Electronic Transactions on Numerical Analysis.
Volume 56, pp. 138–158, 2022.
Copyright © 2022, Kent State University.
ISSN 1068–9613.
DOI: 10.1553/etna_vol56s138

ETNA
Kent State University and
Johann Radon Institute (RICAM)

APPROXIMATION OF A MARINE ECOSYSTEM MODEL BY ARTIFICIAL NEURAL NETWORKS*

MARKUS PFEIL† AND THOMAS SLAWIG†

Abstract. Marine ecosystem models are important to identify the processes that affect for example the global carbon cycle. Computation of an annually periodic solution (i.e., a steady annual cycle) for these models requires a high computational effort. To reduce this effort, we approximate an exemplary marine ecosystem model by different artificial neural networks (ANNs). We use a fully connected network (FCN), then apply the sparse evolutionary training (SET) procedure, and finally apply a genetic algorithm (GA) to optimize, inter alia, the network topology. With all three approaches, a direct approximation of the steady annual cycle is not sufficiently accurate. However, using the mass-corrected prediction of the ANN as initial concentration for additional model runs, the results are in very good agreement. In this way, we achieve a runtime reduction by about 15%. The results from the SET algorithm are comparable to those of the FCN. Further application of the GA may lead to an even higher reduction.

Key words. deep learning, genetic algorithm, sparse evolutionary training, biogeochemical modeling, marine ecosystem modeling, transport matrix method

AMS subject classifications. 68T07, 68U99, 92F05

1. Introduction.

In the field of climate research, marine ecosystem models are important to investigate changes in the marine ecosystem such as the uptake and storage of atmospheric CO_2 by the ocean. In general, these models are a fundamental element for the analysis of various biogeochemical processes. A marine ecosystem model consists of a global circulation model coupled to a biogeochemical model to take the interplay of physical and biogeochemical processes into account; cf. [13, 51]. The equations and variables describing the physical processes are well known while there is generally no set of equations and variables describing the biogeochemical processes. Therefore, biogeochemical models of varying complexity exist, which differ in the number of state variables and parametrizations; see, e.g., [26]. Phosphate, for example, is a crucial nutrient modeled in biogeochemical models, because the amount of this nutrient limits phytoplankton growth. Consequently, the phosphate concentration limits also the CO_2 uptake.

The computational effort of simulation runs with marine ecosystem models is very high. A model evaluation is already expensive due to the fully coupled model [44]. More importantly, the computation of steady annual cycles is expensive, because it requires a long-time integration over several millennia; cf. [3, 6, 8]. The high computational effort becomes even more obstructive when performing parameter sensitivity, uncertainty, or identification studies; cf. [48]. In these types of applications, typically, a high number of model evaluations is necessary.

Several strategies enable the reduction of the computational effort to compute a steady annual cycle of a marine ecosystem model. Firstly, the application of the *offline* simulation instead of the fully coupled model (the so called *online* simulation) lowers the computational effort. An offline simulation neglects the influence of the biogeochemical model on the ocean circulation and uses pre-computed data of the ocean currents. Khatiwala et al. [22, 23] introduced, secondly, the *transport matrix method (TMM)*, which also lowers the computational effort with a reasonable loss of accuracy. Therein, they reduce the computation of the global ocean circulation to a matrix-vector multiplication. The method decouples the evaluation of

*Received September 30, 2020. Accepted October 6, 2021. Published online on January 10, 2022. Recommended by Xiao-Chuan Cai.

†KMS – Centre for Interdisciplinary Marine Science, Dept. of Computer Science, Kiel University, 24098 Kiel, Germany ({mpf, ts}@informatik.uni-kiel.de).

the biogeochemical model from the ocean circulation, allowing for a more modular software structure that can be used with a variety of biogeochemical models; cf. [47].

In recent years, deep learning [16, 30] based on artificial neural networks (ANNs) has led to breakthroughs in various areas, for instance classification, speech recognition, computer vision, and bioinformatics. The ability to learn features from observational data makes deep learning so powerful. In this paper, we apply deep learning to learn the features of steady annual cycles of a marine ecosystem model from pre-computed steady annual cycles. Using the trained ANN reduces the computational effort, because the long-time integration is no longer necessary.

Motivated by the computational and memory benefits, the interest in conceiving ANNs with a sparse topology has, recently, increased; see, e.g., [2, 9, 39, 40]. More specifically, these methods maintain the sparse topology throughout the training. In contrast, there are various techniques that approximate a pre-trained dense neural network by a sparse one (see, e.g., [14, 17, 31]), whereby the training process remains inefficient.

Genetic algorithms simplify the design process of the network architecture; see, e.g., [37, 49, 53]. Usually, the search for an appropriate network architecture is a manual process that is time-consuming and based on experience. The network architecture depends, particularly, on the underlying problem. Genetic algorithms automate this process. NeuroEvolution of Augmenting Topologies (NEAT) [19, 53] is, for example, an evolutionary algorithm optimizing both the parameters (weights) and the architecture of an ANN, but it has difficulty scaling due to the very large search space. Therefore, we combined a genetic algorithm with the sparse evolutionary training procedure [40] to obtain a sparse neural network approximating the steady annual cycle of the marine ecosystem model reasonably.

The remainder of this paper is organized as follows: in the following section, we describe the marine ecosystem model including the computation of a steady annual cycle. In Section 3, we introduce the methods used to train the neural networks together with the resulting ANNs. Numerical results of the approximation of the steady annual cycle by different ANNs are presented in Section 4. Finally, the paper closes with a summary and conclusions.

2. Marine ecosystem model.

In ocean and climate science, the marine ecosystem is modeled by a system of differential equations for a given numbers of ecosystem species; or tracers. The number of considered species differs from model to model. It defines the size of the system of differential equations in the model. One of the simplest models is the one we consider here. It has only one species, namely the amount of nutrients, PO_4. More complex models contain phyto- and zooplankton and include the process of photosynthesis as well as grazing. An example for such kind of complex model is the HAMOCC model [35] with about 50 species. The differential equations in the system are of Lotka-Volterra or predator-prey type. They may include non-autonomous coefficient functions that model the dependency of some processes on light, that is on time and space, and sinking that is on space only. Additionally, the system includes the spatial transport of the tracers by the ocean currents, in the form of spatially discretized advection and diffusion operators.

2.1. Model equations for marine ecosystems.

Putting all of the above together, we arrive at the following tracer transport model, which is written down here for just one tracer variable $y = y(x,t)$. Here, $x \in \Omega \subset \mathbb{R}^3$ is a spatial point in the ocean Ω and $t \in [0,1]$ a time instant in the considered interval of one year.

$$\frac{\partial y}{\partial t}(x,t) + (D(x,t) + A(x,t))\, y(x,t) = q(x,t,y,u), \qquad x \in \Omega, t \in [0,1], \tag{2.1}$$

$$\frac{\partial y}{\partial n}(x,t) = 0, \qquad x \in \partial\Omega, t \in [0,1]. \tag{2.2}$$

Here, (2.2) describes a homogeneous Neumann boundary condition including the normal derivative. The right-hand side q summarizes all biogeochemical terms. Thus, it is sometimes referred to as the *biogeochemical model* in contrast to the *marine ecosystem model*. The latter includes the effects of the ocean circulation and is the whole system (2.1) to (2.6). If more than one tracer is present, the biogeochemical model q will also include the coupling of the different species. Moreover, q depends on some model parameters, here summarized in the vector $u \in \mathbb{R}^{n_u}$, $n_u \in \mathbb{N}$, as well as explicitly on time and space, e.g., due to the variability of solar radiation and its influence on the biogeochemical processes.

The linear operators D and A correspond to diffusion and advection coming from the ocean circulation, respectively. Diffusion is used as a model of turbulent effects of the ocean circulation on the tracer concentration. Molecular diffusion of the tracers themselves is known to be much smaller in relation to the diffusion induced by turbulence and is neglected. In ocean circulation modeling, the diffusion operator is usually split into a sum $D = D_h + D_v$ of a horizontal and a vertical part. This is motivated by the quite different spatial scales in horizontal and vertical direction, which requires an implicit treatment of the vertical part in the time integration. In both directions, diffusion may be modeled in the second-order form as

$$D_h(x,t)y(x,t) := -\text{div}_h\left(\kappa_h(x,t)\nabla_h y(x,t)\right), \tag{2.3}$$

$$D_v(x,t)y(x,t) := -\frac{\partial}{\partial z}\left(\kappa_v(x,t)\frac{\partial y}{\partial z}(x,t)\right), \tag{2.4}$$

where div_h and ∇_h denote the horizontal divergence and gradient, respectively, and where z is the vertical coordinate. Since the diffusion is modeling turbulence of the ocean flow only, the diffusion coefficient fields $\kappa_h, \kappa_v : \Omega \times [0,1] \to \mathbb{R}$ are the same for all tracers. In some models (see, e.g., [45]), a biharmonic (4th order) diffusion is used for the vertical part. The advection is modeled as

$$A(x,t)y(x,t) := \text{div}\left(v(x,t)y(x,t)\right) \tag{2.5}$$

with a given velocity field $v : \Omega \times [0,1] \to \mathbb{R}^3$.

As mentioned above, we are looking for a periodic solution or steady annual cycle, i.e., a solution of (2.1), (2.2) additionally satisfying

$$y(x,0) = y(x,1), \quad x \in \Omega. \tag{2.6}$$

For this purpose, we assume that the operators A, D and the function q are also annually periodic in time.

