



OPEN

Application of physics encoded neural networks to improve predictability of properties of complex multi-scale systems

Marcel B. J. Meinders^{1,2✉}, Jack Yang^{1,3} & Erik van der Linden^{1,3}

Predicting physical properties of complex multi-scale systems is a common challenge and demands analysis of various temporal and spatial scales. However, physics alone is often not sufficient due to lack of knowledge on certain details of the system. With sufficient data, however, machine learning techniques may aid. If data are yet relatively cumbersome to obtain, hybrid methods may come to the rescue. We focus in this report on using various types of neural networks (NN) including NN's into which physics information is encoded (PeNN's) and also studied effects of NN's hyperparameters. We apply the networks to predict the viscosity of an emulsion as a function of shear rate. We show that using various network performance metrics as the mean squared error and the coefficient of determination (R^2) that the PeNN's always perform better than the NN's, as also confirmed by a Friedman test with a p -value smaller than 0.0002. The PeNN's capture extrapolation and interpolation very well, contrary to the NN's. In addition, we have found that the NN's hyperparameters including network complexity and optimization methods do not have any effect on the above conclusions. We suggest that encoding NN's with any disciplinary system based information yields promise to better predict properties of complex systems than NN's alone, which will be in particular advantageous for small numbers of data. Such encoding would also be scalable, allowing different properties to be combined, without repetitive training of the NN's.

Keywords Neural networks, Machine learning, Physics encoded neural networks, Multi-scale modeling, Complex systems

Complex multi-scale systems are systems that are continuously changing, in an interdependent, self-organizing, and adaptive manner. An example of a real-life complex multi-scale system is the agri-food system¹. A systematic way to predict the response of a complex multi-scale system on an intervention will involve analyses on its multiple scales, including the adaptive dynamics at all scales. This has been coined a complex systems approach¹. Such approach has been argued to thrive from integrating various methodologies, including scaling, multi-scale modeling, machine learning, including graphical and evolutionary algorithms¹ and specific references therein). Also computational techniques have been suggested for the same application area by Perrot et al.². However, in both articles, no specific methodology was articulated on how to specifically integrate multi-scale modeling and machine learning techniques in a quantitative fashion.

Conducting a multi-scale analysis is a complex endeavor since multiple connections can exist between different scales. Furthermore, such properties evolve, and the according dynamics usually cannot be covered by formal analytic descriptions. This according complexity makes the use of available mechanistic models alone difficult. A common strategy to relate scales with each other is to introduce intermediate length scales, which contain integrated information of the scales below, which can subsequently be incorporated into descriptions for the scale above the intermediate scale. An example is the so-called meso- or micro-structural scale in between the molecular and macroscopic scale. This strategy still has its challenges, attributed to the ill-reliability of microscale models, the difficulty in simulating the micro-scale properties accurately, and the often intricate “entanglement” between micro-structural and macroscopic scale properties³.

¹Wageningen University and Research Centre, Wageningen, The Netherlands. ²Wageningen Food and Biobased Research, Wageningen, The Netherlands. ³Wageningen University, Wageningen, The Netherlands. ✉email: marcel.meinders@wur.nl

