Nonlinear Dynamics and Neuroscience

Izvestiya Vysshikh Uchebnykh Zavedeniy. Applied Nonlinear Dynamics. 2024;32(1)

Article DOI: 10.18500/0869-6632-003079

Modeling of the Hodgkin–Huxley neural oscillators dynamics using an artificial neural network

P. V. Kuptsov[1,2], *N. V. Stankevich*[2,1]

[1]Saratov Branch of Kotelnikov Institute of Radioengineering and Electronics of the RAS, Russia
[2]HSE University, Nizhny Novgorod, Russia
E-mail: ✉kupav@mail.ru, stankevichnv@mail.ru

Received 27.04.2023, accepted 8.09.2023, available online 10.12.2023, published 31.01.2024

Abstract. The *purpose* of this study — to represent a detailed description of the procedure for creating and training a neural network mapping on the example of the dynamics modeling of a neural oscillator of the Hodgkin –Huxley type; to show that the neural network mappings trained for single oscillators can be used as elements of a coupled system that simulate the behavior of coupled oscillators. *Methods.* Numerical method is used for solving stiff systems of ordinary differential equations. Also a procedure for training neural networks based on the method of back propagation of error is employed together with the Adam optimization algorithm, that is a modified version of the gradient descent supplied with an automatic step adjustment. *Results.* It is shown that the neural network mappings built according to the described procedure are able to reproduce the dynamics of single neural oscillators. Moreover, without additional training, these mappings can be used as elements of a coupled system for the dynamics modeling of coupled neural oscillator systems. *Conclusion.* The described neural network mapping can be considered as a new universal framework for complex dynamics modeling. In contrast to models based on series expansion (power, trigonometric), neural network mapping does not require truncating of the series. Consequently, it allows modeling processes with arbitrary order of nonlinearity, hence there are reasons to believe that in some aspects it will be more effective. The approach developed in this paper based on the neural network mapping can be considered as a sort of an alternative to the traditional numerical methods of modeling of dynamics. What makes this approach topical is the current rapid development of technologies for creating fast computing equipment that supports neural network training and operation.

Keywords: neural network mapping, neural network, dataset, neural network learning, neuromorphic dynamics, numerical simulation.

Acknowledgements. The study of mathematical models was carried out within the framework of the Mirror Laboratories Project of HSE University (Sections 1, 2). Development and research of the neural network mapping (Sections 3–5) was supported by the Russian Science Foundation, 20-71-10048.

For citation: Kuptsov PV, Stankevich NV. Modeling of the Hodgkin–Huxley neural oscillators dynamics using an artificial neural network. Izvestiya VUZ. Applied Nonlinear Dynamics. 2024;32(1):72–95. DOI: 10.18500/0869-6632-003079

Introduction

The study of the dynamics of nonlinear systems largely boils down to processing and analysis of data generated during the evolution of these systems. By this reason for enriching the scientific tools of nonlinear dynamics and for expanding the range of tasks to be solved, it seems natural to turn to experience currently rapidly developing disciplines in the field of machine learning and data science.

Without pretending to be complete, as an example of such interdisciplinary enrichment Several works can be mentioned: the use of neural networks for adaptive modeling and control of systems [1], application of neural networks for reconstructing the El Niño attractor [2], use neural networks to predict dynamics based on the performed reconstruction state space [3], use of machine training for building models and analyzing the dynamics of biological systems [4].

One of the most important elements of the foundation of productive interaction of nonlinear dynamics with data science and machine learning is a series of theorems, with mathematical rigor justifying the possibility of approximating arbitrary functions of many variables using neural networks [5–10]. From these works it follows that already a two-layer fully connected network with a sigmoid function activation is sufficient to approximate any function with a given accuracy many variables. The approximation accuracy is determined by the size of the network layers, and also by choosing an effective learning algorithm. In the most general terms this means that for solving problems related to the search or analysis of functional dependencies, you can use neural networks. One more element, no longer strictly mathematical, but rather heuristic is the ability of neural networks to generalization. With proper organization of the network learning process, it is not easy remembers the data presented to it, and isolates the most significant from them, their most common signs. Subsequently, this allows the already trained network effectively process new data samples that were not presented to it during the process training [11].

This paper explores the applicability of neural networks for modeling nonlinear dynamics: it seems interesting to find a universal architecture, which could, after appropriate training, with a given accuracy reproduce any or almost any behavior of nonlinear systems. In favor the possibility of constructing such an architecture is indicated by the above-mentioned theorems about approximation and generalization ability of neural networks. Theoretical motivation such research is to obtain another universal model of dynamics, along with with models widely used today, for example, based on power or trigonometric series. It can be expected that the use of a neural network model will be in some aspects more effective compared to models based on series expansions. The latter always take into account limited, often small the number of terms of the series and therefore the order of the modeled nonlinearity is also limited. In the case of a neural network, this limitation is removed. Besides, of interest is the ability of neural networks to generalize — ideally even with a limited amount of data, one can hope that the network will be in able to extract from them information about dynamic phenomena that is inaccessible to other modeling methods. From a practical point of view, dynamics modeling using neural networks can be considered as a kind of alternative numerical methods for solving model equations. Traditional numerical methods simulations are best implemented on computers with classical architecture and often even parallelizing them into several computing cores causes difficulties. At the same time, modern computing technology is developing towards adaptation to support the work of neural networks. Examples include gaming video cards and so-called AI accelerators [12–15]. Therefore one can expect that the development of methods for modeling dynamics using neural networks will have great practical significance.

Previously, in the work of [16] it was shown that even the simplest two-layer the network can be trained to reproduce quite different types of dynamics (systems Lorenz and Rössler), as well as the Hindmarsh-Rose neuron model. Demonstrated good agreement between bifurcation patterns, Fourier spectra and indices Lyapunova. The work of [17] proposes a more complex network structure, when each of the variables is modeled by a separate subnetwork. Such a network copes with modeling of rigid dynamics, when system variables have different time scales. It was shown that the network reproduces the behavior of the model physiological neuron specified by a system of equations formulated in based on the Hodgkin-Huxley formalism [18]. It has been demonstrated that Due to its ability to generalize, a neural network model can successfully reproduce the bistability mode even when in the process of learning it only one of the branches of the solution was presented. Neural network model trained only on the oscillatory solution, also finds a stable solution coexisting with it fixed point, and in addition correctly reproduces its dependence eigenvalues of parameters. In this work we will consider a neural network the model proposed in the article [17]. Purpose of the study is demonstrate that such models trained on single systems can be done without additional training used to reproduce the dynamics of related systems It will be shown that there is good qualitative and quantitative correspondence of various dynamic modes of coupled systems and their bifurcation transformations.

1. Model neuron based on the Hodgkin–Huxley formalism

We consider a model neuron whose equations are derived based on Hodgkin-Huxley formalism [18]. In addition to the original system, we We will also consider its modified version, proposed in work [19]. The original system in the oscillation mode has unstable fixed point. The modification leads to the fact that in space parameters, an area appears in which the fixed point stable, that is, bistability occurs in the system, characteristic of neural models [20–22]. Since she is responsible for this modification is activated only in a small area nearby fixed point, this has virtually no visible effect on the quality nature of the oscillatory solution.

$$\begin{aligned} \tau\dot{V} &= -I_{Ca}(V) - I_K(V,n) - I_{K2}(V) - I_S(V,S), \\ \tau\dot{n} &= \sigma\,[n_\infty(V) - n], \\ \tau_S\dot{S} &= S_\infty(V) - S. \end{aligned} \tag{1}$$

Table 1. Parameters of the system (1)

$\tau = 0.02\,\mathrm{s}$	$\tau_S = 35\,\mathrm{s}$	$\sigma = 0.93$	
$g_{Ca} = 3.6$	$g_K = 10$	$g_S = 4$	$g_{K2} = 0.12$
$V_{Ca} = 25\,\mathrm{mV}$	$V_K = -75\,\mathrm{mV}$		
$\theta_m = 12\,\mathrm{mV}$	$\theta_n = 5.6\,\mathrm{mV}$	$\theta_S = 10\,\mathrm{mV}$	$\theta_p = 1\,\mathrm{mV}$
$V_m = -20\,\mathrm{mV}$	$V_n = -16\,\mathrm{mV}$	$V_S = -36\,\mathrm{mV}$	$V_p = -49.5\,\mathrm{mV}$

Here V, n and S are dynamic variables. Functions included in the equations are given by the formulas (2), and the numeric values used parameters are summarized in table 1.

$$I_{Ca}(V) = g_{Ca}\, m_\infty(V)\,(V - V_{Ca}), \tag{2a}$$

$$I_K(V,n) = g_K\, n\,(V - V_K), \tag{2b}$$

$$I_S(V,S) = g_S\, S\,(V - V_K), \tag{2c}$$

$$I_{K2}(V) = g_{K2}\, p_\infty(V)\,(V - V_K), \tag{2d}$$

$$\omega_\infty(V) = \left(1 + \exp\frac{V_\omega - V}{\theta_\omega}\right)^{-1}, \quad \omega = m, n, S, \tag{2e}$$

$$p_\infty(V) = \left(\exp\frac{V - V_p}{\theta_p} + \exp\frac{V_p - V}{\theta_p}\right)^{-1}. \tag{2f}$$

The equations correspond to the original system with $g_{K2} = 0$. Modified the system is considered at $g_{K2} = 0.12$. System variables as well the functions and parameters included in the equations have a biological interpretation, a discussion of which can be found in the works [18, 19].

Fig. 1, *a*, *b* and *c* shows various dynamics modes of the modified version systems (1). Fig. 1, *a* and *b* illustrate bistability: depending on the starting point, the trajectory goes either to burst attractor (see Fig. 1, *a*), or to a fixed point (see Fig. 1, *b*) . It can be seen that in the burst attractor mode the variables V and n changes much faster than the variable S. Systems with this type behaviors are called hard [23]. Fig. 1, *c* demonstrates another mode of the system, spike. In this mode, all three variables change with the same time scales. Transition from a burst attractor to a spike attractor occurs when the parameter V_S increases through a bifurcation called blue sky disaster [19, 24]. Note that batch or spike activity of the system, switching between which occurs when changing parameters is called neuromorphic dynamics [25].