2.2. Biogeochemical model. The model, we consider here as example, is the simplified version of the N-DOP model described in [26] and [47]. The only variable y represents nutrients (thus, N) in the form of PO_4. The biogeochemical model term takes the form

$$q(x,t,y,u) = f(x,t,y) - p(x,t,y(x,t)), \quad x \in \Omega, t \in [0,1]. \tag{2.7}$$

The functions f and p depend on some parameters, summarized in the vector u as argument for the function q on the left-hand side of (2.7), whereas this dependency is omitted in the notation for f and p for simplicity.

The *phytoplankton production* (or *biological uptake*) *function* is defined as

$$p : \Omega \times [0,1] \times \mathbb{R}_{>-K_N} \to \mathbb{R}, \quad p(x,t,y) := \mu_P y_P^* I(x,t)\frac{y}{K_N + y},$$

TABLE 2.1
Reference parameter values (u_{ref}) used for the N model as well as lower (b_ℓ) and upper (b_u) bounds used to generate the Latin hypercube sample.

	Parameter	u_{ref}	b_ℓ	b_u	Unit
k_w	Attenuation coefficient of water	0.02	0.01	0.05	m^{-1}
μ_P	Maximal growth rate	2.0	1.0	4.0	d^{-1}
K_N	Half-saturation constant for PO_4 uptake	0.5	0.25	1.0	$\mathrm{mmol\,P\,m}^{-3}$
K_I	Compensation light intensity	30.0	15.0	60.0	$\mathrm{W\,m}^{-2}$
b	Implicit representation of sinking speed	0.858	0.7	1.5	1
y_P^*	Prescribed amount of phytoplankton	0.0028	*(constant)*		$\mathrm{mmol\,P\,m}^{-3}$

where μ_P, K_N, and y_P^* are parameters; see Table 2.1. Here, we used the notation in [47]. The function p models the uptake of nutrients by phytoplankton depending on the available nutrients and light. The light limitation function $I : \Omega \times [0, 1] \to \mathbb{R}_{\geq 0}$ uses insolation data and has two additional parameters K_I, k_w. Below a certain depth in the ocean, I is set to zero, reflecting the fact that light does not penetrate to the lower layers. Consequently, p is non-zero only in the upper part of Ω.

In contrast to p, the function f in (2.7) depends non-locally on y. It models the nutrient flux from the upper to the lower layers due to sinking. Without going too deep into details of the modeling (see [26] for a description), the production in the upper layers modeled by p is distributed to the lower ones. The sinking speed is indirectly modeled by a parameter $b \in \mathbb{R}_{>0}$ appearing, with a negative sign, as an exponent of the vertical coordinate z in f. As a result, the uptake modeled by the function f decreases with depth, and the parameter b describes this decrease. As a consequence, a smaller value of b implies a higher sinking speed.

Summarizing, the model has $n_u = 5$ parameters that may be varied in certain ranges for parameter studies and model calibration. We combine them in the vector

$$u = (k_w, \mu_P, K_N, K_I, b)\,. \tag{2.8}$$

Reference values and bounds are shown in Table 2.1. The parameter y_P^* is regarded as fixed.

2.3. Transport matrix method. The aim of the application presented here is, as mentioned above, to study the dependence of periodic solutions (or, in other terms, steady annual cycles) of a marine ecosystem model (2.1) to (2.6) w.r.t. the parameter values of the included biogeochemical model function q. For the N model, these would be the $n_u = 5$ parameters in (2.8). Naturally, a periodic solution also depends on the given circulation data, which are the velocity field v and the diffusion fields κ_h, κ_v. But since this dependency is not our focus, we use annually periodic reference values of these fields. These fields can be pre-computed by some ocean circulation model that was run into a steady annual cycle.

Using the fields v, κ_h, and κ_v directly would require to implement a discretization scheme for the diffusion and advection operators D and A. Since the application of the two operators on a spatially discretized tracer vector $\mathbf{y} = \mathbf{y}(t)$ is a linear operation, a more efficient method was presented in [22, 23]. In the *transport matrix method (TMM)*, not the fields v, κ_h, and κ_v are stored, but the matrices that represent the application of the discretized operators (using these fields) on a discrete tracer vector are computed and stored. The details of the computation are presented in [23].

To describe the usage of transport matrices for the simulation of a marine ecosystem model, we assume that the computational domain Ω (i.e., the ocean) is discretized in space, resulting in $n_x \in \mathbb{N}$ grid points x_k, $k = 1, \ldots, n_x$. Moreover, we discretize time by introducing an equidistant grid

$$t_j := j\Delta t, \quad j = 0, \ldots, n_t, \quad \Delta t := \frac{1}{n_t}.$$

The numerical approximation of a spatially discrete tracer vector at fixed time instant t_j is then denoted by

$$\mathbf{y}_j \approx (y(x_k, t_j))_{k=1}^{n_x} \in \mathbb{R}^{n_x}.$$

We discretize the equation (2.1) in time using a semi-implicit Euler scheme, where advection and horizontal diffusion are treated explicitly and vertical diffusion implicitly. We denote by $\mathbf{A}_j$, $\mathbf{D}_j^h$, and $\mathbf{D}_j^v$ the spatially discretized counterparts of the operators A, D_h, and D_v, respectively, at time step j, where the Neumann boundary conditions are already included. This results in a time-stepping

$$\mathbf{y}_{j+1} = \left(I + \Delta t \mathbf{A}_j + \Delta t \mathbf{D}_j^h\right) \mathbf{y}_j + \Delta t \mathbf{D}_j^v \mathbf{y}_{j+1} + \Delta t \mathbf{q}_j(\mathbf{y}_j, u), \quad j = 0, \ldots, n_t - 1.$$

Here, $I \in \mathbb{R}^{n_x \times n_x}$ is the identity matrix and

$$\mathbf{q}_j(\mathbf{y}_j, u) := (q(x_k, t_j, \mathbf{y}_j, u))_{k=1}^{n_x} \in \mathbb{R}^{n_x}, \quad j = 0, \ldots, n_t - 1,$$

the spatially discretized biogeochemical term. Defining the *explicit and implicit transport matrices*

$$\begin{aligned} \mathbf{T}_j^{\text{exp}} &:= I + \Delta t \mathbf{A}_j + \Delta t \mathbf{D}_j^h \in \mathbb{R}^{n_x \times n_x}, \\ \mathbf{T}_j^{\text{imp}} &:= \left(I - \Delta t \mathbf{D}_j^v\right)^{-1} \in \mathbb{R}^{n_x \times n_x}, \quad j = 0, \ldots, n_t - 1, \end{aligned}$$

we obtain

$$\mathbf{y}_{j+1} = \mathbf{T}_j^{\text{imp}} \left(\mathbf{T}_j^{\text{exp}} \mathbf{y}_j + \Delta t \mathbf{q}_j(\mathbf{y}_j, u)\right) =: \phi_j\left(\mathbf{y}_j, u\right), \quad j = 0, \ldots, n_t - 1. \tag{2.9}$$

Thus, a time step of the marine ecosystem simulation consists of two matrix multiplications and one evaluation of the biogeochemical model.

The transport matrices are generated from a grid-point-based ocean circulation model. Hence, the explicit ones are sparse. For the implicit ones (which are the inverse of discretization matrices) this also holds since they only include the vertical diffusion part. In practical computations, pairs of transport matrices cannot be stored for all time-steps in a year. In contrast, monthly averaged matrices are saved. Assuming annually periodicity of the ocean circulation, then only two sets of each 12 matrices have to be stored. This makes the evaluation of (2.9) very efficient.

2.4. Computation of steady annual cycles. Looking for a periodic solution (i.e., a steady annual cycle) in the fully discrete setting means that we try to obtain

$$\mathbf{y}_{n_t} = \mathbf{y}_0 \in \mathbb{R}^{n_x}$$

when applying the above iteration (2.9) over one year model time. The mapping that performs one year is given by

$$\Phi := \phi_{n_t - 1} \circ \cdots \circ \phi_0$$

with ϕ_j defined in (2.9). We therefore look for a fixed-point of this mapping. A classical fixed-point iteration takes the form

$$\mathbf{y}^{\ell+1} = \Phi\left(\mathbf{y}^{\ell}, u\right), \quad \ell = 0, 1, \ldots, \tag{2.10}$$

where we start with an arbitrary vector $\mathbf{y}^0 \in \mathbb{R}^{n_x}$. The superscript ℓ may now be considered as a model year if the fixed-point iteration (2.10) is interpreted as pseudo-time stepping or *spin-up*, a term which is widely used in ocean and climate research.

From numerous numerical experiments performed with the model with a wide range of parameter vectors (e.g., results used in [47]), we found that the fixed-point or steady annual cycle for a distinct parameter vector was unique and was reached by (2.10) independently of the initial value $\mathbf{y}^0$, even though this was not mathematically proved by now.