In order to accommodate these challenges described in the previous paragraph, machine learning (ML) using neural networks (NNs) can be an asset when sufficient data are available. In practice the amount of data available is not sufficient. In this case, using physical information may help. In general, combining physics-based multi-scale modeling with ML techniques may cleverly solve two problems at the same time. Physics may provide the structuring of ML techniques, and turning correlative relations into casual ones. Attempts to combine ML with physics-based modeling for dynamics of lake temperature^{4–6}, and phosphorus concentration⁷ have shown that one can obtain better predictions with a smaller number of data and for scenarios that are distinct from the training scenario used in the ML algorithm itself. A recent review on physics-informed machine learning can be found in⁸ and with application focus on life sciences in⁹. In these reviews, the challenges and possible routes forward to model spatial-temporal evolution combining physics and ML are clearly addressed. To capture the spatio-temporal dynamics, one can use partial differential equations to express physics-based conservation laws, where such conservation laws can be constructed from constitutive laws that represent the local behaviour, in combination with using ordinary differential equations and their spatial derivatives. This has been illustrated in more detail for rheology-informed neural networks (RhiNNs) by Mahmoudabadbozchelou and Jamali¹⁰. Interestingly, Sadaat *et al*¹¹ report the use of a platform of possible constitutive models to have the RhiNN pick from in order to optimize its predictions for more complex fluid behavior. These constitutive models need not represent the entire physics of the problem in all its details. Interestingly, parts of constitutive models can be captured by means of applying scattering under flow, as demonstrated by Young *et al* while using Small Angle X-ray scattering on dilute rod suspensions¹². In trying to take into account spatio-temporal non-linear features, Dabiri *et al*¹³ report the use of fractional derivatives to incorporate into the NN models. Fractional derivatives are used to represent the presence of memory effects, which may be uncovered by introducing hidden variables that describe local effects, as addressed in Weinan *et al*³. In that work, an exciting example of introducing temporal (dynamical) information has been addressed in the form of so-called “recurrent neural networks”. The neural networks are machine learning models for time series. These models use hidden variables, making the relationships, as expressed in the models, local. If no hidden variables are being used, one effectively introduces memory effects³.

There are several ways at our disposal to add physical information into a neural network. For example, one can ascribe physics information to nodes in the network¹⁴. Alternatively, one can add physical information regarding symmetries that need to be obeyed. Another option could be the use of physics-based model data as input to AI models. For a survey on recent progress in various fields, the reader is referred to Willard *et al*¹⁴. It is noted that humans can develop physics-based architectures of neural networks, but this can be automated as well (¹⁴ references 13,73, 115). A concrete set of examples of improvement of neural network performance, which at the same time preserves the correctness of the physics, has been recently published by Takeishi and Kalousis¹⁵.

In regards to the agri-food area, in particular in applying ideas on combining multi-scale modeling with ML directly, i.e. without the need to reprogram the neural networks ML part, a review of Peng *et al*¹⁶ is worthwhile to mention. Works with more direct embedding in food science that address the combination of physics information and ML, without reprogramming the neural networks that underlie the ML are, for example, found elsewhere¹⁷. In this same area already some reviews can be found^{18,19}.

We note that the above addresses adding disciplinary information to the input or output to an AI methodology. This is referred as physical informed neural networks (PiNNs) according to Faroughi *et al*⁸. On the other hand, physics knowledge can also be build into the network itself, referred to as physics encoded neural networks (PeNN's)⁸. It is in this spirit that we like to situate our work on PeNN's.

To our knowledge, encoding of physics information into NN's in the area of multi-layer complex systems as described has not been addressed. Therefore, in the current article, we quantify the effects of including physics information in the architecture of NNs. We investigate uncertainty in prediction as a function of training set size, and effects on uncertainty/errors in inter- and extrapolating beyond a training set. We specifically look into the problem of protein stabilized oil droplets aggregating into clusters, and how the cluster size distribution in turn will determine the shear viscosity versus shear strain. The problem becomes complex as the flow influences the aggregation and vice versa and as one adds more experimental parameters that are known to influence the viscosity, such as the pH, protein type etc. The example lends itself to demonstrating the improvement of the predictability of a NN by integrating physics into it, i.e. transferring an NN to a PeNN.

Flow of complex food fluids

Many complex foods are in fact dispersions, which are composed of fluid or solid particles dispersed in a fluid. Their flow behaviour under deformation is expressed by the viscosity. Texture, consumer acceptance, and processing conditions are strongly related to the viscosity. To optimize formulation, it is important to get insights in the dispersion's viscosity and its controlling factors. Similarly, this also holds for non-food materials like e.g. paints, coatings, and cosmetics. Due to the complex nature of the dispersions, the viscosity depends on the rate of deformation. For our purpose, we focus here on shear deformation.

