
17. Funahashi K-I *Neural Networks* **2** (3) 183 (1989)
18. Pinkus A *Acta Numerica* **8** 143 (1999)
19. DeVore R, Hanin B, Petrova G *Acta Numerica* **30** 327 (2021)
20. Murphy K P *Machine Learning: A Probabilistic Perspective* (Cambridge, MA: MIT Press, 2021)
21. Bishop C M *Pattern Recognition and Machine Learning* (Information Science and Statistics) (New York: Springer-Verlag, 2006)
22. Hastie T, Tibshirani R, Friedman J *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer Series in Statistics) (New York: Springer, 2009) https://doi.org/10.1007/978-0-387-84858-7
23. Johnson J, Alahi A, Fei-Fei L, arXiv:1603.08155
24. Bakhvalov N S, Zhidkov N P, Kobel'kov G M *Chislennye Metody* (Numerical Methods) (Moscow: Nauka, 1987)
25. Raissi M, Perdikaris P, Karniadakis G E *J. Comput. Phys.* **378** 686 (2019)
26. Goodfellow I, Bengio Y, Courville A *Deep Learning* (Cambridge, MA: MIT Press, 2016)
27. Chen R T Q et al., arXiv:1806.07366
28. Pontryagin L S et al. *The Mathematical Theory of Optimal Processes* (New York: Interscience Publ., 1962); Translated from Russian: *Matematicheskaya Teoriya Optimal'nykh Protsessov* (Moscow: Fizmatgiz, 1961)
29. Pearson K *Philos. Mag.* **2** (11) 559 (1901)
30. Johnstone I M, Paul D *Proc. IEEE* **106** 1277 (2018)
31. Kingma D P, Welling M, arXiv:1312.6114
32. Kingma D P, Welling M *Found. Trends Machine Learn.* **12** (4) 307 (2019)
33. Rezende D, Mohamed Sh, in *Proc. of the 32nd Intern. Conf. on Machine Learning, Lille, France, 7–9 Jul 2015* (Proc. of Machine Learning Research, Vol. 37, Eds F Bach, D Blei) (MLR Press, 2015) p. 1530
34. Kobyzev I, Prince S J D, Brubaker M A *IEEE Trans. Pattern Analysis Machine Intelligence* **43** 3964 (2021)
35. Goodfellow I J et al., arXiv:1406.2661
36. Arjovsky M, Chintala S, Bottou L, arXiv:1701.07875
37. Kovachki N et al., arXiv:2108.08481
38. Jouppi N P et al., arXiv:1704.04760
39. Zeiler M D, arXiv:1212.5701
40. Kingma D P, Ba J, arXiv:1412.6980
41. Duchi J, Hazan E, Singer Y *J. Machine Learning Res.* **12** 2121 (2011)
42. Chen X et al., arXiv:2302.06675
43. McMahan B, in *Proc. of the Fourteenth Intern. Conf. on Artificial Intelligence and Statistics* (Proc. of Machine Learning Research, Vol. 15, Eds G Gordon, D Dunson, M Dudík) (MLR Press, 2011) p. 525
44. Bottou L, in *On-Line Learning in Neural Networks* (Ed. D Saad) (Cambridge: Cambridge Univ. Press, 1999) p. 9, https://doi.org/10.1017/CBO9780511569920.003
45. Polyak B T, Juditsky A B *SIAM J. Control Optim.* **30** 838 (1992)
46. Zhu C et al. *ACM Trans. Math. Software* **23** 550 (1997)
47. Riedmiller M, Braun H, in *IEEE Intern. Conf. on Neural Networks, 28 March 1993–01 April 1993, San Francisco, CA, USA* (Piscataway, NJ: IEEE, 1993) p. 586, https://doi.org/10.1109/ICNN.1993.298623
48. Rall L B (Ed.) *Automatic Differentiation: Techniques and Applications* (Lecture Notes in Computer Science, Vol. 120) (Berlin: Springer-Verlag, 1981) https://doi.org/10.1007/3-540-10861-0
49. Baydin A G et al. *J. Machine Learning Res.* **18** (153) 1 (2018)
50. Rumelhart D E, Hinton G E, Williams R J *Nature* **323** 533 (1986)