Testing for numerical convergence of the iteration, we used the difference between two consecutive iterates determined by

$$\varepsilon_\ell := \left\|\mathbf{y}^{\ell} - \mathbf{y}^{\ell-1}\right\| \tag{2.11}$$

for iteration (model year) $\ell = 1, 2, \ldots$. As norm we used the weighted Euclidean norm

$$\|v\|_{2,w} := \left(\sum_{k=1}^{n_x} w_k v_k^2\right)^{\frac{1}{2}}, \quad v = (v_k)_{k=1}^{n_x} \in \mathbb{R}^{n_x},$$

with weights $w_k \in \mathbb{R}_{>0}, k \in \{1, \ldots, n_x\}$. Here, the weights can be set to the box volumes corresponding to the grid points x_k, denoted by $|V_k|, k = 1, \ldots, n_x$. For all weight vectors $w \in \mathbb{R}^{n_x}_{>0}$ and any vector $v \in \mathbb{R}^{n_x}$, we have

$$\min_{1 \le k \le n_x} \sqrt{w_k}\, \|v\|_2 \le \|v\|_{2,w} \le \max_{1 \le k \le n_x} \sqrt{w_k}\, \|v\|_2,$$

i.e., all these norms are equivalent to the Euclidean norm. We used similar norms for the whole trajectory over one model year. For a weight vector w as above, we define

$$\|v\|_{2,w,T} := \left(\sum_{j=0}^{n_t-1} \Delta t \sum_{k=1}^{n_x} w_k v_{jk}^2\right)^{\frac{1}{2}}, \quad v = (v_{jk})_{j=0,\ldots,n_t-1,k=1,\ldots,n_x} \in \mathbb{R}^{n_t \times n_x},$$
$$\|v\|_{2,T} := \|v\|_{2,w,T} \quad \text{for } w_k = 1, k = 1, \ldots, n_x.$$

3. Artificial neural networks.

We approximated the steady annual cycle of the marine ecosystem model by ANNs. In this study, we applied a multilayer perceptron with a feed-forward architecture. Multilayer perceptrons consist of an input layer, at least one hidden layer and an output layer. Typically, only the neurons between two consecutive layers are fully connected without internal loops.

To predict the tracer concentrations of the steady annual cycle from the model parameters, the input layer consisted of $n_u = 5$ neurons corresponding to the model parameters defined in (2.8). Although the steady annual cycle comprises the complete trajectory of a model year, we used only 52 749 neurons in the output layer corresponding to the discretization of the tracer concentration for the first time of the model year. Taking the whole annual trajectory with 2880 time-steps, the output layer would consist of almost 152 million neurons so that the training of the ANN would be infeasible.

We trained the ANNs with the approach of supervised learning [18] using backpropagation. For the optimization, we applied either the stochastic gradient descent (SGD) [4, 50] or the adaptive moment estimation (Adam) algorithm [24]. The loss function was the mean squared error; activation functions were the exponential linear unit (ELU) [7] for the neurons of the hidden layers and the scaled exponential linear unit (SELU) [25] or rectified linear unit (ReLU) [15, 43] for the neurons of the output layer, respectively. In addition, we made partial use of the regularization strategy dropout [16, 52].

Since the marine ecosystem model preserves mass, the total mass of the tracer concentration predicted by an ANN turned out to be crucial for the accuracy of the prediction. To guarantee mass conservation, we applied the following pointwise adjustment on a prediction $\mathbf{y}_{\text{ANN}}$:

$$\tilde{\mathbf{y}}_{\text{ANN}} = \frac{\sum_{k=1}^{n_x} |V_k|\, m}{\sum_{k=1}^{n_x} |V_k|\, \mathbf{y}_{\text{ANN}}(x_k, t_0)} \mathbf{y}_{\text{ANN}}. \tag{3.1}$$

It uses the box volumes $|V_k|$, $k = 1, \dots, n_x$, of the discretization and a global mean tracer concentration $m \in \mathbb{R}_{>0}$ in order to obtain the required overall mass.

In the training procedure of an ANN (i.e., the optimization of its internal weights), a given set of pairs of input and output data is used. A subset of these pairs is used in the optimization as *training data* to adjust the network weights. Another part is used during the optimization process to monitor how the already partially optimized network behaves on these *validation data* that were not used in the weight optimization. After finishing the training procedure, another part is used as *test data* to assess the quality of the trained network.

3.1. Sparse evolutionary training algorithm. Motivated by the sparsity in biological neural networks [46, 54], the sparse evolutionary training (SET) algorithm [33, 40] replaces the fully connected layers with sparse ones. The network size of neural networks with fully connected layers is limited by computational limitations. Besides, the training of these networks often ends up with many weights around zero [17]. Furthermore, an increasing number of neurons and the interlayer connectivity both increase the computational complexity. The SET procedure applies sparse neural networks using sparsely connected layers to lower the computational complexity by quadratically reduction of the number of parameters in both phases; training and inference. It also reduces memory requirements. In particular, sparsity remains unchanged due to adaptive sparse connectivity during training. The SET procedure evolves an initial sparse network using an evolutionary approach into a scale-free network [1] with more structured connections. This approach differs from methods that generate the sparse topology only during or after the training process; e.g., weight pruning [17, 31]. Moreover, this method achieves a very high accuracy performance [40, 57] exceeding, often, even its dense counterpart [34].

The SET algorithm initializes each layer of the sparse network as an Erdős-Rényi random graph [11, 12]. In this graph, a neuron can be connected to any number of neurons of the subsequent layer. The weight matrix $W^k \in \mathbb{R}^{n^{k-1} \times n^k}$ contains the weights of each connection between the neurons of the successive layers $k-1$ and k, where n^ℓ represents the number of neurons of layer ℓ. A hyperparameter $\varepsilon \in \mathbb{R}_{>0}$ controls the sparsity of the Erdős-Rényi random graph in the following way. The probability of the connection between the i-th neuron of layer $k-1$, $i \in \{1, \dots, n^{k-1}\}$, and the j-th neuron of layer k, $j \in \{1, \dots, n^k\}$, is chosen as

$$P\left(W_{ij}^k\right) = \min\left(\frac{\varepsilon(n^k + n^{k-1})}{n^k n^{k-1}}, 1\right).$$

Thus, the sparse topology contains $|W^k| = \varepsilon(n^k + n^{k-1})$ connections instead of $n^k n^{k-1}$ connections of a fully connected layer.

This SET procedure adjusts the topology after each training epoch, following an evolutionary idea. This is motivated by the fact that the randomly generated initial sparse topology is not necessarily suitable to learn the particularities of the data of the underlying problem. More specifically, the algorithm removes a fraction $\zeta \in (0, 1)$ of the connections with the smallest positive and largest negative weights (i.e., the weights closest to zero) whose removal does not noticeably affect the model performance [17, 56]. Afterwards, the procedure adds an amount of random selected connections with random weights, equal to the amount of previously removed connections, so that the number of connections remains constant during training. In analogy to an evolutionary algorithm, the removal of connections corresponds to the selection and the random insertion to the mutation. In short, the sparse initialization of the networks as Erdős-Rényi random graph together with the removal and random insertion of connections constitutes the core of the sparse evolutionary procedure.

3.2. Hyperparameter adaption using genetic algorithm. In this study, we applied a genetic algorithm (GA) to design a network with (preferably) optimal architecture and hyperparameters. GAs, introduced by John Holland in the 1960s [21], belong to the class of evolutionary algorithms (see, e.g., [10]) based on the Darwinian principle of evolution through selection. Due to the number of possible combinations of the hyperparameters in an ANN (such as number of layers, number of neurons in each layer, activation functions, network optimizer, learning rate), it is not possible to train the network for all of their combinations in order to determine the optimal one and to obtain, as result, the "best" ANN.

Starting from a number of randomly chosen vectors (the initial "population") of hyperparameters, a GA (see, e.g., [38, Chapter 1.6]) evolves the population over several iterations ("generations") by creating new parameter vectors ("offspring") from the best members of the current population. The population size and the number of generated offspring are important parameters of a GA. They depend on the size and complexity of the underlying problem. In every generation, new parameter vectors ("individuals") are built by random changes ("mutation") and/or recombination of the "genes" (i.e., parameter values) of the individuals. The best among parents and offspring make it into the next generation. This selection is based on a fitness function, which in our case was the training loss. Lastly, the GA terminates after a defined number of generations.

We used truncation selection [42] and discrete recombination (see, e.g., [5, 42]) to create the next generation. We divided the member of the next generation into three groups:

1. The best 25% of the current population become part of the next generation.
2. Another 7.5% of the new population were taken randomly from the remaining individuals. For all of them, we randomly mutated one (also randomly chosen) hyperparameter.
3. The remaining 67.5% of the new population were generated by discrete recombination of the individuals in the above two groups. For this purpose, we randomly selected two members out of them as parents. We generated one offspring by taking the parents' hyperparameter values with a probability of 0.5 each. As a consequence, the offspring usually received hyperparameters from both parents. Moreover, a randomly selected hyperparameter was mutated with probability of 0.3 after recombination.

We trained the ANNs, whose hyperparameters were selected by the GA, with the SET procedure described in Section 3.1. Table 3.1 contains all hyperparameter adapted by the GA including the parameter ε, ζ of the SET procedure. We applied the activation function for all layers of the ANN.

Among the hyperparameters, there is a dependency between the number of layers and the number of neurons for each layer. In order to correct this possible discrepancy the number of neurons for each layer was adjusted after a recombination or mutation such that the length of

TABLE 3.1
Hyperparameter used in the genetic algorithm.

Hyperparameter	Range of values
Number of layers n_n	$\{1, \dots, 10\}$
Number of neurons	$\{s \in \{1, \dots, 1000\}^{n_n}, s_i \leq s_{i+1}, i = 1, \dots, n_n - 1\}$
Activation function	ELU, SELU, ReLU, Linear
Network optimizer	SGD, Adam
Learning rate	$\{10^{-1}, 10^{-2}, \dots, 10^{-5}\}$
ε	$[1, 1000]$
ζ	$[0.1, 0.9]$

the number of neurons always corresponds to the number of layers. Furthermore, we sorted the neurons per layer in ascending order to ensure a good information flow from the few number of model parameters n_u to the large number of grid points n_x in the ocean discretization.

4. Results. In this section, we present the results obtained with three different ANNs. On the one hand, we looked at the approximation quality. On the other hand, we investigated the possible runtime reduction taking the ANNs' approximations as initial values for the original simulation and running it into a steady annual cycle.