The shear rate-dependent viscosity depends on the structure of the dispersions and the interactions between the dispersed particles. The interactions control the assembly of (primary) particles into clusters. During deformation of the complex fluid, formation and breakdown of these clusters depend on time, shear rate, particle concentrations, cluster sizes, and inter-particle and inter-cluster interactions and temperature. Many studies have been published to describe the shear-rate dependent viscosity of complex dispersions^{20–23}. Models to describe this kind of shear rate-dependent viscosity should contain the structural dynamics. One such model is the constitutive model by Quemada and coworkers^{24–28}, which describes the rheology of complex colloidal systems in a large range of volume fractions using key physical parameters. Because the model fits very well with viscosities of food dispersions^{20,21,23,29} (see also Fig. 1), we used the Quemada model to study the predictive power of

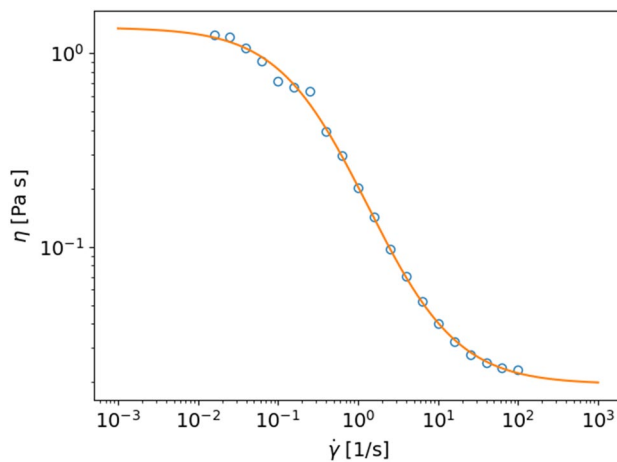


Figure 1. Example of a flow curve of an oil-in-water emulsion and Quemada-model-fit.

different NNs, and hybrid neural networks containing physics, for predicting the shear rate dependent viscosity of complex dispersions.

Quemada model

The Quemada model describes the viscosity of complex fluids in terms of an effective volume fraction, which depends on the properties of the primary particles, including diffusion coefficients, interaction parameters, packing fraction of particles in a cluster, and shear rate. Here, we will focus on the semi-stationary regime, where the system is in equilibrium at a certain shear rate. The starting point of the model is the relation between the viscosity of the fluid η and the effective volume fraction ϕ_e . This relation is given by^{24–28}

$$\eta = \eta_f \left(1 - \frac{\phi_e}{\phi_m}\right)^{-2} \quad (1)$$

with η_f the viscosity of the continuous phase, and ϕ_m the maximum volume fraction. The effective volume fraction is given by^{24–28}

$$\phi_e = \phi_{pA}/\varphi + (\phi_p - \phi_{pA}) = \phi_p(1 + CS) \quad (2)$$

with ϕ_p the volume fraction of the primary particles, $C = 1/\varphi - 1$, a compactness factor where φ is the volume of the particles in a cluster divided by the volume of that cluster. S is a structural parameter defined as the ratio between the volume fraction of primary particles in the cluster ϕ_{pA} and the total volume fraction of primary particles ϕ_p ^{24–28}

$$S = \frac{\phi_{pA}}{\phi_p} \quad (3)$$

The structural parameter follows a certain kinetic reaction scheme, which, in its basic form, reads^{24–28}

$$\frac{dS}{dt} = \kappa_D(S_0 - S) - \kappa_h(S - S_\infty) \quad (4)$$

with κ_D and κ_h characteristic relaxation rates of Brownian (diffusion) and hydrodynamic (shear stress) forces. S_0 and S_∞ correspond to the value of the structural parameter at zero and infinite shear rate. Additional terms related to particle interactions can be added²⁸. The steady-state value of the structural parameter at a certain shear-rate $\dot{\gamma}$ is then given by^{24–28}