Fig. 2 shows the phase portraits of the modified system in three dimensions. Fig. 2, *a* demonstrates the burst attractor It is clearly seen that in this regime the attractor has a characteristic structure of several turns with a loop closing them. The spike attractor is presented in fig. 2, *b*. Since all variables in this mode fluctuate with identical time scales, the attractor has the form of a limit cycle.

The original version of the system also demonstrates burst and spike modes. Visually they are indistinguishable from those shown in Fig. 1 and 2 and are therefore not shown in a separate figure.

To clearly demonstrate changes in the nature of system behavior in depending on the parameters and the choice of initial conditions, we will calculate scalar characteristic quantity

$$Q = \sqrt{\frac{1}{T}\int_{t_0}^{t_0+T} S^2(t)dt}. \tag{3}$$

Here $S(t)$ is a dynamic variable of the system (1), t_0 is some, rather large, time for the system to enter the mode, T is time observations. We use $t_0 = 100$ and $T = 100$. Below, looking at the system connected model neurons (11), calculate value $Q = (Q_1 + Q_2)/2$, where Q_1 and Q_2 are obtained from formula (3) with S replaced by the corresponding value S_1 or S_2.

The meaning of the formula (3) is to compare scalars with invariant ones sets in the phase space of the system in order to be able to compactly and clearly visualize its structure. It doesn't matter how exactly it will be look like the dependence of Q on the parameters. We are interested in regime restructuring, Therefore, it is important that when the nature of the system

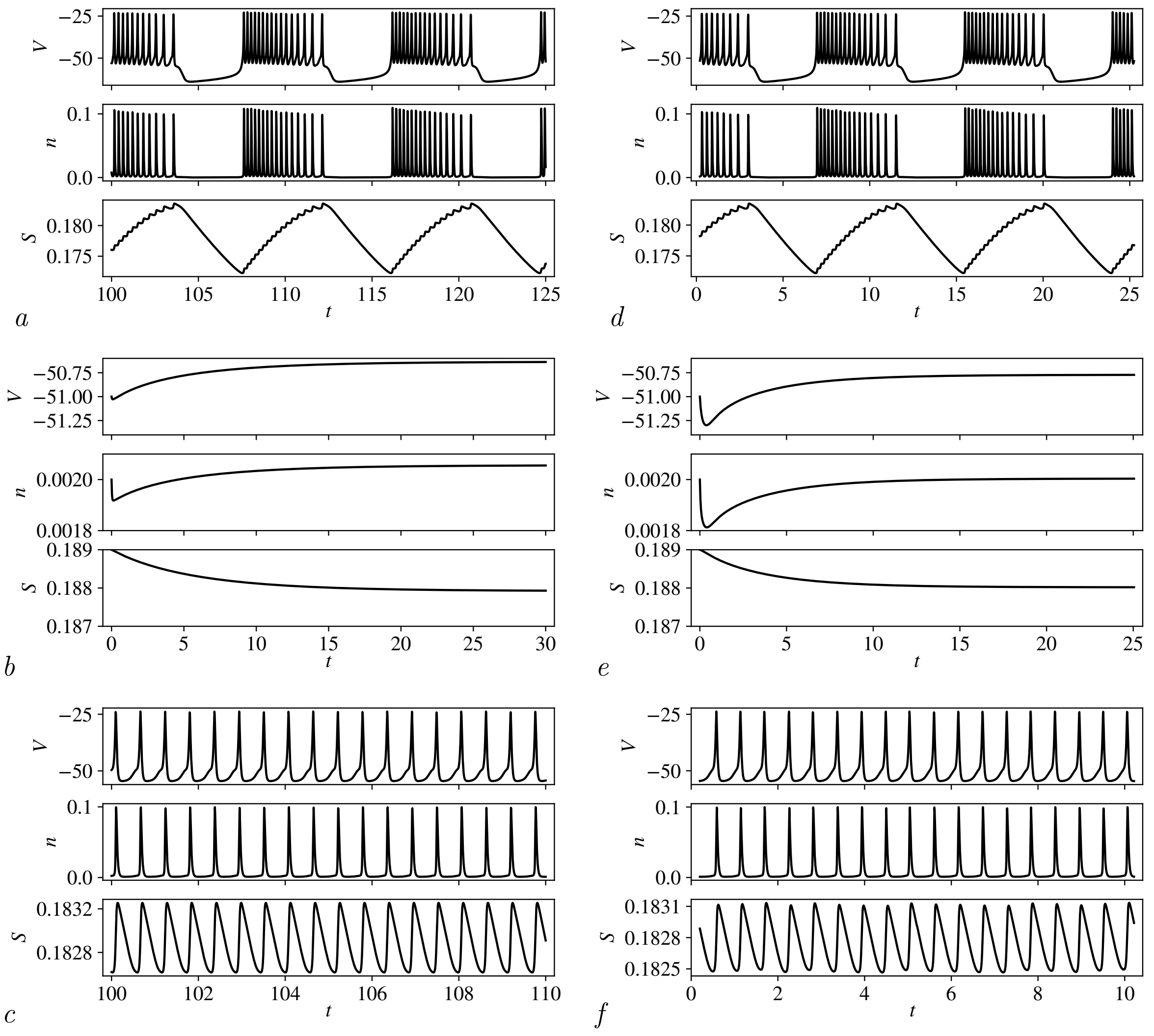


Fig 1. Solutions for modified system (1), i.e. e. at $g_{K2} = 0.12$ (a, b, c), and for the corresponding neural network mapping (5) (d, e, f). Diagrams a, d, b and e illustrate bistability at $V_S = -36$: a, d — bursts when starting from the point $V_0 = -51$, $n_0 = 2 \cdot 10^{-3}$, $S_0 = 0.185$; b, e — approach a fixed point when starting from the point $V_0 = -51$, $n_0 = 2 \cdot 10^{-3}$, $S_0 = 0.189$. Diagrams c, f demonstrate spikes at $V_S = -34$

behavior changes, the value of Q would change quite significantly. Theoretically, different sets can be described by the same Q and be indistinguishable on this basis. However the likelihood of this is quite low. To further reduce this likelihood $S(t)$ is squared so that in the hypothetical case of occurrence negative values, Q would be affected only by their absolute values. Doesn't have it makes no sense to construct more complex formulas for Q, since the representation multidimensional set using a scalar quantity always leads to loss any information and, therefore, cannot be considered exhaustive. Therefore, when using Q, one should correlate its values with information about the nature of the system dynamics obtained from other considerations. In relation to the system under study, we know in advance that in it burst and spike modes may be observed, and in addition there may be stable fixed point. Therefore, calculating Q depending on the parameters, we expect three typical values of this quantity to occur.

In Fig. 3, a and 3, b for the original and modified systems is shown as changing depending on V_S distribution of Q obtained with a large number of system runs with random initial values

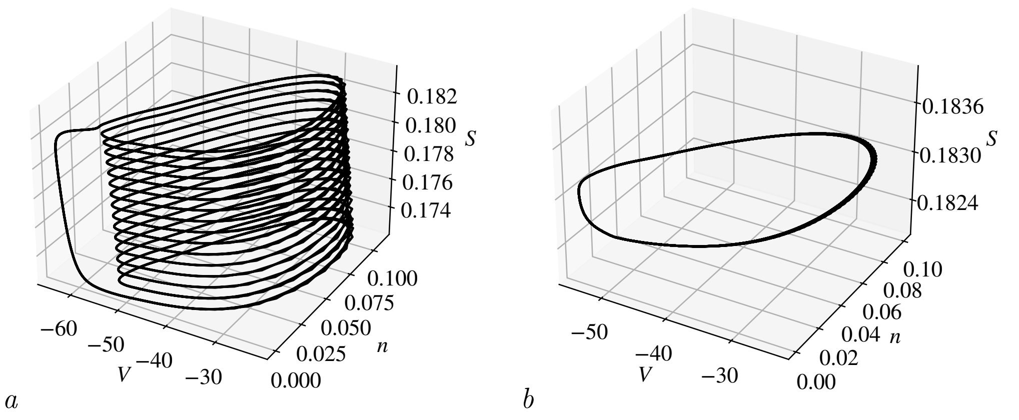


Fig 2. Three-dimensional phase portraits of the modified system (1): *a* — bursts at $V_S = -36$, *b* — spikes at $V_S = -34$

V, n and S. The ensemble of initial values was chosen from scope of the system definition (see table below 3 and section 4) as follows. Range of parameter values V_s was divided into 300 equally spaced points, and at each point 3000 initial values of V, n and S from those indicated in table 3 ranges. Grayscale display in logarithmic scale of the frequency of occurrence of solutions characterized by corresponding values of Q. It can be seen that these values are well grouped and form a system of lines, by the appearance of which one can judge what is happening in the systems bifurcation rearrangements.