4.1. Experimental setup. For the training of the ANNs, we created a Latin hypercube (cf. [36]) sample of the parameter vectors defined in (2.8) within the bounds from Table 2.1 and computed a steady annual cycle for each parameter vector. The sample had 1100 parameter vectors that were generated by the LHS routine of [32]. For every parameter, the routine divided the range uniformly into 1100 non-overlapping intervals and selected randomly one point from each interval. The Latin hypercube sampling combined the selected points for all parameters in a random manner so that each selected point was present in exactly one parameter vector. We then applied the marine ecosystem toolkit for optimization and simulation in 3D (Metos3D) [47] for the computation of the steady annual cycle. Here, we performed a spin-up over 10 000 model years, starting with a constant global mean tracer concentration of $2.17\,\mathrm{mmol\,P\,m^{-3}}$ for PO_4.

We trained the ANNs with the parameter vectors of the Latin hypercube sample as input. As output, we considered only the concentration of the first time instant in the last, the 10 000th model year. We randomly split these data to use 80% as training and 20% as validation data. As test data for the trained ANNs, we used 100 randomly chosen parameter vectors of the sample. We compared the predictions of the ANNs with a reference solution, denoted by $\mathbf{y}^{10000}$, namely the result obtained by a spin-up with Metos3D using a constant initial concentration, also over 10 000 model years. We used the following quantities for comparison:

- $\mathbf{y}_{\mathrm{ANN}}$: Prediction of the ANN.
- $\mathbf{y}_{\mathrm{ANN}}^{1000}$: Result of a spin-up over 1000 model years, using the prediction as initial concentration.
- $\tilde{\mathbf{y}}_{\mathrm{ANN}}$: Mass-corrected prediction of the ANN.
- $\tilde{\mathbf{y}}_{\mathrm{ANN}}^{1000}$: Result of a spin-up over 1000 model years, using the mass-corrected prediction as initial concentration.

In particular, we measured the accuracy of an approximation by the pointwise relative difference

$$\frac{\left|\mathbf{x} - \mathbf{y}^{10000}\right|}{\left\|\mathbf{y}^{10000}\right\|_2} \quad \text{for } \mathbf{x} \in \left\{\mathbf{y}_{\mathrm{ANN}}, \mathbf{y}_{\mathrm{ANN}}^{1000}, \tilde{\mathbf{y}}_{\mathrm{ANN}}, \tilde{\mathbf{y}}_{\mathrm{ANN}}^{1000}\right\}. \tag{4.1}$$

We call this quantity the *(relative) error* of the respective result $\mathbf{x}$.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

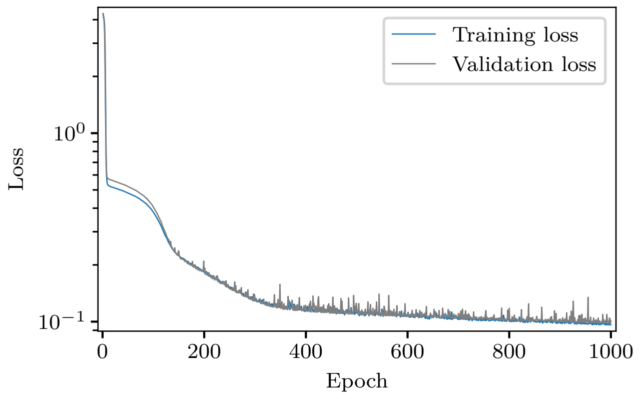

FIG. 4.1. *Training progress for the FCN with predefined structure.*

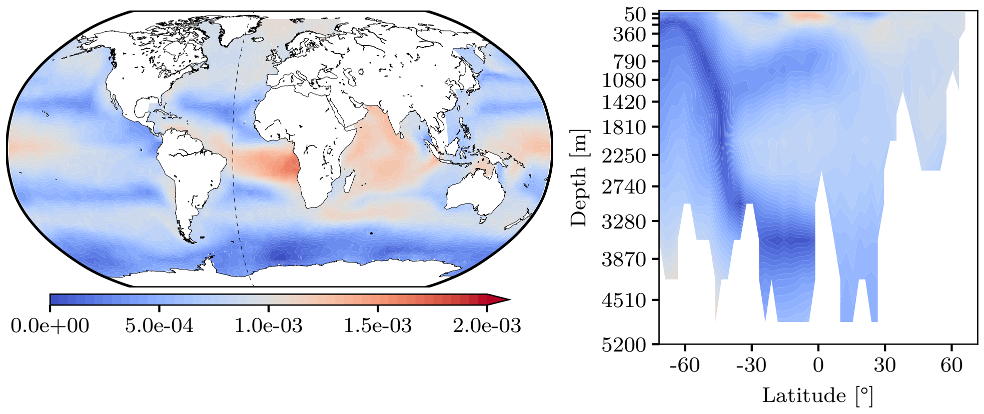

FIG. 4.2. *Pointwise relative error* (4.1) *with* $\mathbf{x} = \mathbf{y}_{ANN}$ *of the FCN's prediction for the selected test parameter vector* u_{ref} (4.2)*. Shown are concentrations at the surface layer* ($0\,\mathrm{m}$ *to* $50\,\mathrm{m}$*, left) and at a slice plane of the Atlantic at* $29.531\,25°$ *W (located at the dashed line, right) at the first time instant of the model year; in January.*

To compare possible runtime reduction, we also computed $\mathbf{y}^{10^{-4}}$ and $\tilde{\mathbf{y}}_{\mathrm{ANN}}^{10^{-4}}$, the results of spin-ups with a tolerance of 10^{-4} in (2.11), using either a constant concentration or the mass-corrected prediction of the ANN, $\tilde{\mathbf{y}}_{\mathrm{ANN}}$, as initial value.

4.2. Fully connected network with predefined structure. We trained a fully connected network (FCN) with three hidden layers using the backpropagation combined with SGD and SELU as activation function for the neurons of the output layer. Inspired by the prime decomposition of 52 750, the hidden layers of this ANN had 10, 25, and 211 neurons. Figure 4.1 displays the convergence behavior during the training, i.e., the loss function for training and validation data, respectively. Both values decreased in a similar way. Looking at the reduction, it would be possible to stop the training even after a much lower number of iterations/epochs.

Quality of the approximation. The accuracy of the prediction $\mathbf{y}_{\mathrm{ANN}}$ of the FCN was not sufficient to represent the steady annual cycle of the marine ecosystem model. Figure 4.2 depicts the pointwise error for the exemplary parameter vector

$$u_{\mathrm{ref}} = (0.02, 2.0, 0.5, 30.0, 0.858)\,; \tag{4.2}$$

TABLE 4.1
Relative error (4.1) in different norms for the parameter vector u_{ref} (4.2).

$\mathbf{x}$	$\lVert\cdot\rVert_2$	$\lVert\cdot\rVert_{2,V}$	$\lVert\cdot\rVert_{2,T}$	$\lVert\cdot\rVert_{2,V,T}$
$\mathbf{y}_{\text{ANN}}$	1.805e-01	1.738e-01	1.650e-01	1.869e-01
$\mathbf{y}_{\text{ANN}}^{1000}$	8.529e-02	8.390e-02	7.936e-02	9.052e-02
$\tilde{\mathbf{y}}_{\text{ANN}}$	1.748e-01	1.392e-01	1.595e-01	1.496e-01
$\tilde{\mathbf{y}}_{\text{ANN}}^{1000}$	2.342e-02	2.449e-02	2.178e-02	2.640e-02

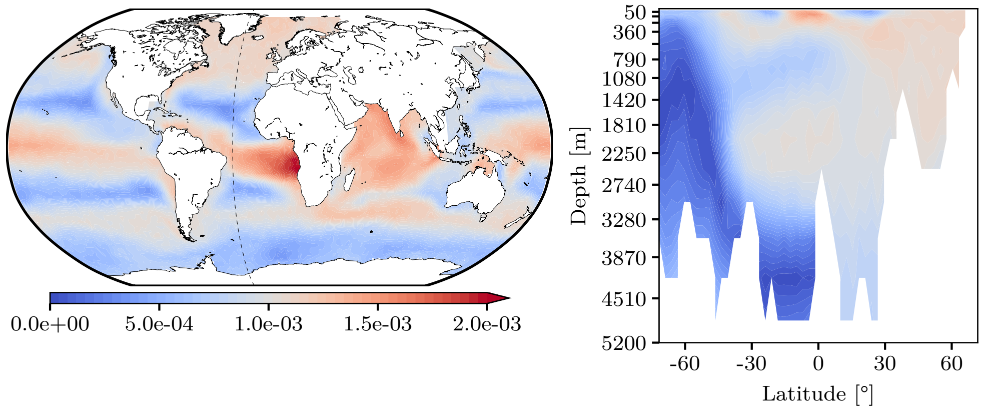

FIG. 4.3. *As Figure 4.2, but for $\mathbf{x} = \tilde{\mathbf{y}}_{ANN}$, i.e., using the mass-corrected prediction $\tilde{\mathbf{y}}_{ANN}$.*

see Table 2.1. Although the structure of the prediction resembled the tracer concentration of the reference solution, the error is big, especially near the equator. Using the prediction $\mathbf{y}_{\text{ANN}}$ as initial concentration of an additional spin-up calculation over 1000 model years (giving $\mathbf{y}_{\text{ANN}}^{1000}$) improved the approximation as listed in Table 4.1. However, the nutrient concentration in the Antarctic as well as near the equator showed still large deviations. The Table 4.1 additionally indicates that the choice of the error norm (i.e., using volume-weighted differences or comparing the whole annual trajectory) makes no qualitative difference.