$$S = \frac{S_0 + \theta S_\infty}{1 + \theta} \quad (5)$$

with

$$\theta = \frac{\kappa_h}{\kappa_D} = \frac{\dot{\gamma}}{a^2/D_p} = Pe = \frac{6\pi\eta_f a^3 \dot{\gamma}}{k_B T} \quad (6)$$

and a is the size of the primary particle, D_p the diffusion coefficient of the primary particles, k_B Boltzmann's constant, T temperature, and Pe the Péclet number. The above equations give the fluid's viscosity in the stationary state as a function of the key parameters^{24–28}

$$\eta = \eta(\dot{\gamma}, \phi_p, S_0, S_\infty, C, \eta_f, a, T) \quad (7)$$

Figure 1 shows an example of a measured flow curve of a pea protein stabilized oil-in-water emulsion, with an oil volume fraction of 0.5. The figure also shows the fit with the Quemada model with $\dot{\gamma}_c = \frac{6\pi\eta_f a^3}{k_B T} = 4.7$ (corresponding to an average droplet size of $a = 1 \mu\text{m}$), $CS_0 = 0.25$ (indicating that less than about 25% of the oil droplets are flocculated at zero shear rate) and $CS_\infty = 0$ (indicating that all clusters broke up at a high shear rate).

Neural networks and physics-encoded neural networks

To investigate the predictive power of NNs and PeNNs, shear rate dependent viscosity data was generated with the Quemada model and used to train and test NNs and PeNNs with different architectures. In order to align with physical experiments, we aim to predict $y = \eta$ from a set of input parameters $\{x_i\}$. In an experimental setting, η is measured as a function of the shear rate $\dot{\gamma}$, so $\dot{\gamma} \in \{x_i\}$. Various other input characteristics of the dispersion can be measured or are known. For example, when preparing a sample, the volume fraction ϕ_p of the primary particles is known, like the volume fraction of oil in an emulsion. Other parameters can be measured and estimated using various different, often indirect, techniques, like e.g. the particle and cluster size distributions from light scattering techniques and inter-particle forces from DLVO theory.

The NNs and PeNNs were built using Python (version 3.9.16) in combination with the open source platform for machine learning TensorFlow (version 2.11.0), which uses Keras (version 2.11.0) as the high-level API. Data was generated and used to train various NNs and PeNNs with different architectures. The shear rate $\dot{\gamma}$ and viscosity η can vary over several orders of magnitude. To train, validate, and test the NNs and PeNNs, we used $\log \dot{\gamma}$ and $\log \eta$, so that all input and output parameters are in the order of 1 to 10, and no normalization of the input and output data is needed. We performed some extra calculations using batch normalization within the neural network layers. However, it turned out that the performance of the networks did not improve.

Different architectures of NN's were investigated, varying in number of hidden layers and number of neurons per layer. The rectified linear (ReLU) activation function was used in all layers except for the final layer, for which we used the linear activation function because we are dealing here with a regression problem^{30–32} and not a classification problem. As default we used the Adam optimization algorithm to train the NN's, which is generally accepted to be one of the most efficient algorithms in machine learning. We used a learning rate of 10^{-3} with a decay of $5 \cdot 10^{-6}$, mean squared error (mse) as loss function or performance metric, and 20% of the learning data for validation. The training was stopped when the loss function of the validation set has stopped improving in 500 epochs using the build-in TensorFlow callback function Early-Stopping.

We also performed hyper parameter tuning by investigating different NN's with different complexities, optimization routines like Nadam, AdamW, and Lion, learning rates, and activation functions.

We investigated two cases to compare. In case one, we predict the viscosity as a function of two input parameters, while in case two, we predict the viscosity as a function of four input parameters.