In Fig. 3, *a* and 3, *b* the line on the left parts is represented by a burst attractor. The corresponding solution is shown in Fig. 1, *a* and 2, *a*. When driving in side of increasing V_S near the value of $V_S \approx -34$ for the original system (see Fig. 3, *a*) and at $V_S \approx -35$ modified (see Fig. 3, *b*) the value of Q changes jump. This corresponds to the transition from a burst attractor to a spike attractor (see fig. 1, *c* and 2, *b*). In Fig. 3, *b*, which is built for the modified system, there is another line of values Q in the range between $V_S \approx -37$ and $V_S \approx -35$. This line corresponds to a fixed point that is stable in this range of values parameter. The line representing the fixed point is noticeably lighter than the line for oscillatory solution existing at the same

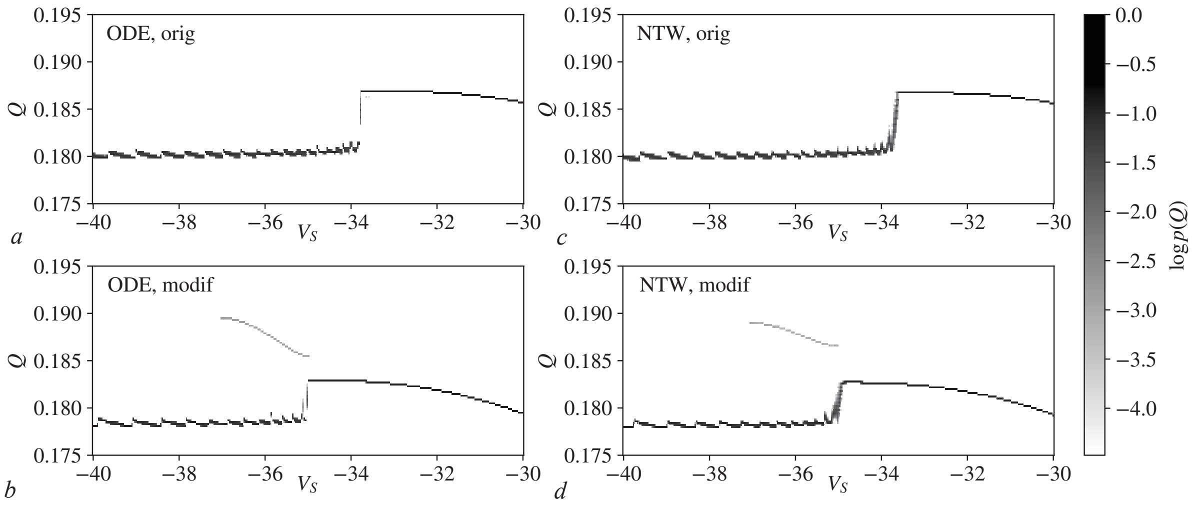


Fig 3. Distributions of Q, see Eq. (3), vs. V_S: *a* and *b* — original and modified systems (1), respectively; *c* and *d* — the neural network mapping (5) trained for these systems

values of V_S. It says that a stable fixed point is small compared to oscillatory solution the size of the basin of attraction, and therefore the probability if randomly selected initial conditions fall into it, it is sufficient small This is discussed in more detail in the work [19].

In Fig. 4, *a* shows how one-dimensional sections of basins of attraction of various invariant sets in phase space of the modified system depending on V_S. For building of this figure, first the fixed point V_f, n_f, S_f of the system is calculated at $V_S = -36$. The initial values of the variables n and S are always set equal $n_0 = n_f$ and $S_0 = S_f$, and the initial value V_0 varies around V_f and these values are plotted vertically in the figure. Parameter V_S varies around the -36 point, and these values are plotted horizontally. For every pair V_S and V_0 are calculated by Q and its values are displayed in the figure using grayscale.

In Fig. 4, *a* three areas are distinguished. Dark rectangle in the center is the area from which the trajectories reach a stable fixed point. The light gray area on the left is the set starting points from which the system reaches the burst attractor. These two regions coexist at the same values of V_S, which corresponds to the regime bistability in the system. The dark gray area on the right shows the points from which the system reaches the spike attractor.

A similar construction for the original system is presented in fig. 5, *a*. There are only two areas here: more the light one on the left side corresponds to the exit to the burst attractor, and the darker one on the right is to the spike one.

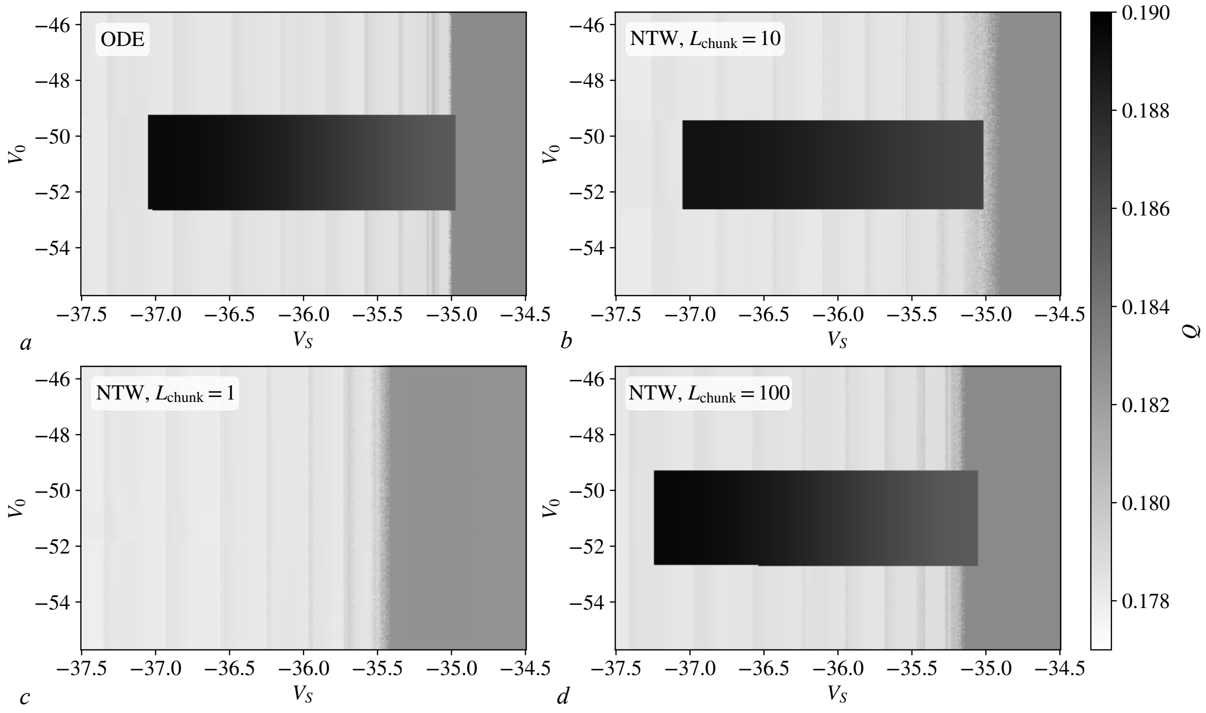


Fig 4. One-dimensional sections of attraction basins of different dynamic regimes: *a* — the modified system (1); *b*, *c* and *d* — the neural network mapping (5) trained on datasets, consisting of trajectory segments of lengths $L_{\text{chunk}} = 10$, $L_{\text{chunk}} = 1$ and $L_{\text{chunk}} = 100$, respectively. Diagrams are constructed in the vicinity of the point $V_f = -50.6357$, $n_f = 2.05598 \times 10^{-3}$, $S_f = 0.187922$ that is a stable fixed point of (1) at $V_S = -36$. The initial value of V_0, vertical axis, is varied within the range $V_f \pm 0.1V_f$, and starting values for variables n_0 and S_0 are taken equal to n_f and S_f, respectively. For each trajectory Q is calculated and its values are represented via gray scale

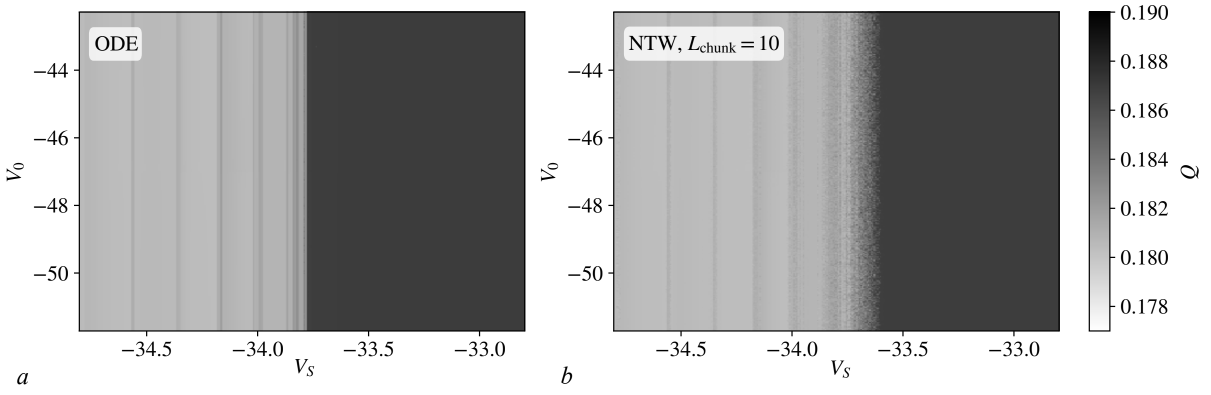


Fig 5. One-dimensional sections of basins of attraction: *a* — original system (1); *b* — the neural network mapping trained on a dataset with $L_{\text{chunk}} = 10$. The diagrams are built in the vicinity of the point $V_f = -46.9978$, $n_f = 3.92943 \times 10^{-3}$, $S_f = 0.210855$. This is an unstable fixed point of the original system at $V_S = -33.8$

2. Related systems

Let us now consider two systems of the form (1) and introduce a connection between them, as this is discussed in [26]. The equation for subsystem 1 has view:

$$\begin{aligned}
\tau\dot{V}_1 &= -I_{Ca}(V_1) - I_K(V_1, n_1) - I_{K2}(V_1) - I_S(V_1, S_1) + g_{c,V}(V_1 - V_2),\\
\tau\dot{n}_1 &= \sigma\left[n_\infty(V_1) - n_1\right],\\
\tau_S\dot{S}_1 &= S_{\infty,1}(V_1) - S_1.
\end{aligned} \tag{4}$$

The equations for the second system are obtained by replacing the indices 1 ↔2. The parameter $g_{c,V}$ is responsible for the strength of the connection. Note that in in equations (4) the function $S_{\infty,1}$ is provided with the index “1”. This is done because it includes the V_S parameter, see (2e). We will consider subsystems 1 and 2 with the same parameters with the exception of V_S, which will be set differently for each of subsystems as $V_{S,1}$ and $V_{S,2}$, respectively.