The mass correction of the prediction defined in (3.1) led to a better approximation. A potential reason for the big errors of the above two approximations $\mathbf{y}_{\text{ANN}}$ and $\mathbf{y}_{\text{ANN}}^{1000}$ was a mass loss. The model preserves mass and also its implementation approximately does, but the ANN did not. In our calculations, the mass of the ANN predictions contained only 92% of the mass of the reference solution for the parameter vector u_{ref} (4.2). Thus, neither the prediction itself nor its use as an initial concentration for Metos3D was likely to provide an appropriate approximation. The mass correction of the prediction (giving $\tilde{\mathbf{y}}_{\text{ANN}}$) reduced, instead, slightly the error as can be seen from Table 4.1. More specifically, it significantly lowered the deviation in the deeper layers of the ocean, especially in the North Pacific and the northern Indian Ocean. In contrast, the discrepancy at the surface layer actually increased as we can see in Figure 4.3. Using the mass-corrected prediction $\tilde{\mathbf{y}}_{\text{ANN}}$ as initial concentration for 1000 additional simulation years considerably reduced the error and gave a much better approximation $\tilde{\mathbf{y}}_{\text{ANN}}^{1000}$; Figure 4.4 and Table 4.1. The euphotic zone (i.e., approximately the upper 150 m of the ocean; see, e.g., [41, 55]), is subject to great variability due to the annual cycle while it takes several millennia for concentrations to change in the deep ocean. Due to the the small deviation of $\tilde{\mathbf{y}}_{\text{ANN}}$ in the deeper layers of the ocean (Figure 4.3), the additional spin-up over 1000 model years (Figure 4.4) was, therefore, sufficient to decrease noticeably the error, especially at surface. Indeed, 1000 model years were not enough to significantly

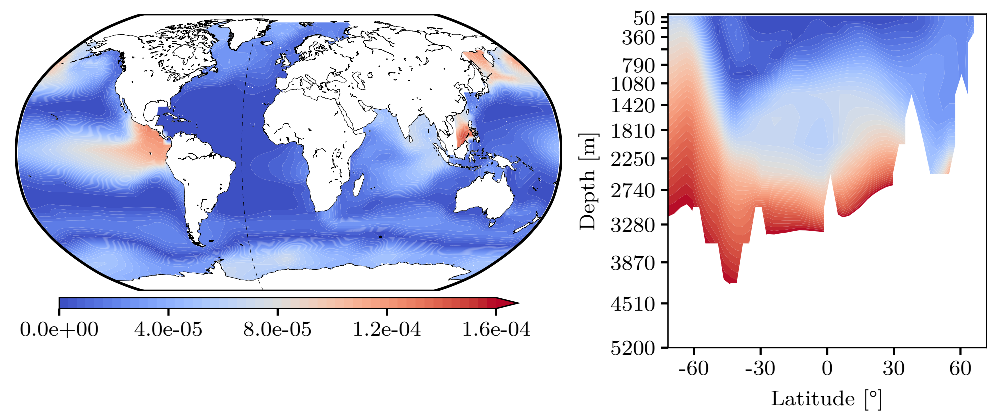

FIG. 4.4. *As Figure 4.2, but for* $\mathbf{x} = \tilde{\mathbf{y}}_{ANN}^{1000}$*, i.e., using the mass-corrected prediction* $\tilde{\mathbf{y}}_{ANN}$ *as initial value for* 1000 *additional simulation years. The range of the colorbar is smaller as in Figure 4.2, emphasizing that the error was significantly reduced.*

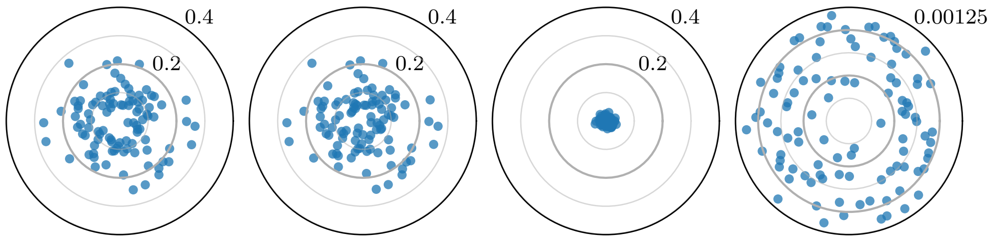

FIG. 4.5. *Visualization of the relative error (4.1) using the FCN for the whole test data set. Left to right:* $\mathbf{x} = \mathbf{y}_{ANN}, \tilde{\mathbf{y}}_{ANN}, \tilde{\mathbf{y}}_{ANN}^{1000}, \tilde{\mathbf{y}}_{ANN}^{10^{-4}}$*. The relative error corresponds to the distance to the center. In the first three plots, the distance of the circles is* 0.1*. In the rightmost, the distance is* $2.5 \cdot 10^{-4}$*, showing that the error is further reduced.*

reduce the error in the deeper layers. If we ran the simulation for additional years model time, we could reduce the error even more.

Taking the mass-corrected prediction as initial value for a spin-up was the only benefit we may take from application of the FCN. As a pictorial summary, Figure 4.5 illustrates the different quality of the approximations for the whole test data set. Comparing the first and the second graph from the left, we see that the mass correction of the prediction defined in (3.1) reduced the relative error only slightly. However, the mass correction of the prediction was crucial, because the mass of the predictions diverged by up to 10% from the required mass. In order to obtain an acceptable error (third graph from the left in Figure 4.5), it remained necessary to run the simulation for additional model years (here, 1000 model years) using the mass-corrected prediction as initial concentration. The main question was how many spin-up iterations we can save using this strategy.

Reduction of computing time. Using the mass-corrected prediction $\tilde{\mathbf{y}}_{\text{ANN}}$ of the FCN as initial concentration shortened the computational time for the spin-up calculation. We computed two spin-ups with a tolerance of 10^{-4}, first one using the constant initial concentration and the second one using the mass-corrected prediction as initial concentration. The results are $\mathbf{y}^{10^{-4}}$ and $\tilde{\mathbf{y}}_{\text{ANN}}^{10^{-4}}$, respectively. Figure 4.6 reveals that the same tolerance was reached earlier when using the mass-corrected prediction. Figure 4.7 demonstrates, moreover, this behavior for nearly the whole test data set. However, there were some parameter vectors where the spin-up calculation required more model years using the mass-corrected prediction as

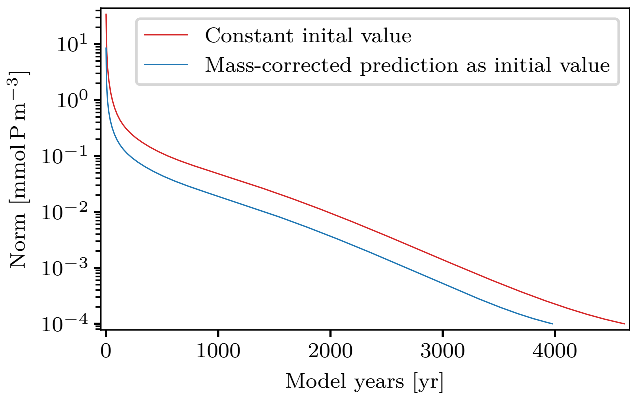

FIG. 4.6. *Difference* (2.11) *between two successive iterations (i.e., model years) in the spin-up for the exemplary parameter vector* u_{ref} (4.2)*, using either a constant or the mass-corrected prediction* $\tilde{\mathbf{y}}_{ANN}$ *of the FCN as initial value.*

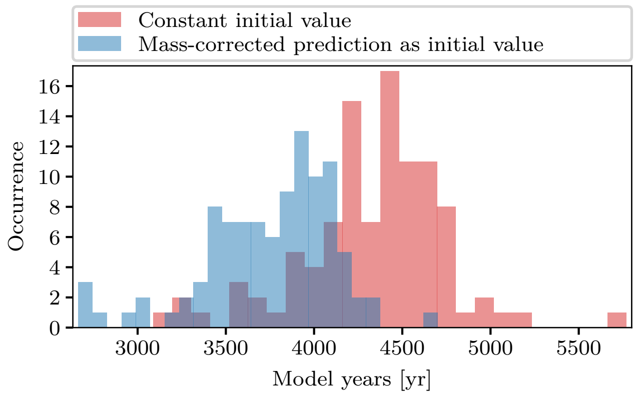

FIG. 4.7. *Required model years for reaching a tolerance* 10^{-4} *in* (2.11) *in the spin-up using either the mass-corrected prediction* $\tilde{\mathbf{y}}_{ANN}$ *of the FCN as initial value or the constant one. Displayed are results for the whole test data set.*

initial concentration than the constant one. Furthermore, we obtained a relative error less than 10^{-3}; Figure 4.5, rightmost graph. Thus, we received a suitable approximation in lowered computational time for a significant number of parameters vectors of the test data set.