51. Hecht-Nielsen R, in *Intern. 1989 Joint Conf. on Neural Networks, IJCNN, Washington DC, 18–22 June 1989* Vol. 1 (Piscataway, NJ: IEEE, 1989) p. 593, https://doi.org/10.1109/IJCNN.1989.118638
52. Schmidhuber J *Neural Networks* **61** 85 (2015)
53. Yao Y, Rosasco L, Caponnetto A *Constr. Approx.* **26** 289 (2007)
54. Hurlburt N et al. *Solar Phys.* **275** 67 (2012)
55. Müller D et al. *Astron. Astrophys.* **606** A10 (2017)
56. Sadykov V M et al. *Astrophys. J. Suppl.* **231** (1) 6 (2017)
57. Illarionov E, Kosovichev A, Tlatov A *Astrophys. J.* **903** 115 (2020)
58. Zhao Z et al. *Sci. Data* **10** 178 (2023)
59. Kucuk A, Banda J M, Angryk R A *Sci. Data* **4** 170096 (2017)
60. Galvez R et al. *Astrophys. J. Suppl.* **242** (1) 7 (2019)
61. Illarionov E A, Tlatov A G *Izv. Krym. Astrofiz. Observ.* **114** (1) 118 (2018)
62. Bolzern R, Aerni M "A machine learning image dataset for the prediction of solar flares" (2024)
63. Kosovichev A G et al., in *AGU Fall Meeting 2023, San Francisco, CA, 11–15 December 2023*, id. SH33C-307, Poster No. 307
64. Palmroos C et al. *Front. Astron. Space Sci.* **9** 395 (2022)
65. Bobra M G et al. *Solar Phys.* **289** 3549 (2014)
66. Bobra M G et al. *Astrophys. J. Suppl.* **256** 26 (2021)
67. Rotti S A, Martens P C H, Aydin B *Astrophys. J. Suppl.* **249** (2) 20 (2020)
68. Boucheron L E et al. *Sci. Data* **10** 825 (2023)
69. Gallagher P T, Moon Y-J, Wang H *Solar Phys.* **209** 171 (2002)
70. Illarionov E, Tlatov A *Solar Phys.* **297** (2) 19 (2022)
71. Angryk R A et al. *Sci. Data* **7** (1) 227 (2020)
72. Freeland S L, Handy B N *Solar Phys.* **182** (2) 497 (1998)
73. Barnes W T et al. *J. Open Source Software* **5** (55) 2801 (2020)
74. McGregor S et al., in *Workshop on Deep Learning for Physical Sciences, DLPS 2017, NIPS 2017, Long Beach, CA, USA, 2017*
75. Shneider C et al., arXiv:2108.06394
76. The SunPy Community, Barnes W T et al. *Astrophys. J.* **890** 68 (2020)
77. Illarionov E A, Tlatov A G *Mon. Not. R. Astron. Soc.* **481** 5014 (2018)
78. Ronneberger O, Fischer P, Brox T, in *Medical Image Computing and Computer-Assisted Intervention — MICCAI 2015. 18th Intern.*

Conf., Munich, Germany, October 5–9, 2015, Proc. Pt. 3 (Lecture Notes in Computer Science, Vol. 9351, Eds N Navab et al.) (Cham: Springer, 2015) p. 234, https://doi.org/10.1007/978-3-319-24574-4_28; arXiv:1505.04597
79. Reiss M A et al. *Astrophys. J.* **913** 28 (2021)
80. Reiss M A et al. *Astrophys. J. Suppl.* **271** 6 (2024)
81. Verbeeck C et al. *Astron. Astrophys.* **561** A29 (2014)
82. Garton T M, Gallagher P T, Murray S A *J. Space Weather Space Clim.* **8** A02 (2018) https://doi.org/10.1051/swsc/2017039
83. Zhu G et al. *Solar Phys.* **294** 117 (2019)
84. Jiang H et al. *Astrophys. J. Suppl.* **250** 5 (2020)
85. Mackovjak Š et al. *Mon. Not. R. Astron. Soc.* **508** 3111 (2021)
86. Díaz Castillo S M et al. *Front. Astron. Space Sci.* **9** 896632 (2022)
87. Jarolim R et al. *Astron. Astrophys.* **652** A13 (2021)
88. Baek J-H et al. *Solar Phys.* **296** 160 (2021)
89. Ahmadzadeh A et al., arXiv:1912.02743
90. Guo X et al. *Solar Phys.* **297** 104 (2022)
91. Diercke A et al. *Astron. Astrophys.* **686** A213 (2024)