Performance of the NNs and PeNNs was assessed using the performance metrics coefficient of determination R^2 and the mean square error (mse) between predicted and ground truth viscosity values of the test set, for unbiased evaluation. To check to what extent the performance of the studied networks differ, a graphical comparison of the metrics was performed. We also included a Friedman test to quantify the performance differences, similar as done by Zamri *et al*³³. In addition, we compared the predicted and ground truth viscosities graphically.

Case 1: networks with 2 input parameters and 1 output parameter

For case one, two different NNs were studied (see Fig. 2). One with an architecture consisting of two hidden dense layers. The first layer has $n_1 = 32$ neurons and is densely connected to the $n_{in} = 2$ input nodes, while the second layer has $n_2 = 8$ neurons. Another NN studied has an architecture consisting of three hidden dense layers. The first, second and third layer has $n_1 = 128$, $n_2 = 32$ and $n_3 = 8$ neurons, respectively. For both NNs, the first layer is densely connected to the $n_{in} = 2$ input nodes, while the last layer is densely connected with the output layer having $n_{out} = 1$ neuron.

The PeNN consists of 3 layers, each corresponding to a physical quantity, in other words, the activation functions are completely physics-based. In this sense, this PeNN is actually totally dominated by physics, to illustrate the importance of physics in a NN. The first layer corresponds to the structure parameter S . It has one input $x = \log \dot{\gamma}$ and activation function $S_{act} = 1/(10^x w + 1)$. Here, w is a trainable parameter and should correspond to $\frac{6\pi\eta_f a^3}{k_B T}$ as can be simply derived from Eqs. 5 and 6. The second layer corresponds to the effective volume fraction ϕ_e having two inputs, being the output of the S -layer and the input ϕ_p . The activation of the ϕ_e -layer is

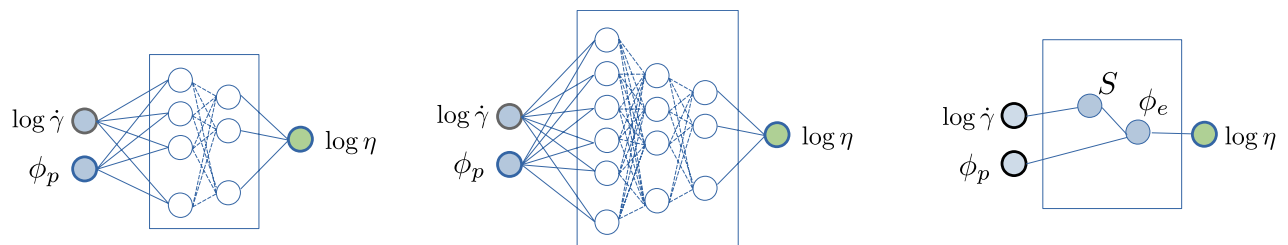


Figure 2. Schematic architecture of the NN (left, middle) and PeNN (right) configuration.

$\phi_{e,act} = x_1(x_2w + 1)$, with input $\mathbf{x} = [x_1, x_2]$ and w a trainable parameter corresponding to C (equation 2). The third layer corresponds to the viscosity with activation function $\eta_{act} = -2 \log(1 - xw) + b$, where w and b are trainable parameters, corresponding to $1/\phi_m$ and $\log \eta_f$, respectively, and x is the input equal to the output of the effective-volume-fraction-layer.

In case one, only the shear rate $\dot{\gamma}$ and particle volume fraction ϕ_p were varied and taken as input parameters for the constitutive model to generate the steady-state fluid viscosity η . Other parameters were taken constant, being $T = 293$ K, $a = 5$ nm, $C = 2$, $S_0 = 1$, $S_\infty = 0$, and $\eta_f = 10^{-3}$ Pa.s.