4.3. ANN obtained by sparse evolutionary training algorithm. We trained an ANN with the sparse evolutionary training procedure (see Section 3.1) using $\varepsilon = 20$ and $\zeta = 0.3$ as well as the topology of the FCN described in Section 4.2. Thus, the neural network consisted of three hidden layers with 10, 25, and 211 neurons and sparse connected layers. We used additionally dropout to temporarily drop randomly selected 30% of the input units of each hidden layer during the training. Moreover, we used backpropagation combined with SGD and ReLU as activation function of the neurons of the output layer. Figure 4.8 displays the development of the training and validation loss during the training. Here, we observed more ups and downs in both loss functions. This is typically for the SET algorithm since it removes connections and randomly adds new connections between the layers. These adaptations may turn out to be beneficial or not in the next iteration and can thus increase the loss function. Moreover, we observed from Figure 4.8 that the training loss was larger than the validation loss. One reason of this larger training loss was also the removing and adding of connections

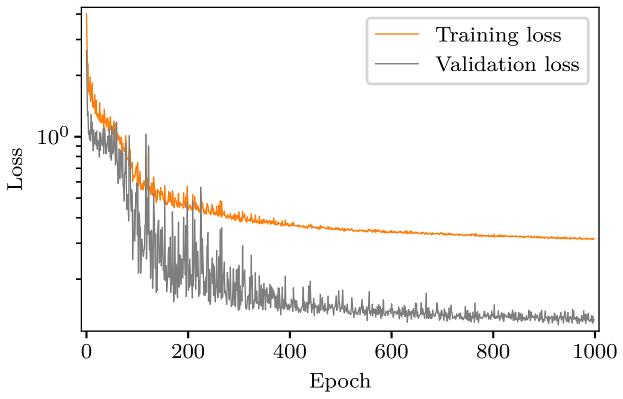

FIG. 4.8. *Training progress for an ANN using the sparse evolutionary training procedure.*

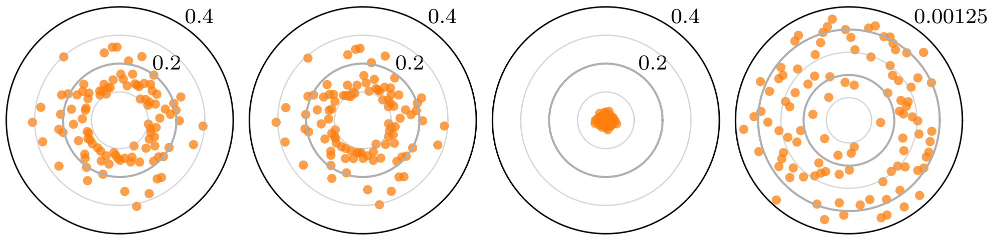

FIG. 4.9. *As Figure 4.5, but for the ANN built with the SET algorithm.*

at the beginning of each epoch. The training loss is calculated as the average of the losses for each batch of the training data. As a consequence, the loss decreases during the training (in an epoch) because the weights (especially those of the randomly added connections) are updated after each batch. Wherefore, the loss over the first batches is usually greater than over the last ones. On the contrary, the validation loss is computed at the end of an epoch using the adapted weights. A second reason for the larger training loss was the use of the regularization mechanism dropout which is only used during training but not for computing the validation loss. Again, the process could have been stopped earlier with no big change in the loss functions, if they are averaged over some iterations.

Quality of the approximation. A main difference and advantage of the ANN, we created using the SET algorithm, was that the predicted concentrations $\mathbf{y}_{\mathrm{ANN}}$ now already had approximately the correct mass. Comparing the two leftmost graphs in Figure 4.9 shows that the errors of the prediction $\mathbf{y}_{\mathrm{ANN}}$ and the mass-corrected prediction $\tilde{\mathbf{y}}_{\mathrm{ANN}}$ were almost identical. Nevertheless, both did not appropriately approximate the steady annual cycle. Applying 1000 model years in the simulation (giving $\tilde{\mathbf{y}}_{\mathrm{ANN}}^{1000}$) or computing down to a tolerance (2.11) of 10^{-4} (giving $\tilde{\mathbf{y}}_{\mathrm{ANN}}^{10^{-4}}$), we reduced the error significantly as evident from the two rightmost graphs in Figure 4.9.

Reduction of computing time. The runtime reduction using the mass-corrected prediction was similar to the one of the FCN. There was a reduction for nearly all parameter vectors in the test data set, and there were almost the same outliers as for the FCN. The results for the whole test data set can be seen in Figure 4.10.

4.4. ANN obtained by sparse evolutionary training and genetic algorithm. We applied the ANN with sparse layers maintained by the GA (see Section 3.2) using 10 generations

ETNA
Kent State University and
Johann Radon Institute (RICAM)

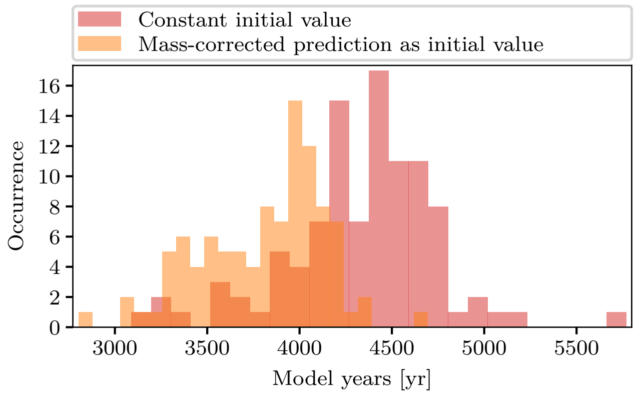

FIG. 4.10. *As Figure 4.7, but using the mass-corrected prediction of the ANN built with the SET algorithm.*

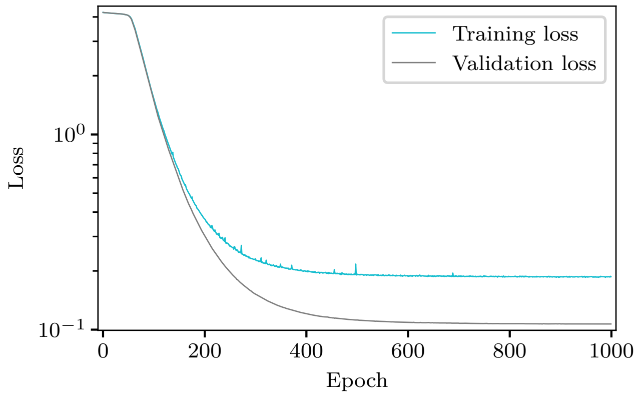

FIG. 4.11. *Training progress for the network designed by the SET algorithm and the GA.*

and a population size of 30. This ANN consisted of seven hidden layers with dropout (30% of input units set to 0 at each step during training) containing all 500 neurons except the first and seventh layer which contained 80 and 600 neurons, respectively. For this neural network, SGD was, moreover, used as optimizer, SELU as activation function for the neurons of all layers and $\varepsilon = 50$ and $\zeta = 0.1$ as parameter for the sparse evolutionary training procedure. Figure 4.11 illustrates the training progress for this ANN. As for the ANN obtained by the SET algorithm (Section 4.3), the training loss was larger than the validation loss. This resulted, on the one hand, from the property of the SET algorithm to remove and add weights at the beginning of each epoch and, on the other hand, from the use of dropout.

Quality of the approximation. The mass of the prediction of this ANN deviated only negligibly from the mass of the constant initial concentration. Hence, there were little differences seen between the predictions $\mathbf{y}_{\text{ANN}}$ and their mass-corrected counterparts $\tilde{\mathbf{y}}_{\text{ANN}}$ for the whole test data set, as depicted in the two leftmost graphs of Figure 4.12. Besides, the prediction of the ANN built with the GA (Figure 4.12, leftmost graph) approximated the steady annual cycle better than the prediction of the ANN trained with the sparse evolutionary training algorithm; Figure 4.9, leftmost graph. Taking the mass-corrected prediction $\tilde{\mathbf{y}}_{\text{ANN}}$ as initial value for 1000 additional spin-up steps or down to a tolerance of 10^{-4}, we obtained an ever better approximation of the reference solution. They are displayed in the two rightmost graphs of Figure 4.12.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

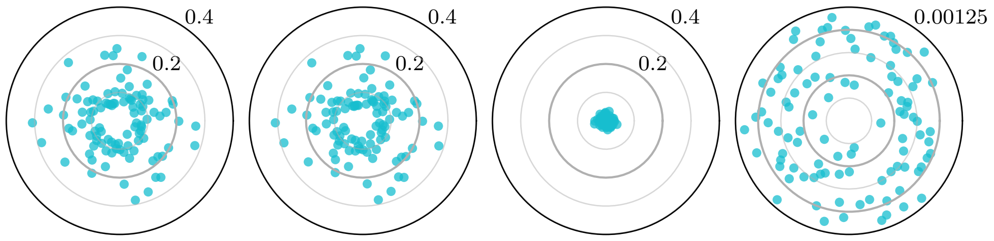

FIG. 4.12. *As Figure 4.5, but for the application of the ANN built with the SET algorithm and the GA.*

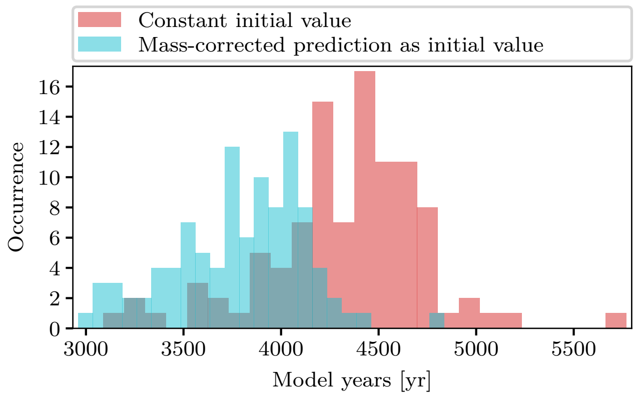

FIG. 4.13. *As Figure 4.7, but using the mass-corrected prediction of the ANN built with the SET algorithm and the GA.*

Reduction of computing time. The spin-up calculation using the mass-corrected prediction as initial concentration lowered the computational time. Figure 4.13 reveals these reductions of the computational time. The simulations achieved the spin-up tolerance of 10^{-4} in less model years than for constant initial concentration for almost all parameter vectors of the test data set. Again, there was no decrease of the computational time for the same outliers already obtained for the FCN. In particular, we obtained an appropriate approximation with a relative error less than 10^{-3} (Figure 4.12, rightmost graph) for nearly all parameter vectors.