An idea of the general picture of the behavior of coupled systems is given by fig. 6, *a*–*d*, fig. 7, *a*–*c* and Fig. 8, *a*–*c*. In Fig. 6, similar to how it was done in Fig. 3, halftones are shown distribution of the characteristic quantity $Q = (Q_1 + Q_2)/2$, where Q_1 and Q_2 calculated using the formula (3) for subsystems 1 and 2, respectively. The values of the parameter $V_{S,1}$ are plotted horizontally, and $V_{S,2}$ is equal to $V_{S,1} + 0.1$. Modes of behavior of two related originals systems largely repeat the modes of a single system (compare fig. 6, *a* and 3, *a*). On the left side diagrams, that is, for smaller values of $V_{S,1}$ and $V_{S,2}$, burst mode. This illustrates fig. 7, *a*. On the right, for greater $V_{S,1}$ and $V_{S,2}$ a spike mode occurs (Fig. 7, *c*). In contrast of single system is manifested in how the transition from one mode to to another. Instead of a strictly defined bifurcation point, coupled systems a transition region arises in which burst oscillations with irregular length. As the parameter increases, the average duration of bursts The burst increases until it turns into a spike mode. The transient behavior is illustrated in Fig. 7, *b*.

For a pair of modified systems, the scenario of mode changes when moving along parameters $V_{S,1}$ and $V_{S,2}$ are generally the same: from bursts to spikes through an extended transition region. As with a single modified system, a pair of linked Such systems have a region of bistability, in which with the oscillatory solution a stable fixed point coexists (compare Fig. 3, *b* и 6, *b*).

The dynamics of the pair are original. The modified system is characterized by broadening areas of transition from bursts to spikes (see Fig. 6, *c*), as well as to the emergence of more complex transition regimes. As an illustration in Fig. 8, *a* shows oscillations when bursts occur irregular lengths are different for different subsystems, and Fig. 8, *b* demonstrates the coexistence of burst and spike modes for the first and second subsystems.

By increasing the strength of the connection between the original and modified systems the region of stability of the fixed point is significantly reduced, and the transition the area becomes even wider (see Fig. 6, *d*). On Fig. 8, *c* shows that immediately after the transition region in In such a system, an irregular spike regime occurs.

In Fig. 9, *a*–*c* shows one-dimensional sections of basins of attraction of various modes associated original and modified systems with increasing bond strength $g_{c,V}$. This the figure is constructed similarly to Fig. 4 and 5. Only now the initial values change simultaneously variable V for both the first and second subsystems. Black area in the central part represents the starting values, from which the system goes to fixed point. Light gray areas around are points from which the system goes into oscillatory mode. It can be seen that with increasing coupling strength $g_{c,V}$ the region the attraction of the fixed point decreases.

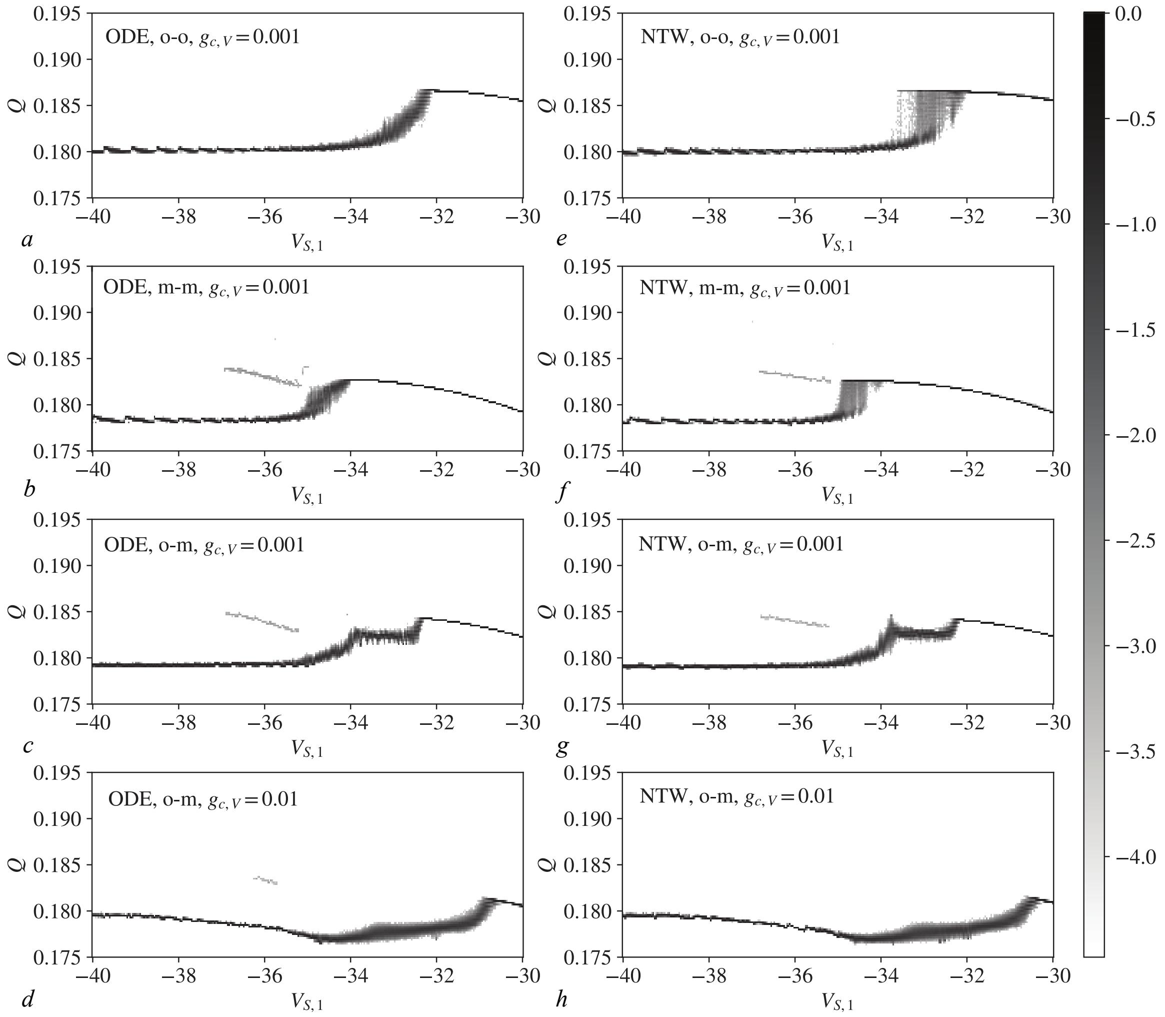


Fig 6. Distributions of $Q = (Q_1 + Q_2)/2$ vs. $V_{S,1}$ and $V_{S,2} = V_{S,1} + 0.1$: *a*–*d* — Eqs. (4); *e*–*h* — the neural network mapping (11). The following combinations of subsystems are represented: *a* and *e* — two original systems ($g_{K2} = 0$); *b* and *f* — two modified systems ($g_{K2} = 0.12$); *c*, *d*, *g* and *h* — original and modified systems. Coupling parameter values: *a*–*c* and *e*–*g* — $g_{c,V} = 0.001$; *d* and *h* — $g_{c,V} = 0.01$

3. Neural network mapping

We are interested in constructing a display of the form $u(t + \Delta t) = F(u(t), p, w)$, capable of values of w reproduce the behavior of various dynamic systems. Here $u(t)$ is vector of dynamic variables, p is vector of control parameters, Δt is time step, which we will set as fixed size. Mathematically proven, see works [5–10], which is two-layer a fully connected neural network can be used to approximate arbitrary functions of many variables. It follows that it must be is suitable for constructing the above-mentioned universal mapping. This the question is studied in the work [16]. It was shown quite good quality of dynamics reproduction for various systems: Lorenz, Rössler, Hindmarsh-Rose neuron model. However, it was discovered that neural network mapping based on a simple two-layer architecture is quite difficult to train for rigid type systems in which the variables have very different time scales. Therefore, in the work [17] a more complex architecture. Instead of one network receiving vector $u(t)$ as input and returning the vector $u(t+\Delta t)$, for each dynamic variable now a separate subnetwork is created. Also a two-layer fully connected one. At the same time, everything other variables and values of control parameters are entered after passing through one more, additional fully connected layer. Such a network was able to not only simulate the dynamics of a rigid system, but also correctly reproduce behavior, samples of which were not presented to her during training.