Case 2: networks with 4 input parameters and 1 output parameter

In case 2, also S_0 and S_∞ were varied to generate flow curve data to train NNs and PeNNs with architectures, as depicted in Fig. 3. These parameters correspond to the structure parameter at zero and infinite shear rate, respectively. In this sense, this PeNN is not totally dominated by physics. In general, parameters are difficult to assess as they can be related to handling history, inter-particle forces, amount of protein denaturation, pH, salt concentration, etc. Here, we took two representative input parameters p_1 and p_2 between 0 and 1.

Similar as above, two different NNs were studied with architectures consisting of two and three dense hidden layers consisting of 32-8 and 128-32-8 neurons. For both, the first layer is densely connected to the $n_{in} = 4$ input nodes, while the last layer is densely connected with the output layer having $n_{out} = 1$ neuron.

The PeNN consist of 6 layers, of which the first 3 are a dense connected NN with outputs that should mimic the structure factor at low and high shear rate S_0 and S_∞ , respectively. The following and last 3 layers correspond each to the physical quantities, as explained above. The only exception is that the S-layer now has three inputs $\mathbf{x} = [x_1 x_2 x_3]$ corresponding to $[S_0 S_\infty \log \dot{\gamma}]$ and physics-based activation function $S_{act} = (x_1 + 10^{x_3} w x_2) / (10^{x_3} w + 1)$. Here, again w is a trainable parameter and should correspond to $\frac{6\pi \eta_f a^3}{k_B T}$. The last layers are the same as described above.

Data sets

For both cases 1 and 2, various data sets were generated and used to train and test the performance of the NNs and PeNNs, varying in number of input parameters (as discussed above) as well as varying in number of data points per input parameter. The generated data was split into a training set (75%, randomly chosen) used to train NNs and PeNNs, and a test-holdout set (25%) to test the performance after training. During training, 20% of the set was used for validation.

It is noted that in rheology experiments, in general the number of data points for the shear rate $\dot{\gamma}$ is much larger than that for a parameter like the volume fraction of the primary particles ϕ_p . This is because it is rather simple to obtain 100 or more data points ($\eta(\dot{\gamma})$) in a viscosity measurement for just one sample (with a certain ϕ_p). For better comparison with real life experiments we therefore generated the data sets in a similar way: for each of n different ϕ_p 's (with $\phi_p \in \{\phi_1 \dots \phi_n\}$), N different $\log \dot{\gamma}$ (with $\dot{\gamma} \in \{\dot{\gamma}_1 \dots \dot{\gamma}_N\}$) were chosen as input parameters, with $N \gg n$. This yields $n \times N$ data triplets ($\eta(\phi_p, \dot{\gamma})$) for case 1, and $n \times N \times n_{p1} \times n_{p2}$ data quintets ($\eta(\phi_p, \dot{\gamma}, p_1, p_2)$) for case 2. Here n_{p1} and n_{p2} are the number of input parameters p_1 and p_2 respectively. Here we choose $n_{p1} = n_{p2} = 2$. The generated data set was randomly split into a training set (75%, thus $0.75nN$ and $0.75nNn_{p1}n_{p2}$ data points for case 1 and 2, respectively) and a test-holdout set ($0.25nN$ and $0.25nNn_{p1}n_{p2}$ data points for case 1 and 2, respectively). In general, the test-holdout set is used to check the performance of a NN to unseen data. Although the NN did not see the data triplets ($\eta(\phi_p, \dot{\gamma})$) of the test set, it did see numerous data with $\phi_p \in \{\phi_1 \dots \phi_n\}$. In order to check to what extent the NNs and PeNNs can also generalize to unseen volume fractions, thus how they perform for $\phi_p \notin \{\phi_1 \dots \phi_n\}$, another set, referred to as the test set, was created by choosing randomly $n_{tst} = 10^4$ input parameters ($\phi_p, \log \dot{\gamma}$) with ϕ_p between $\phi_{p,min} = 0.01$ and $\phi_{p,max} = 0.2$ and $\log \dot{\gamma}$ between $\log \dot{\gamma}_{min} = -5$ and $\log \dot{\gamma}_{max} = 3$. This test set (10^4 data points for case 1 and 2) was used for an unbiased comparison of the performance of the networks.