92. McIntosh P S *Solar Phys.* **125** 251 (1990)
93. Makarenko N G, Knyazeva I S, Karimova L M *Astron. Lett.* **38** 531 (2012); *Pis'ma Astron. Zh.* **38** 597 (2012)
94. Sadykov V M et al., in *2021 Intern. Conf. on Content-Based Multimedia Indexing CBMI, 28–30 June 2021, Lille, France* (Piscataway, NJ: IEEE, 2021) p. 1, https://doi.org/10.1109/CBMI50038.2021.9461879
95. Giger M, Csillaghy A *Space Weather* **22** e2023SW003516 (2024)
96. Brown E et al. "Learning the solar latent space: sigma-variational autoencoders for multiple channel solar imaging", in *Fourth Workshop on Machine Learning and the Physical Sciences, NeurIPS 2021*
97. Wang J et al. *Front. Astron. Space Sci.* **10** 1082737 (2023) https://doi.org/10.3389/fspas.2023.1082737
98. Ahmadzadeh A et al. *Astrophys. J. Suppl.* **254** 23 (2021)
99. Bobra M G, Couvidat S *Astrophys. J.* **798** 135 (2015)
100. Barnes G et al. *Astrophys. J.* **829** 89 (2016)
101. Leka K D et al. *Astrophys. J. Suppl.* **243** 36 (2019)
102. Leka K D et al. *Astrophys. J.* **881** 101 (2019)
103. Park S-H et al. *Astrophys. J.* **890** 124 (2020)
104. Georgoulis M K et al. *J. Space Weather Space Clim.* **11** 39 (2021)
105. Whitman K et al. *Adv. Space Res.* **72** 5161 (2023)
106. Sadykov V et al., arXiv:2107.03911
107. Ali A et al. *Astrophys. J. Suppl.* **270** 15 (2024)
108. Lipton Z C, Berkowitz J, Elkan C, arXiv:1506.00019
109. Sherstinsky A *Physica D* **404** 132306 (2020)
110. Nandy D *Solar Phys.* **296** 54 (2021)
111. Wang Q-J, Li J-C, Guo L-Q *Res. Astron. Astrophys.* **21** (1) 012 (2021)
112. Bizzarri I et al. *Mon. Not. R. Astron. Soc.* **515** 5062 (2022)
113. Prasad A et al. *Solar Phys.* **298** 50 (2023)
114. Obridko V N, Nagovitsyn Yu A *Solnechnaya Aktivnost', Tsiklichnost' i Metody Prognoza* (Solar Activity, Cyclicity, and Prediction Methods) (St. Petersburg: VVM, 2017)
115. Myagkova I, Shiroky V, Dolenko S *E3S Web Conf.* **20** 02011 (2017)
116. Efitorov A O et al. *Cosmic Res.* **56** 434 (2018); *Kosmich. Issled.* **56** 420 (2018)
117. Tlatov A G, Illarionov E A, Berezin I A, Shramko A D *Cosmic Res.* **58** 444 (2020); *Kosmich. Issled.* **58** 479 (2020)
118. Bernoux G et al. *J. Geophys. Res. Space Phys.* **127** e2022JA030868 (2022)
119. Vladimirov R D et al. *Geomagn. Aeronom.* **63** (2) 190 (2023)
120. Telloni D et al. *Astrophys. J.* **952** 111 (2023)
121. Upendran V et al. *Space Weather* **18** (9) e02478 (2020) https://doi.org/10.1029/2020SW002478
122. Brown E J E et al. *Space Weather* **20** (3) e2021SW002976 (2022)
123. Son J et al. *Astrophys. J. Suppl.* **267** 45 (2023)
124. Petrukovich A A et al. *Phys. Usp.* **63** 801 (2020); *Usp. Fiz. Nauk* **190** 859 (2020)
125. Vitinskii Yu I *Solnechnaya Aktivnost'* (Solar Activity) (Moscow: Nauka, 1983)
126. McIntosh P S, in *Solar Activity Observations and Predictions* (Progress in Astronautics and Aeronautics, Vol. 30, Eds P S McIntosh, M Dryer) (Cambridge, MA: MIT Press, 1972) p. 65; Translated into Russian: in *Nablyudeniya i Prognoz Solnechnoi Aktivnosti* (Eds P S McIntosh, M Dryer) (Moscow: Mir, 1976)