In this work we will consider a neural network mapping from works [17] with minor modification:

$$u_i(t + \Delta t) = (1 - \chi)u_i(t) + \chi\Big(f\Big(u_i(t)a_i + \mu_i + g\big(u_{\neg i}(t)A_i + pB_i + \beta_i\big)\Big)b_i + \gamma_i\Big). \qquad (5)$$

The modification boils down to the introduction of a stabilizing factor $\chi = 0.001$. This eliminates situations where, as a result, "unsuccessful" initialization of weight coefficients with random numbers trained the display showed divergence instead of the expected behavior. Here $u(t)$ is

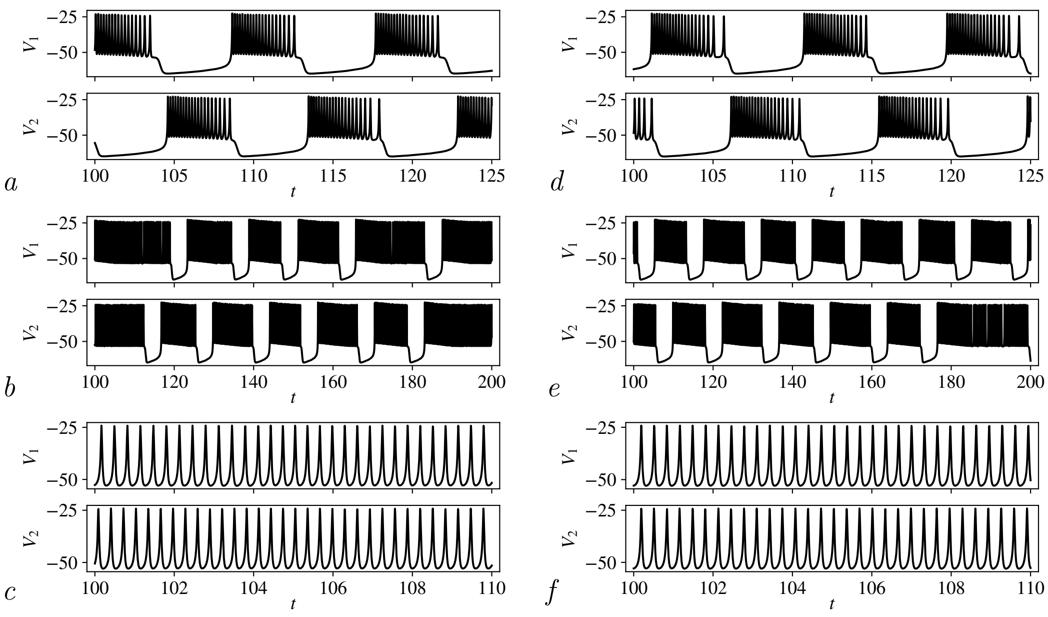


Fig 7. Solutions for the coupled original systems: a–c — Eqs. (4); d–f — corresponding coupled neural network mappings (11). Represented regimes: a and d — bursts with $V_{S,1} = -36$; b and e — irregular bursts with $V_{S,1} = -33$; c and f — spikes at $V_{S,1} = -31$. In all cases $V_{S,2} = V_{S,1} + 0.1$. The coupling parameter is $g_{c,V} = 0.001$

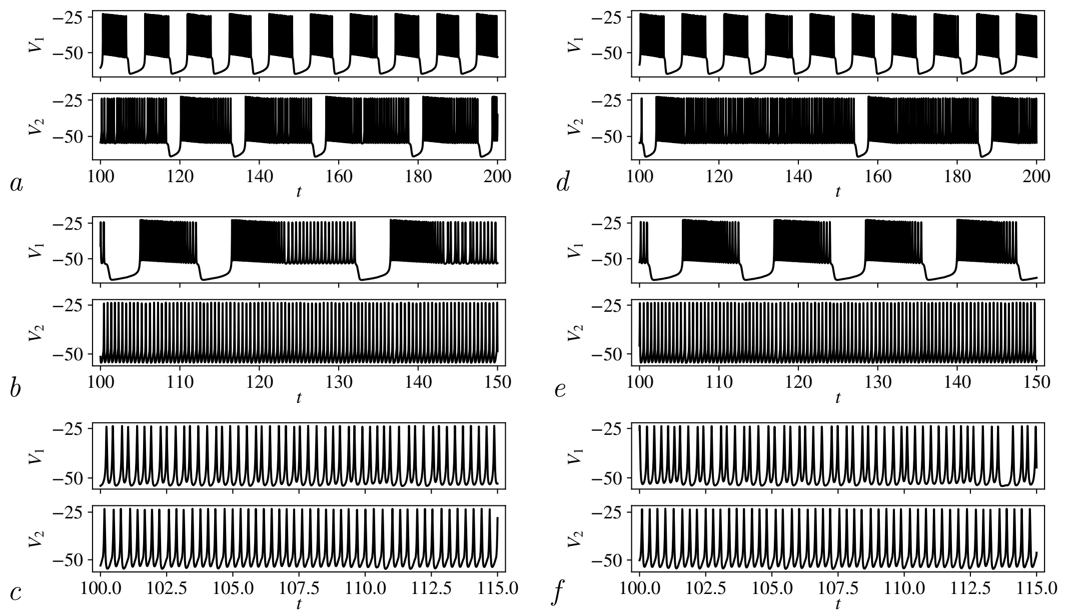


Fig 8. Solution for the coupled original and modified systems: *a*–*c* — Eqs. (4); *d*–*f* — coupled neural network mappings (11). Represented regimes: *a* and *d* — irregular bursts of different lengths when $V_S = -34.4$; *b* and *e* — bursts and spikes at $V_S = -33.3$; *c* and *f* — irregular spikes at $V_S = -30.5$. In all cases $V_{S,2} = V_{S,1} + 0.1$. Coupling parameter: *a*, *b*, *d* and *e* — $g_{c,V} = 0.001$; *c* and *f* — $g_{c,V} = 0.01$

Table 2. Description of the elements of Eq. (5)

$u(t)$	row vector	$D_u = 3$
$u_i(t)$	scalar	
$u_{\neg i}(t)$	row vector	$D_u - 1$
p	row vector	$D_p = 1$
a_i, μ_i и β_i	row vectors	$N_h = 100$
b_i	column vector	N_h
γ_i	scalar	
A_i	matrix	$(D_u - 1) \times N_h$
B_i	matrix	$D_p \times N_h$
χ	constant	0.001
$f(\cdot), g(\cdot)$	scalar functions	$\tanh(\cdot)$

vector of dynamic system variables of size D_u, $u_i(t)$ is the ith variable, and $u_{\neg i}(t)$ is a vector obtained from $u(t)$ by removing ith variable. In our case, the vector $u(t)$ contains components V, n and S. As is customary for neural networks, we will assume that $u(t)$ and $u_{\neg i}(t)$ these are row vectors. The symbol p denotes a row vector control parameters of the D_p dimension system. We are considering change only one of the system parameters, namely V_S. Therefore he one-dimensional. A description of other elements of the formula (5) is given in tab. 2. Functions $f(\cdot)$ and $g(\cdot)$ in terms of neural networks are called activation functions. These are scalar functions of scalars arguments. It is assumed that when they are applied to vectors they act element by element.

The structure of the formula (5) corresponds to the structure of a two-layer fully connected network. The first layer receives the scalar value i-th as input dynamic variable u_i, and its output is a vector of dimension N_h (see table 2), obtained after calculating the expression $f(u_i(t)a_i + \mu_i +$

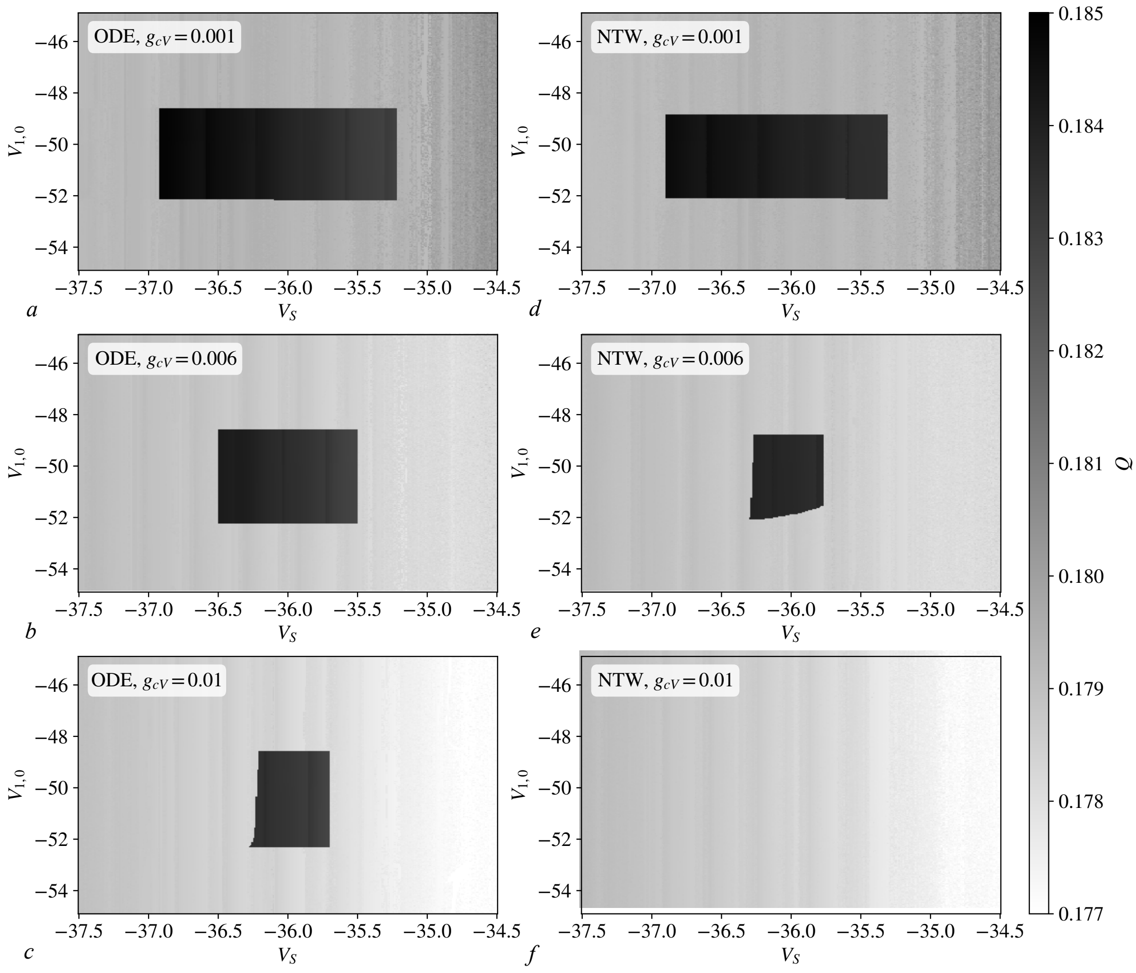


Fig 9. One-dimensional sections of attraction basins for coupled original and modified system: *a*, *b*, *c* — Eqs. (4); *d*, *e*, *f* — coupled neural network mappings (11). Coupling parameter $g_{c,V}$: 0.001 (*a*, *d*); 0.006 (*b*, *e*); 0.01 (*c*, *f*). The diagrams are obtained in the neighborhood of a fixed point $V_{1,f} = -49.8965$, $n_{1,f} = 2.34541 \times 10^{-3}$, $S_{1,f} = 0.199464$, $V_{2,f} = -50.5546$, $n_{2,f} = 2.08592 \times 10^{-3}$, $S_{2,f} = 0.187634$ calculated at $V_S = -36$ and $g_{c,V} = 0.001$. Initial values of $V_{1,0}$ plotted vertically vary within the range $V_{1,f} \pm 0.1V_{1,f}$, the initial value of $V_{2,0}$ also varies: $V_{2,0} = V_{1,0} - V_{1,f} + V_{2,f}$, and initial values of other variables are chosen equal to the values given above