The pseudo code of the algorithms used can be found in the appendix.

Results

Case 1: networks with 2 input parameters and 1 output parameter

Figure 4 shows an example of the results of a NN for case 1, with two input nodes (ϕ_p and $\log \dot{\gamma}$), three hidden layers with 128, 32, and 8 neurons, respectively, and one output layer $\log \eta$. This NN was trained and tested using a data set generated from $n = 3$ different ϕ_p and for each ϕ_p $N = 300$ different $\dot{\gamma}$. The top-left panel shows the loss function as a function of the number of epochs. The top-right panel shows the predicted output as a function of

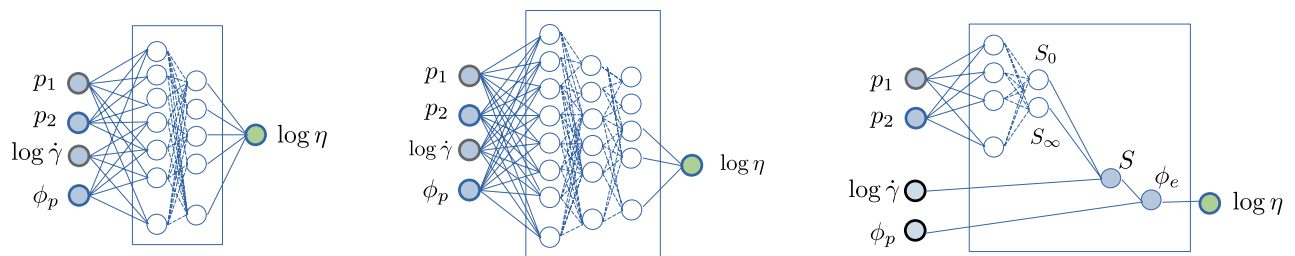


Figure 3. Schematic architecture of the NN (left, middle) and PeNN (right) configuration.