127. McIntosh P S "Synoptic maps composites observed from McIntosh" (1964) https://doi.org/10.7289/V5765CCQ
128. Makarov V I, Fatianov M P, Sivaraman K R *Solar Phys.* **85** 215 (1983)
129. Makarov V I, Sivaraman K R *Solar Phys.* **85** 227 (1983)
130. Kisielius V, Illarionov E *Solar Phys.* **299** (5) 69 (2024)
131. Metcalf T R et al. *Astrophys. J.* **439** 474 (1995)
132. Jarolim R et al. *Nat. Astron.* **7** 1171 (2023)
133. Baty H, Vigon V *Mon. Not. R. Astron. Soc.* **527** 2575 (2024)
134. Moschou S P et al. *Machine Learn. Sci. Technol.* **4** 035032 (2023)
135. Kim T et al. *Nat. Astron.* **3** 397 (2019)
136. Park E et al. *Astrophys. J. Lett.* **884** L23 (2019)
137. Son J et al. *Astrophys. J.* **920** 101 (2021)
138. Jeong H-J et al. *Astrophys. J. Suppl.* **262** (2) 50 (2022)
139. Lawrance B et al. *Astrophys. J.* **937** (2) 111 (2022)
140. Díaz Baso C J, Asensio Ramos A *Astron. Astrophys.* **614** A5 (2018)
141. Díaz Baso C J, de la Cruz Rodríguez J, Danilovic S *Astron. Astrophys.* **629** A99 (2019)
142. Rahman S et al. *Astrophys. J. Lett.* **897** L32 (2020)
143. Xu C et al. *Solar Phys.* **299** 36 (2024)
144. Muñoz-Jaramillo A et al. *Astrophys. J. Suppl.* **271** 46 (2024)
145. Song W et al. *Astron. Astrophys.* **686** A272 (2024)
146. Salvatelli V et al. *Astrophys. J.* **937** 100 (2022)
147. Illarionov E, Arlt R *Solar Phys.* **297** 79 (2022)
148. Rani V et al. *Arch. Computat. Methods Eng.* **30** 2761 (2023)
149. Huang S-C et al. *npj Digit. Med.* **6** 74 (2023)
150. Shwartz Ziv R, LeCun Y *Entropy* **26** (3) 252 (2024)
151. Zhao Z et al. *Expert Syst. Appl.* **242** 122807 (2024)
152. Sokoloff D D *Phys. Usp.* **58** 601 (2015); *Usp. Fiz. Nauk* **185** 643 (2015)
153. Käpylä P J et al. *Space Sci. Rev.* **219** 58 (2023)
154. Brandenburg A, Sokoloff D, Subramanian K *Space Sci. Rev.* **169** 123 (2012)
155. Brandenburg A et al. *Space Sci. Rev.* **219** 55 (2023)
156. Petrosyants M, Trifonov V, Illarionov E, Koroteev D *Comput. Geosci.* **28** 605 (2024) https://doi.org/10.1007/s10596-024-10284-z
157. Williams J, Wolfram U, Ozel A *Phys. Fluids* **34** 113315 (2022)
158. Peng W, Yuan Z, Wang J *Phys. Fluids* **34** 025111 (2022)
159. Rosofsky S G, Huerta E A *Mach. Learn. Sci. Technol.* **4** 035002 (2023)
160. Lu L, Jin P, Karniadakis G E *Nat. Mach. Intell.* **3** 218 (2021); arXiv:1910.03193
161. Li Z et al., arXiv:2010.08895
162. Kovachki N et al. *J. Mach. Learn. Res.* **24** (89) 1 (2023)
163. Azizzadenesheli K et al. *Nat. Rev. Phys.* **6** 320 (2024)
164. Anirudh R et al. *IEEE Trans. Plasma Sci.* **51** 1750 (2023)
165. Asensio Ramos A et al. *Living Rev. Solar Phys.* **20** 4 (2023)
166. Illarionov E A, Sadykov V M *Zemlya Vselennaya* (4) 35 (2021) https://doi.org/10.7868/S0044394821040034
167. Camporeale E, Wing S, Johnson J R (Eds) *Machine Learning Techniques for Space Weather* (Cambridge, MA: Elsevier, 2018)
168. Bobra M G, Mason J P "Machine Learning, Statistics, and Data Mining for Heliophysics" (2020)
169. Xu L, Yan Y, Huang X *Deep Learning in Solar Astronomy* (SpringerBriefs in Computer Science) 1st ed. (Singapore: Springer, 2022) https://doi.org/10.1007/978-981-19-2746-1