$+\ g(\ldots))$. This layer is called hidden. However, in Unlike a simple two-layer network, here the displacement vector (bias) μ_i adjusted by values supplied through an additional layer $g(u_{\neg i}(t)A_i + pB_i + \beta_i)$, which depends on other dynamic variables and parameters. The second, output layer of the network has the form: $f(\ldots)b_i + \gamma_i$. This layer returns a scalar value, which is then used to calculate the value of the dynamic variable u_i at the new step time as $(1-\chi)u_i + \chi(\ldots)$. Thus, the neural network the mapping (5) is constructed as a neural network that adapts to the required system by learning weight parameters, presented in the form of the following matrices and vectors:

$$w = \{a_i, \mu_i, A_i, B_i, \beta_i, b_i, \gamma_i \mid i = 1, 2, 3\}. \tag{6}$$

When actually implementing a neural network mapping (5) it is necessary to take into account that all modern computing tools for work with neural networks, as well as relevant theoretical developments, imply standardization of the data set on which the education. This means that all quantities supplied to the network input must fall into a unit interval near zero. Therefore, the network will learn and function on data rescaled according to formulas (elementwise operations are assumed)

$$u \to (u - m_u)/s_u,\ p \to (p - m_p)/s_p. \tag{7}$$

Here m_u, m_p, s_u and s_p are, respectively, vectors of average values and standard deviations in the domain of definition of the neural network display. The calculation of these vectors is discussed in section 4, and their values are given in table 3. Thus, the original data supplied to the display input (5) is rescaled according to (7), then the required number of iterations is performed, and for When presenting the results, the obtained values are rescaled back.

When implementing software mapping in the form of a neural network, we use the following operators. The $\mathrm{Take}(u, i)$ operator returns the ith element of the vector u, and the operator $\mathrm{Sans}(u, i)$, on the contrary, removes the ith element and returns vector $u_{\neg i}$ without this element. Operator $\mathrm{Dense}(x, W, b) = xW + b$ represents a fully connected layer without an activation function: multiplying the input vector x by matrix W and shift by vector b. Finally, the square brackets $[\cdot, \cdot]$ denotes the concatenation of two vectors or matrices.

Using these notations, the action of a neural network mapping can be illustrated using the diagram in Fig. 10 and corresponding formulas (8), which show how calculations for the i-th variable are organized. The neural network has two inputs vectors, $p = (V_S)$ and $u = (V(t), n(t), S(t))$, formula (8a). At first u is split into a scalar u_i and a vector $u_{\neg i}$, in which the ith element missing, formula (8b). Layer (8c) accepts $u_{\neg i}$ and p and calculates a vector h_i of dimension N_h, which includes information about the influence of control parameters and all variables for with the exception of ith. This vector is added to the output of the processing layer ith component, formula (8d). The result is a vector hidden layer q_i of dimension N_h. It is transmitted on the weekend layer (8e), after which it is used directly for calculating the solution at a new time step $v'_i \equiv u(t + \Delta t)$. Repeating these calculations for $i = 1, 2, 3$ we get the solution vector generated by the network $u'(t + \Delta t)$.

$$p = \mathrm{Input}(), \quad u = \mathrm{Input}(), \tag{8a}$$

$$u_i = \mathrm{Take}(u, i), \quad u_{\neg i} = \mathrm{Sans}(u, i), \tag{8b}$$

$$h_i = g(\mathrm{Dense}([u_{\neg i}, p], [A_i, B_i], \beta_i)), \tag{8c}$$

$$q_i = f(\mathrm{Dense}(u_i, a_i, \mu_i) + h_i), \tag{8d}$$

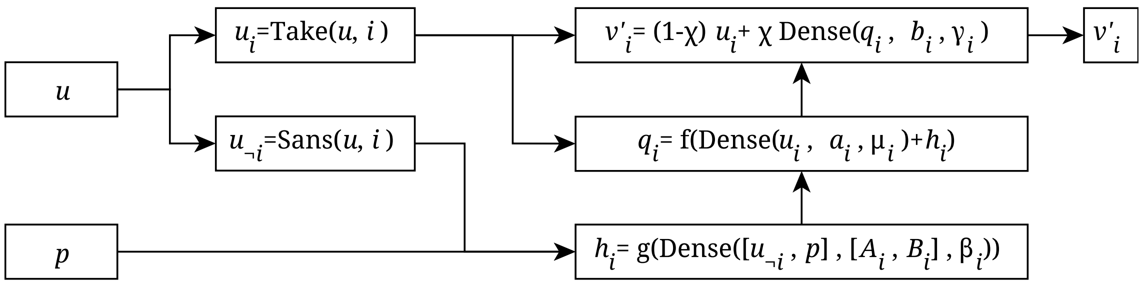
$$v'_i = (1 - \chi)u_i + \chi\,\mathrm{Dense}(q_i, b_i, \gamma_i). \tag{8e}$$

When the network has already been trained and is used to calculate the trajectory of the system, you need put $u'(t + \Delta t) = u(t + \Delta t)$. During the training process, weights network coefficients (6) are adjusted so that difference between the generated result $u'(t + \Delta t)$ and the required $u(t + \Delta t)$ was minimal. Therefore, it is natural to choose the square as the objective function norms of the difference between these vectors:

$$L = \|v - v'\|^2. \tag{9}$$

Table 3. Datasets parameters

$V_S \in [-40, -30]$, $V \in [-70, -18]$, $n \in [0.0, 0.13]$, $S \in [0.14, 0.26]$,
$m_u = (-44, 0.065, 0.2)$, $m_p = -35$, $s_u = (26, 0.065, 0.06)$, $s_p = 5$,
$L_{\text{chunk}} = 10$, $N_{\text{chunks}} = 10^5$, $\Delta t = 0.005$,
$L_{\text{train}} = L_{\text{chunk}} \times N_{\text{chunks}} = 10^6$, $L_{\text{validation}} = 10^5$, $L_{\text{batch}} = 10^4$



u
p
u_i=Take(u, i)
$u_{\neg i}$=Sans(u, i)
v'_i= (1-χ) u_i+ χ Dense(q_i, b_i, γ_i)
q_i= f(Dense(u_i, a_i, μ_i)+h_i)
h_i= g(Dense([$u_{\neg i}$, p], [A_i, B_i], β_i))
v'_i

Fig 10. The structure of the neural network that corresponds to the mapping (5) for ith variable. Here $u \equiv u(t)$, $v'_i \equiv u'_i(t + \Delta t)$. The prime denotes a value computed by the network. This value in the course of training is compared with the “correct” value $u_i(t + \Delta t)$ from the training dataset

The training consists of using the gradient descent method, step by step calculate corrections to the weighting coefficients (6), leading to decrease L:

$$w \to w - \varkappa \nabla_w L, \tag{10}$$

where $\varkappa$ is usually a small parameter that regulates the value step. In fact, when teaching, we use a modification of the method gradient descent — method Adam [27]. This is most common currently used algorithm for optimizing network weights. Its advantage is that the value of $\varkappa$ is automatically adjusted during iterations, that provides a high speed of their convergence.

4. Training dataset and network training

We are creating a neural network that is somewhat similar to the so-called recurrent network [11]: the vector $u(t)$ is supplied to the input, and the output is $u(t + \Delta t)$ is formed, which is again fed to the input of the same network for calculation of the next value. However, there are important differences. Firstly, the internal structure of the network is selected so that it is possible to describe it in form of mapping, explicit and sufficiently compact, allowing further theoretical analysis, see formula (5). Modern recurrent cells such as LSTM and GRU are designed significantly more difficult [11]. It doesn’t make much sense for them to try to write out appropriate display due to its bulkiness. Secondly, the procedure training of our network will be organized differently. Traditional recurrent networks are trained on fixed sequences, usually not very large length. This assumes that they will function on sequences the same length. The neural network mapping (5) must be capable of generating trajectories of arbitrary length. Therefore, in the learning process we do not use recursion. Pairs will be used as training data $(u(t), u(t+\Delta t))$, where the first value is supplied to the input, the network makes only one iteration and the calculated result $u'(t + \Delta t)$ is compared with expected $u(t + \Delta t)$.

To form a data set, we set the definition area in this way: so that all invariant objects of phase space appear in it, attractors of oscillatory solutions and a fixed point. To do this, we first set range of variation of the V_S parameter, see table 3. Let us remind you that we will change the values of only this one parameter, the values of the rest are given in table 1. Changing V_S, we will construct numerical solutions to the system (1), repeatedly starting from randomly selected starting points and defining the minimum and maximum values that accept V, n and S. Additionally, we will calculate the position of the stationary points at different V_S and adjust the found ranges of change variables to include her in them. We will then expand the found ranges approximately 10% from each edge and round to two significant figures. Where in the lower bound for n goes a little into the negative area, replace it zero. The resulting ranges for

V, n and S along with the range for V_S, see table 3, set the scope of definition of the neural network mapping when it models the system (1). Note that in work [17], where the main goal is modeling of a single systems, training was carried out on fragments of the oscillatory solution systems. Since this work explores the possibility of modeling connected systems (4), the network is trained over the entire definition domain.