5. Conclusions.

We approximated a marine ecosystem model using ANNs designed in three different ways to predict the steady annual cycle. The ANNs took 5 model parameters as input and should predict (the first time instant in a model year of) a steady state solution of the ecosystem as output. We applied a FCN with prescribed structure, used the SET algorithm to obtain a sparse network with the same number of layers and nodes as the FCN, and finally used a GA to optimize the hyperparameters of the network and, thus, also its structure.

None of the three networks was able to directly predict a sufficiently accurate approximation of the steady annual cycle. The FCN with fixed structure did not even predict a solution with the correct mass of the reference solution, whereas the original model preserves the mass over time. The other two networks, designed using the SET algorithm or the SET algorithm and the GA, respectively, were able to predict a solution with correct mass. Especially for the FCN, we corrected the mass by a post-processing step; see (3.1). As an alternative, we also extended the ANN by an additional layer which served exclusively for conservation of mass. Such a layer did basically the same as the correction step (3.1), i.e., its weights were defined by the mass correction property alone. The main difference was that the loss function in the

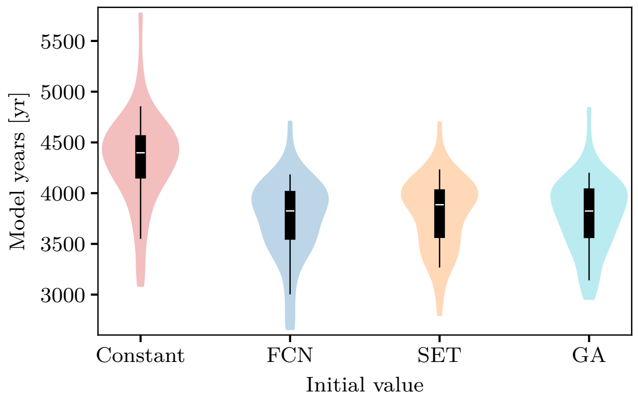

FIG. 5.1. *Comparison of the required model years of the spin-up with a spin-up tolerance* 10^{-4} *using the mass-corrected prediction of the three different ANNs as initial value.*

training was evaluated after applying the mass correction, whereas it was measured before if using (3.1) as post-processing step. Thus, the results with the mass-conserving layer inside the ANN were expected to be superior. However, this alternative did not lead to any improvement in our experiments. In fact, the error was even greater.

Using the mass-corrected prediction as initial concentration for the marine ecosystem model simulation shortened the computational costs. The results after a certain number of additional model steps were in good agreement with the reference solutions. This was the case for a test data set of 100 parameter vectors. Due to the mass conversation property of the model, the correct mass of the prediction was crucial. Any difference in the mass is always preserved during an additional spin-up calculation. Consequently, such mass difference resulted in an error. In contrast, starting from a tracer concentration with the correct mass, the concentration approached arbitrarily close to the steady annual cycle. The reached accuracy depended on the additionally performed model steps. For all three networks and nearly all parameter vectors in the test data set, this procedure led to a significant reduction of necessary iteration steps to reach the steady annual cycle. Thus, the ANNs can be used to generate a better initial value for a computation of the steady annual cycle and to save computing time. We achieved a reduction of the computational costs of about 15%. The violin plot [20] in Figure 5.1 indicates that the improvement in runtime was similar for the three presented ANNs.

Both the quality of the approximation and the reduction of the computing time were almost identical for the three ANNs. In analogy to [40], we could show that the ANN built with the SET algorithm gave results of the same quality as the FCN, despite the quadratic reduction of the number of parameters. We have not yet used the computational and memory benefits of the SET algorithm in our implementation. An efficient implementation (see, e.g., [33]) could be used to reduce the computational costs. Usage of the GA for hyperparameter optimization did not improve the results of the obtained ANN so far, whose results were comparable to those of the FCN. In this work, we used 10 generations with 30 individuals each, resulting in nearly 250 network training processes. These two parameters only determine the computational effort of the GA, since all other operations are negligible. However, since we optimized seven hyperparameters (see Table 3.1), even 250 combinations of them can only cover the hyperparameter space in a very sparse way. Therefore, we still think that it is promising to apply a GA to further improve the prediction results, always taking into account the necessary additional effort.

The reduction of the computational costs resulted from the predicted tracer concentration resembling already the structure of the concentration for the steady annual cycle in contrast to the constant initial concentration. While the annual variability affects exclusively concentrations in the upper ocean, concentration changes in deeper ocean requires thousands of model years; cf. [3, 6, 8]. For this reason, predictions reflecting well especially the concentrations in the deeper ocean should lead to a reduction of the required model years to compute the steady annual cycle, if they are taken as initial concentrations.

The computational costs for the computation of a steady annual cycle decisively affects the computational effort of a parameter optimization or sensitivity analysis (see, e.g., [27, 28, 29, 48]) of a global marine biogeochemical model, since several hundred computations of steady annual cycles are necessary for this. Our approximation approach of a marine ecosystem model could lower this computational effort.

The number of available training data limited our approach to approximate marine ecosystem models using ANNs. In typical applications of neural networks, using 1100 data sets for training is regarded to be a small number. On the other hand, the computation of steady states of the ecosystem model is quite expensive, taking thousands of years model time. Hence, this size of the training data set was reasonable and already high in the area of ocean and climate science. The small sample size of the training data may lead to overfitting and, consequently, to a larger generalization error. The use of larger neural networks with a higher number of parameters would increase this effect. In our approach, we used dropout as regularization to reduce the risk of overfitting. The SET algorithm reduced, in addition, the number of trainable parameters and, thus, the complexity of the neural network. Nevertheless, some complexity of the ANN was necessary to approximate the nonlinear function that maps the few model parameters to the concentration of the steady annual cycle.

We see four options to improve the results of our work: At first, the network structure could be designed differently. We mapped few parameters to a three-dimensional output, namely the spatially dependent tracer concentration of the steady state. In this work, we used the output as an one-dimensional vector. Preserving the three-dimensional distribution and using deconvolutional layers (that take into account the spatial neighborhood of the output) might lead to better results. Secondly, we could try to run the model for a couple of years until some basic spatial structure already has developed. We then could enhance the ANN's input (being only the few model parameters in this work) by this still unsteady state of the model. Naturally, this will substantially change the dimension of the input. As a consequence, also a different network structure would be necessary. As a third option, we could reduce the output dimension by performing a pre-processing; e.g., a principal component analysis. We then could try to predict the main components of the steady state instead of its full spatial resolution. Finally, the use of reinforcement learning, if the ANN is used in parameter optimization runs, sensitivity studies, or model assessments, would be possible.

In summary, the main points of this paper are the following:

- The predictions of the three ANNs were not a sufficient accurate approximation of the steady annual cycle of the marine ecosystem model.
- The mass-corrected predictions as initial value reduced the computational costs of the simulation of a marine ecosystem model to compute a steady annual cycle.
- Similar quality of the approximation and reduction of the computing time for the three ANNs.

Code and data availability. The code used to generate the data in this publication is available at `https://github.com/slawig/bgc-ann` and `https://metos3d.github.io/`. All used and generated data are available at `https://doi.org/10.5281/zenodo.4058319`.

Acknowledgments. The authors would like to thank Tamila Ghanbari-Ghazvini for the suggestion of the prime decomposition for the initial structure of the FCN and two anonymous reviewers for their helpful comments and suggestions.