For the resulting domain of definition, the average values of m_u and m_p are calculated, and also the corresponding standard deviations s_u and s_p, see table 3. The data supplied to the network input, that is, to display (5), pre-scaled by formulas (7) so that in fact the values of all variables and parameters range from -1 to 1. After calculating the trajectory data, generated by the network are subject to inverse transformation for subsequent analysis.

The training data set is created as follows. Randomly selected value parameter V_S and initial values of dynamic variables $u(0) = (V, n, S)$ from domain of definition, the boundaries of which are given in table 3. Then the solution to the equations (1) is calculated in $L_{\text{chunk}} + 1$ steps with time interval Δt, see table 3. From the received sequences are constructed by L_{chunk} records of the form (V_S, $u(0)$, $u(\Delta t)$), (V_S, $u(\Delta t)$, $u(2\Delta t)$), (V_S, $u(2\Delta t)$, $u(3\Delta t)$) and so on, which are placed in the array training data. In general, for the training data set it is generated N_{chunks} segments of trajectories so that the total length of the training data set is equal $L_{\text{train}} = L_{\text{chunk}} \times N_{\text{chunks}}$. Before the beginning training records in the training set are mixed, so that the network "sees" in Each moment of learning is individual points of trajectories, and not entire fragments.

To control the quality of training, a validation data set of length $L_{\text{validation}}$. For validation, trajectory segments of length one step. This means that for randomly generated V_S and $u(0)$ it is calculated $u(\Delta t)$, entry (V_S, $u(0)$, $u(\Delta t)$) is placed in the array validation data and then a new record is generated.

The standard way to save memory during the learning process is to apply to the input network data in portions, which in English-language literature are called "batches", which is usually translated into Russian as "packages". For our network, packet size is equal to L_{batch}, see table 3. Networks first The training data set is presented, packet by packet. For each package the gradient descent step (10) is performed and adjusted network weights w. When all training set packets have been processed, it is said that one era of learning has passed. After this, the network is presented with validation data, for which the value of the objective function is calculated. Gradient descent and correction no weighting coefficients are applied to the validation data. Curves dependence of the objective function on the epoch number for training and validation data are called learning curves.

The result of training a neural network significantly depends on the length fragments

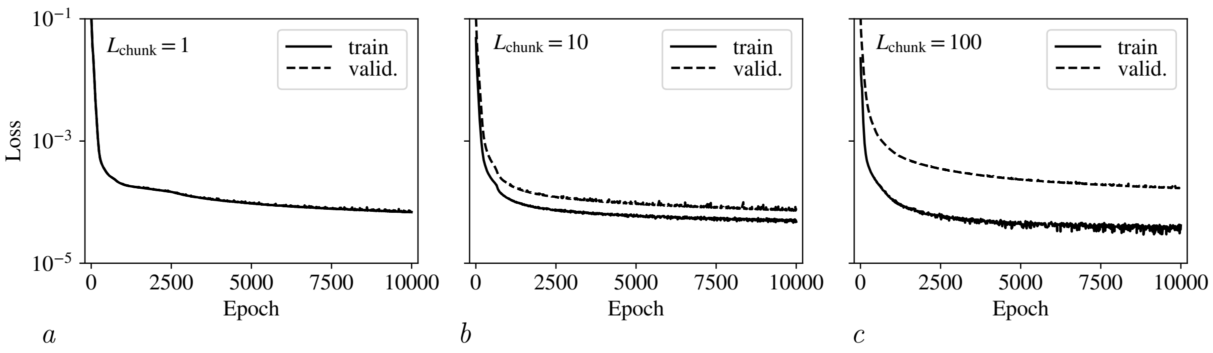


Fig 11. Learning curves of the neural network mappings (5) for modified system: *a*, *b* and *c* — lengths of the training trajectory chunks L_{chunk} are 1, 10 and 100, respectively. Both training and validation curves are sgown. On the diagram *a* the curves coincide

of trajectories L_{chunk}. In Fig. 11 learning curves on the modified system data set are shown. Vertically the value of the objective function is postponed (9). Fig. 11, *a*, *b*, *c* obtained for sets data with different lengths of trajectory segments, $L_{\text{chunk}} = 1$, $L_{\text{chunk}} = 10$, $L_{\text{chunk}} = 100$, respectively. It is clear that what higher L_{chunk}, the more the training and validation ones diverge curves: the target function on the training data remains at the same level, and validation gets worse.

Since, unlike the training data, the validation data has an adjustment network weighting coefficients w are not produced, that is, they are not produced adapting the network directly to this data, comparing the values of the objective function on training and validation data allows us to judge how well the network has learned to find generalized features, and therefore how well it will function upon completion of training. Strong discrepancy between validation and training curves indicate overfitting. It means that the network mostly just remembers the training data. An example of this behavior is shown in Fig. 11, *c* — when fragments trajectories are long, the network more "willingly" remembers them, whereas on at arbitrary points of phase space its behavior differs noticeably from behavior of the simulated system.

On the other hand, decreasing L_{chunk} also leads to negative effect. Figures 4, *b*, *c* and *d* show one-dimensional sections of basins of attraction of different display modes (5) trained with $L_{\text{chunk}} = 10$, $L_{\text{chunk}} = 1$, $L_{\text{chunk}} = 100$, respectively. Comparing these diagrams from Fig. 4, *a*, which is built directly for numerical solutions of the system (1), we see that the network trained on one-step trajectory segments, $L_{\text{chunk}} = 1$, does not "see" at all stable fixed point. Absent in Fig. 4, *c* the characteristic dark rectangle in the center that is present on all others diagrams. Retrained network at $L_{\text{chunk}} = 100$ is also bad reproduces the features of the dynamics of the system (1). It is clear that the area stability of a fixed point (width of the dark central rectangle), and also, the transition point from the burst mode to the spike mode (the border between the light gray and dark gray areas) differs from the correct values by fig. 4, *a*. Thus, it is clear that the optimal value for training neural network mapping (5) for reproduction modified system L_{chunk}=10.

For the original system, learning curves are not shown, since they are visual indistinguishable from the corresponding curves for the modified system on fig. 11. One-dimensional cross-section of basins of attraction in depending on the V_S parameter for the neural network display (5) trained with $L_{\text{chunk}} = 10$ is shown in Fig. 5, *b*. It can be seen that it reproduces well the diagram for modeled system (1).

Thus, to simulate the dynamics of the original and modified versions (1) systems are best suited for training data sets, containing trajectory segments of length $L_{\text{chunk}} = 10$. In the future we We consider neural network mappings trained on such data sets.

5. Dynamics of neural network mapping

Neural network mapping (5), trained as described above, reproduces well the dynamics of the original and modified versions systems (1). Figures 1, *d*, *e* and *f* show the time dependences for the modified system, generated by neural network mapping. There is a very good agreement with Fig. 1, *a*, *b* and *c*, obtained as numerical solutions (1). Similar ones not shown here pairs of solutions for the original system, since they also correspond well each other and are visually indistinguishable from those shown in Fig. 1.

Changes in the nature of dynamics depending on the V_S parameter are demonstrated fig. 3. It can be seen that the neural network mapping reproduces well bifurcation restructuring from burst to spike attractors as in original (compare Fig. 3, *a* and *c*), and in modified (compare fig. 3, *b* and *d*) systems. In addition, for the modified system, the neural network mapping correctly repeats the bistability region in which the burst attractor coexists with a fixed point.

As mentioned above, in Fig. 4, *a* and *b* it is shown that neural network mapping reproduces with a high degree of accuracy structure of basins of attraction of various modes of the system. Similar the correspondence of the attraction pools for the original system is shown in fig. 5.

Let us now discuss coupled systems. We are going to demonstrate that if between neural network mappings trained for single systems, enter connection, they will reproduce the dynamics of coupled systems (4).

Since in the equations (4) the connection is introduced through the components V_1 and V_2, when moving to mappings you need to modify the corresponding parts neural network mapping (5), adding to them the one responsible for connection term proportional to $g_{c,V}$:

$$V_1(t+\Delta t) = F_0^{(1)}(V_1(t), n_1(t), S_1(t)) + \frac{\Delta t\, g_{c,V}}{\tau}(V_1(t) - V_2(t)). \qquad (11)$$

Here $F_0^{(1)}$ denotes the right side of the neural network mapping for zero components of the first subsystem, that is, for V_1. The mapping for V_2 is by replacing the indices $1 \leftrightarrow 2$. All other components are calculated as and earlier, using the appropriate neural network mapping like (5).

Thus, to simulate the dynamics of coupled systems (4) we we will use neural network mappings trained for single systems, like described in sections 3 and 4, introducing a relationship between them like (11).

Fig. 6 compares the transformations of the dynamics modes of coupled oscillators (4) with the corresponding neural network model (11). The left column is obtained based on numerical solutions to equations (4), and the column on the right is for iterations associated neural network mappings. Pairs matched original (see fig. 6, *a* and *e*), modified (see fig. 6, *b* and *f*), and the original is a modified subsystem (see Fig. 6, *c*, *d*, *g* and *h*). For the latter, cases of two different values are shown bond strength $g_{c,V}$. A very good correspondence between the drawings can be seen. Character mode rearrangements, demonstrated by neural network mappings, corresponds to rearrangements in the system specified by the equations (4). it's the same illustrated by Fig. 7 and 8, which show examples of solutions for different values of V_S. In these pictures the right columns are obtained as numerical solutions (4), and on the right are iterations of related neural network mappings. We see very good compliance across all cases.