REFERENCES

[1] A.-L. BARABÁSI AND R. ALBERT, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512.
[2] G. BELLEC, D. KAPPEL, W. MAASS, AND R. A. LEGENSTEIN, *Deep rewiring: training very sparse deep networks*, Preprint on arXiv:1711.05136, 2017. https://arxiv.org/abs/1711.05136
[3] E. BERNSEN, H. A. DIJKSTRA, AND F. W. WUBS, *A method to reduce the spin-up time of ocean models*, Ocean Model., 20 (2008), pp. 380–392.
[4] L. BOTTOU AND O. BOUSQUET, *The tradeoffs of large scale learning*, in Advances in Neural Information Processing Systems, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, eds., vol. 20 of NIPS'07, Curran Associates, 2008, pp. 161–168.
[5] H. J. BREMERMANN, M. ROGSON, AND S. SALAFI, *Global properties of evolution processes*, in Natural Automata and Useful Simulations, H. H. Pattee, ed., Spartan Books, Washington, 1966, pp. 3–42.
[6] K. BRYAN, *Accelerating the convergence to equilibrium of ocean-climate models*, J. Phys. Oceanogr., 14 (1984), pp. 666–673.
[7] D.-A. CLEVERT, T. UNTERTHINER, AND S. HOCHREITER, *Fast and accurate deep network learning by exponential linear units (ELUs)*, Preprint on arXiv:1511.07289, 2015. https://arxiv.org/abs/1511.07289
[8] G. DANABASOGLU, J. C. MCWILLIAMS, AND W. G. LARGE, *Approach to equilibrium in accelerated global oceanic models*, J. Clim., 9 (1996), pp. 1092–1110.
[9] T. DETTMERS AND L. ZETTLEMOYER, *Sparse networks from scratch: faster training without losing performance*, Preprint on arXiv:1907.04840, 2019. https://arxiv.org/abs/1907.04840
[10] A. E. EIBEN AND J. E. SMITH, *Introduction to Evolutionary Computing*, 2nd ed., Natural Computing Series, Springer, Berlin, 2015.
[11] P. ERDŐS AND A. RÉNYI, *On random graphs I*, Publ. Math. Debrecen, 6 (1959), pp. 290–297.
[12] ———, *On the evolution of random graphs*, Publ. Math. Instit. Hungar., 5 (1960), pp. 17–61.
[13] M. J. R. FASHAM, ed., *Ocean Biogeochemistry*, Global Change–The IGBP Series, Springer, Berlin, 2003.
[14] T. GALE, E. ELSEN, AND S. HOOKER, *The state of sparsity in deep neural networks*, Preprint on arXiv:1902.09574, 2019. https://arxiv.org/abs/1902.09574
[15] X. GLOROT, A. BORDES, AND Y. BENGIO, *Deep sparse rectifier neural networks*, in Proceedings of Machine Learning Research, Fort Lauderdale 2011, vol. 15, G. Gordon, D. Dunson, and M. Dudík, eds., JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.
[16] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep Learning*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, 2016.
[17] S. HAN, J. POOL, J. TRAN, AND W. J. DALLY, *Learning both weights and connections for efficient neural networks*, in Proceedings of the 28th International Conference on Neural Information Processing Systems, vol. 1 of NIPS'15, MIT Press, Cambridge, 2015, pp. 1135–1143.
[18] T. HASTIE, R. TIBSHIRANI, AND J. FRIEDMAN, *The Elements of Statistical Learning*, 2nd ed., Springer Series in Statistics, Springer, New York, 2009.
[19] M. HAUSKNECHT, J. LEHMAN, R. MIIKKULAINEN, AND P. STONE, *A neuroevolution approach to general Atari game playing*, IEEE Trans. Comput. Intel. AI, 6 (2014), pp. 355–366.
[20] J. L. HINTZE AND R. D. NELSON, *Violin plots: a box plot-density trace synergism*, The American Statistician, 52 (1998), pp. 181–184.
[21] J. H. HOLLAND, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.
[22] S. KHATIWALA, *A computational framework for simulation of biogeochemical tracers in the ocean*, Glob. Biogeochem. Cycles, 21 (2007), GB3001, 14 pages.
[23] S. KHATIWALA, M. VISBECK, AND M. A. CANE, *Accelerated simulation of passive tracers in ocean circulation models*, Ocean Model., 9 (2005), pp. 51–69.
[24] D. P. KINGMA AND J. BA, *Adam: a method for stochastic optimization*, Preprint on arXiv:1412.6980, 2014. https://arxiv.org/abs/1412.6980
[25] G. KLAMBAUER, T. UNTERTHINER, A. MAYR, AND S. HOCHREITER, *Self-normalizing neural networks*, in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds., vol. 30, Curran Associates, Inc., 2017, pp. 971–980.
[26] I. KRIEST, S. KHATIWALA, AND A. OSCHLIES, *Towards an assessment of simple global marine biogeochemical models of different complexity*, Prog. Oceanogr., 86 (2010), pp. 337–360.

[27] I. KRIEST, A. OSCHLIES, AND S. KHATIWALA, *Sensitivity analysis of simple global marine biogeochemical models*, Glob. Biogeochem. Cycles, 26 (2012), GB2029, 15 pages.
[28] E. Y. KWON AND F. PRIMEAU, *Optimization and sensitivity study of a biogeochemistry ocean model using an implicit solver and in situ phosphate data*, Glob. Biogeochem. Cycles, 20 (2006), GB4009, 13 pages.
[29] ———, *Optimization and sensitivity of a global biogeochemistry ocean model using combined in situ DIC, alkalinity, and phosphate data*, J. Geophys. Res. Oceans, 113 (2008), C08011, 23 pages.
[30] Y. LECUN, Y. BENGIO, AND G. HINTON, *Deep learning*, Nature, 521 (2015), pp. 436–444.
[31] Y. LECUN, J. S. DENKER, AND S. A. SOLLA, *Optimal brain damage*, in Advances in Neural Information Processing Systems 1990, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, eds., Morgan-Kaufmann, San Francisco, 1990, pp. 598–605.
[32] A. LEE, *pydoe: Design of Experiments for Python*, 2014, available at https://pythonhosted.org/pyDOE/index.html.
[33] S. LIU, D. C. MOCANU, A. R. R. MATAVALAM, Y. PEI, AND M. PECHENIZKIY, *Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware*, Neural Comput. Appl., 33 (2020), pp. 2589–2604.
[34] S. LIU, D. C. MOCANU, AND M. PECHENIZKIY, *Intrinsically sparse long short-term memory networks*, Preprint on arXiv:1901.09208, 2019. https://arxiv.org/abs/1901.09208
[35] E. MAIER-REIMER, I. KRIEST, J. SEGSCHNEIDER, AND P. WETZEL, *The HAMburg Ocean Carbon Cycle Model HAMOCC5.1—Technical Description Release 1.1*, 2005.
[36] M. D. MCKAY, R. J. BECKMAN, AND W. J. CONOVER, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, Technometrics, 21 (1979), pp. 239–245.
[37] T. MICONI, *Neural networks with differentiable structure*, Preprint on arXiv:1606.06216, 2016. https://arxiv.org/abs/1606.06216
[38] M. MITCHELL, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, 1996.
[39] D. C. MOCANU, E. MOCANU, P. H. NGUYEN, M. GIBESCU, AND A. LIOTTA, *A topological insight into restricted Boltzmann machines*, Mach. Learn., 104 (2016), pp. 243–270.
[40] D. C. MOCANU, E. MOCANU, P. STONE, P. H. NGUYEN, M. GIBESCU, AND A. LIOTTA, *Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science*, Nat. Commun., 9 (2018), pp. 2383–2394.
[41] A. MOREL, *Optical modeling of the upper ocean in relation to its biogenous matter content (case I waters)*, J. Geophys. Res. Oceans, 93 (1988), pp. 10749–10768.
[42] H. MÜHLENBEIN AND D. SCHLIERKAMP-VOOSEN, *Predictive models for the breeder genetic algorithm*, Evol. Comput., 1 (1993), pp. 25–49.
[43] V. NAIR AND G. E. HINTON, *Rectified linear units improve restricted Boltzmann machines*, in Proceedings of the 27th International Conference on Machine Learning, J. Fürnkranz and T. Joachims, eds., Haifa, Israel, 2010, Omnipress, Madison, 2010, pp. 807–814.
[44] A. OSCHLIES, *On the use of data assimilation in biogeochemical modelling*, in Ocean Weather Forecasting, E. P. Chassignet and J. Verron, eds., Springer, Dordrecht, 2006, pp. 525–547.
[45] A. OSCHLIES AND V. GARÇON, *An eddy-permitting coupled physical-biological model of the North Atlantic 1. Sensitivity to advection numerics and mixed layer physics*, Glob. Biogeochem. Cycles, 13 (1999), pp. 135–160.
[46] L. PESSOA, *Understanding brain networks and brain organization*, Phys. Life Rev., 11 (2014), pp. 400–435.
[47] J. PIWONSKI AND T. SLAWIG, *Metos3D: the marine ecosystem toolkit for optimization and simulation in 3-d – part 1: Simulation package v0.3.2*, Geosci. Model Dev., 9 (2016), pp. 3729–3750.
[48] M. PRIESS, J. PIWONSKI, S. KOZIEL, A. OSCHLIES, AND T. SLAWIG, *Accelerated parameter identification in a 3D marine biogeochemical model using surrogate-based optimization*, Ocean Model., 68 (2013), pp. 22–36.
[49] D. RIVERO, J. DORADO, E. FERNÁNDEZ-BLANCO, AND A. PAZOS, *A genetic algorithm for ann design, training and simplification*, in Bio-Inspired Systems: Computational and Ambient Intelligence, J. Cabestany, F. Sandoval, A. Prieto, and J. M. Corchado, eds., Springer, Berlin, 2009, pp. 391–398.
[50] D. E. RUMELHART, G. E. HINTON, AND R. J. WILLIAMS, *Learning representations by back-propagating errors*, Nature, 323 (1986), pp. 533–536.
[51] J. L. SARMIENTO AND N. GRUBER, *Ocean Biogeochemical Dynamics*, Princeton University Press, Princeton, 2006.
[52] N. SRIVASTAVA, G. HINTON, A. KRIZHEVSKY, I. SUTSKEVER, AND R. SALAKHUTDINOV, *Dropout: a simple way to prevent neural networks from overfitting*, J. Mach. Learn. Res., 15 (2014), pp. 1929–1958.
[53] K. O. STANLEY AND R. MIIKKULAINEN, *Evolving neural networks through augmenting topologies*, Evol. Comput., 10 (2002), pp. 99–127.
[54] S. H. STROGATZ, *Exploring complex networks*, Nature, 410 (2001), pp. 268–276.
[55] L. D. TALLEY, G. L. PICKARD, W. J. EMERY, AND J. H. SWIFT, *Descriptive Physical Oceanography*, 6th ed., Elsevier Science, London, 2011.
[56] A. S. WEIGEND, D. E. RUMELHART, AND B. A. HUBERMAN, *Generalization by weight-elimination with*

application to forecasting, in Advances in Neural Information Processing Systems 1990, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, eds., Morgan-Kaufmann, San Francisco, 1990, pp. 875–882.

[57] H. ZHU AND Y. JIN, *Multi-objective evolutionary federated learning*, IEEE Trans. Neural Netw. Learn. Syst., 31 (2020), pp. 1310–1322.