Let us now discuss the situation when the behavior of coupled neural network mappings corresponds to the behavior of the system (4) insufficiently accurately. On Fig. 6, *c* and *d*, which are built for the pair original - modified subsystem with $g_{c,V} = 0.001$ and $g_{c,V} = 0.01$, in the region of $V_S = -36$, along with an oscillatory solution there is range of stability of a fixed point. Note the low saturation of the image this solution, which indicates a low probability of the system reaching it when starting from random starting points. Also note that the width by V_S range of existence of a stable fixed point decreases with increasing $g_{c,V}$. The corresponding neural network mapping reproduces very well stability region of a fixed point at $g_{c,V} = 0.001$ (compare fig. 6, *g* and *c*). However, with By increasing $g_{c,V}$ to 0.01, it disappears for the neural network mapping (compare Fig. 6, *h* and *d*). This situation analyzed in Fig. 9. Shown here are one-dimensional cross-sections pools of attraction of different modes depending on V_S for a pair original–modified system for different force values communications. Fig. 9, *a*–*c* are constructed for solutions of coupled equations (4), and Fig. 9, *в*–*f* — for associated neural network mappings. The black central area in these pictures represents the values of the variables, starting from which the system goes to fixed point. With weak coupling, these regions are almost identical for equations and for neural network mappings. However, as $g_{c,V}$ increases modification of the region occurs in different ways: in both cases it narrows along V_S, but for displays this is faster. The difference is clearly visible when $g_{c,V} = 0.006$ in Fig. 9, *b* and *e*: black central area in the picture on the right much narrower than on the left. Further, at $g_{c,V} = 0.01$ the region of exit to the fixed point for equations is still preserved, and for neural network mappings is no longer available. This suggests that with increasing coupling strength in the system neural network mappings, a fixed point completely loses stability earlier than in the simulated equations.

Thus, in general, we can conclude that the associated neural network mappings, trained separately, generally reproduce well the behavior of related systems For a fixed point there are minor differences in the boundaries areas of its stability observed when the bond strength changes.

Conclusion

We analyzed the procedure for constructing and training a neural network mapping, capable of modeling the dynamics of nonlinear systems. Considered in detail the architecture of such a mapping, the method of generating a data set and the learning process itself. The modeled system was considered neuron defined by equations based on formalism Hodgkin-Huxley [18]. Distinctive feature of dynamics neuron is rigidity, that is, the presence of very different temporal the scale of change in variables. This is an additional factor complicating the construction of an appropriate neural network mapping and requires using a special neural network architecture. Each variable should be modeled by a separate subnet [17].

The main goal of this work was to show that the proposed neural network display can be used as part of linked systems. Linked neural network mappings trained for single systems can without additional training to reproduce the dynamics of relevant related neurons. The example of connected neurons shows a good correspondence between the dynamics original equations and associated neural network mappings. Restructuring the nature of behavior reproduced well, when changing a parameter. In particular, the appearance and disappearance of bistability is reproduced.

The approach developed in the work is formulated in the most general way form. You can try to build the considered neural network mapping for any dynamic systems and introduce arbitrary connections between them. Wherein the question of the limits of its applicability has not yet been worked out. In particular, it is not clear Is it possible to indicate practically interesting, not "exotic" examples dynamic systems for which neural network mapping training impossible. Also of interest is the question of whether neural networks are capable of displays without additional training are good at reproducing the dynamics of more than two connected systems with different, including nonlinear, connections. All these questions require further study.

References

1. Levin E, Gewirtzman R, Inbar GF. Neural network architecture for adaptive system modeling and control. Neural Networks. 1991;4(2):185–191. DOI: 10.1016/0893-6080(91)90003-N.
2. Grieger B, Latif M. Reconstruction of the El Niño attractor with neural networks. Climate Dynamics. 1994;10(6–7):267–276. DOI: 10.1007/BF00228027.
3. Zimmermann HG, Neuneier R. Combining state space reconstruction and forecasting by neural networks. In: Bol G, Nakhaeizadeh G, Vollmer KH, editors. Datamining und Computational Finance. Vol. 174 of Wirtschaftswissenschaftliche Beiträge. Heidelberg: Physica; 2000. P. 259–267. DOI: 10.1007/978-3-642-57656-0_13.
4. Gilpin W, Huang Y, Forger DB. Learning dynamics from large biological data sets: Machine learning meets systems biology. Current Opinion in Systems Biology. 2020;22:1–7. DOI: 10.1016/j.coisb.2020.07.009.
5. Kolmogorov AN. On the representation of continuous functions of several variables by

superpositions of continuous functions of a smaller number of variables. Amer. Math. Soc. Transl. 1961;17:369–373.

6. Arnold VI. On functions of three variables. Amer. Math. Soc. Transl. 1963;28:51–54.
7. Kolmogorov AN. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. Amer. Math. Soc. Transl. 1963;28:55–59.
8. Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems. 1989;2(4):303–314. DOI: 10.1007/BF02551274.
9. Gorban AN. Generalized approximation theorem and the exact representation of polynomials in several variables via the superpositions of polynomials in one variable. Russian Mathematics. 1998;42(5):4–7.
10. Haykin S. Neural Networks: A Comprehensive Foundation. Upper Saddle River, NJ: Prentice Hall; 1999. 842 p.
11. Nikolenko S, Kadurin A, Arkhangelskaya E. Deep Learning. Saint Petersburg: Piter; 2018. 480 p. (in Russian).
12. Cook S. CUDA Programming: A Developer's Guide to Parallel Computing with GPUs. Morgan Kaufmann; 2012. 592 p.
13. Jouppi NP, Young C, Patil N, Patterson D, Agrawal G, Bajwa R, Bates S, Bhatia S, Boden N, Borchers A, Boyle R, Cantin PL, Chao C, Clark C, Coriell J, Daley M, Dau M, Dean J, Gelb B, Ghaemmaghami TV, Gottipati R, Gulland W, Hagmann R, Ho CR, Hogberg D, Hu J, Hundt R, Hurt D, Ibarz J, Jaffey A, Jaworski A, Kaplan A, Khaitan H, Killebrew D, Koch A, Kumar N, Lacy S, Laudon J, Law J, Le D, Leary C, Liu Z, Lucke K, Lundin A, MacKean G, Maggiore A, Mahony M, Miller K, Nagarajan R, Narayanaswami R, Ni R, Nix K, Norrie T, Omernick M, Penukonda N, Phelps A, Ross J, Ross M, Salek A, Samadiani E, Severn C, Sizikov G, Snelham M, Souter J, Steinberg D, Swing A, Tan M, Thorson G, Tian B, Toma H, Tuttle E, Vasudevan V, Walter R, Wang W, Wilcox E, Yoon DH. In-datacenter performance analysis of a Tensor Processing Unit. ACM SIGARCH Computer Architecture News. 2017;45(2):1–12. DOI: 10.1145/3140659.3080246.
14. Welser J, Pitera JW, Goldberg C. Future computing hardware for AI. In: 2018 IEEE International Electron Devices Meeting (IEDM). 1-5 December 2018, San Francisco, CA, USA. New York: IEEE; 2018. P. 131–136. DOI: 10.1109/IEDM.2018.8614482.
15. Karras K, Pallis E, Mastorakis G, Nikoloudakis Y, Batalla JM, Mavromoustakis CX, Markakis E. A hardware acceleration platform for AI-based inference at the edge. Circuits, Systems, and Signal Processing. 2020;39(2):1059–1070. DOI: 10.1007/s00034-019-01226-7.
16. Kuptsov PV, Kuptsova AV, Stankevich NV. Artificial neural network as a universal model of nonlinear dynamical systems. Russian Journal of Nonlinear Dynamics. 2021;17(1):5–21. DOI: 10.20537/nd210102.
17. Kuptsov PV, Stankevich NV, Bagautdinova ER. Discovering dynamical features of Hodgkin–Huxley-type model of physiological neuron using artificial neural network. Chaos, Solitons & Fractals. 2023;167:113027. DOI: 10.1016/j.chaos.2022.113027.
18. Sherman A, Rinzel J, Keizer J. Emergence of organized bursting in clusters of pancreatic beta-cells by channel sharing. Biophysical Journal. 1988;54(3):411–425. DOI: 10.1016/S0006-3495(88)82975-8.
19. Stankevich N, Mosekilde E. Coexistence between silent and bursting states in a biophysical Hodgkin-Huxley-type of model. Chaos. 2017;27(12):123101. DOI: 10.1063/1.4986401.
20. Malashchenko T, Shilnikov A, Cymbalyuk G. Six types of multistability in a neuronal model based on slow calcium current. PLoS ONE. 2011;6(7):e21782. 10.1371/journal.pone.0021782.

21. Rozhnova MA, Pankratova EV, Stasenko SV, Kazantsev VB. Bifurcation analysis of multistability and oscillation emergence in a model of brain extracellular matrix. Chaos, Solitons & Fractals. 2021;151:111253. DOI: 10.1016/j.chaos.2021.111253.
22. Pankratova EV, Sinitsina MS, Gordleeva S, Kazantsev VB. Bistability and chaos emergence in spontaneous dynamics of astrocytic calcium concentration. Mathematics. 2022;10(8):1337. DOI: 10.3390/math10081337.
23. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical Recipes: The Art of Scientific Computing. 3rd Edition. New York: Cambridge University Press; 2007. 1256 p.
24. Shilnikov A, Cymbalyuk G. Transition between tonic spiking and bursting in a neuron model via the blue-sky catastrophe. Phys. Rev. Lett. 2005;94(4):048101. DOI: 10.1103/PhysRevLett.94.048101.
25. Marković D, Mizrahi A, Querlioz D, Grollier J. Physics for neuromorphic computing. Nature Reviews Physics. 2020;2:499–510. DOI: 10.1038/s42254-020-0208-2.
26. Stankevich N, Koseska A. Cooperative maintenance of cellular identity in systems with intercellular communication defects. Chaos. 2020;30(1):013144. DOI: 10.1063/1.5127107.
27. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv:1412.6980. arXiv Preprint; 2014. DOI: 10.48550/arXiv.1412.6980.