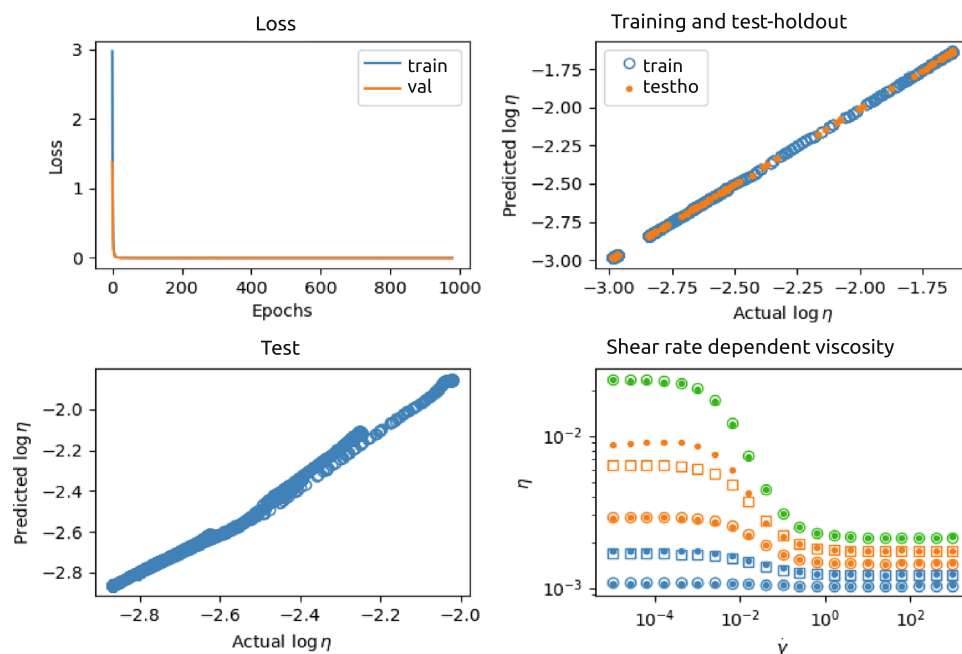


Figure 4. Results of the NN 2-128-32-8-1 neural network (see Fig. 3). This NN was trained and tested using a data set generated from $n = 3$ different ϕ_p and for each ϕ_p $N = 300$ different $\dot{\gamma}$. Top-left: loss as a function of number of epochs; Top-right: predicted versus actual values of the training (blue) and test-holdout (orange) set; Bottom-left: predicted versus ground truth of the test set containing unseen ϕ_p ; Bottom-right: examples of ground truth (actual) shear rate dependent viscosity (\circ (seen ϕ_p) and \square (unseen ϕ_p)) and predicted shear rate dependent viscosity (\bullet). The colors indicate different ϕ_p .

the actual (ground truth) output, for the training set (675 data points) as well as the test-holdout set (225 data points). The predicted values and actual values of the validation set (10^4 data points), also containing ϕ_p -values not seen by the NN, are plotted against each other in the left-bottom panel. In addition, the right-bottom panel of the figure shows the predicted and ground truth viscosity as function of shear rate and for different ϕ_p , seen, as well as unseen during training.

Figure 5 shows an example of the results of the PeNN for case 1, with two input nodes (ϕ_p and $\log \dot{\gamma}$), two layers completely physics-encoded, and one output layer $\log \eta$. The top-left panel shows the loss function as a function of the number of epochs. The top-right panel shows the predicted output as a function of the actual (ground truth) output, for the training set (675 datapoints) as well as the test-holdout set (225 data points). The predicted values and actual values of the test set (10^4 data points) containing ϕ_p 's not seen by the PeNN are plotted against each other in the left-bottom panel. In addition, the right-bottom panel of the figure shows the predicted and ground truth (actual) viscosity as function of shear rate and for different ϕ_p , seen, as well as unseen during training.

We conclude that in this case 1, replacing neural nodes by physics does improve NNs significantly. The viscosity as function of shear rate and primary particle volume fraction are predicted very well, also for values of input parameters which the model has not been trained for. Thus interpolation and extrapolation are captured well for the PeNNs.

Case 2: networks with 4 input parameters and 1 output parameter

Figure 6 shows an example of the results of a NN for case 2, with four input nodes (ϕ_p , $\log \dot{\gamma}$, p_1 , and p_2), two hidden layers with 32 and 8 neurons, respectively, and one output layer $\log \eta$. This NN was trained and tested using a data set generated from $n = 3$ different ϕ_p and for each ϕ_p $N = 300$ different $\dot{\gamma}$ and $n_{p1} = n_{p2} = 2$ different values for p_1 and p_2 respectively. The top-left panel shows the loss function as a function of the number of epochs. The top-right panel shows the predicted output as a function of the actual (ground truth) output, for the training set (2700 data points) as well as the test-holdout set (900 data points). The predicted values and actual values of the test set (10^4 data points) are plotted against each other in the left-bottom panel. In addition, predicted and actual curves with different ϕ_p , seen, as well as unseen during training, as function of $\dot{\gamma}$ are shown in the right-bottom panel.

Figure 7 shows an example of the results of the PeNN for case 2, with four input nodes (ϕ_p , $\log \dot{\gamma}$, p_1 , and p_2), 2 dense NN-layers and 3 layers completely physics-encoded, of which the last one is the output layer $\log \eta$. The top-left panel shows the loss function as a function of the number of epochs. The top-right panel shows the predicted output as a function of the actual (ground truth) output, for the training (2700 data points) and test-holdout set (900 data points), while the left-bottom panel shows this for the test set (10^4 data points). The

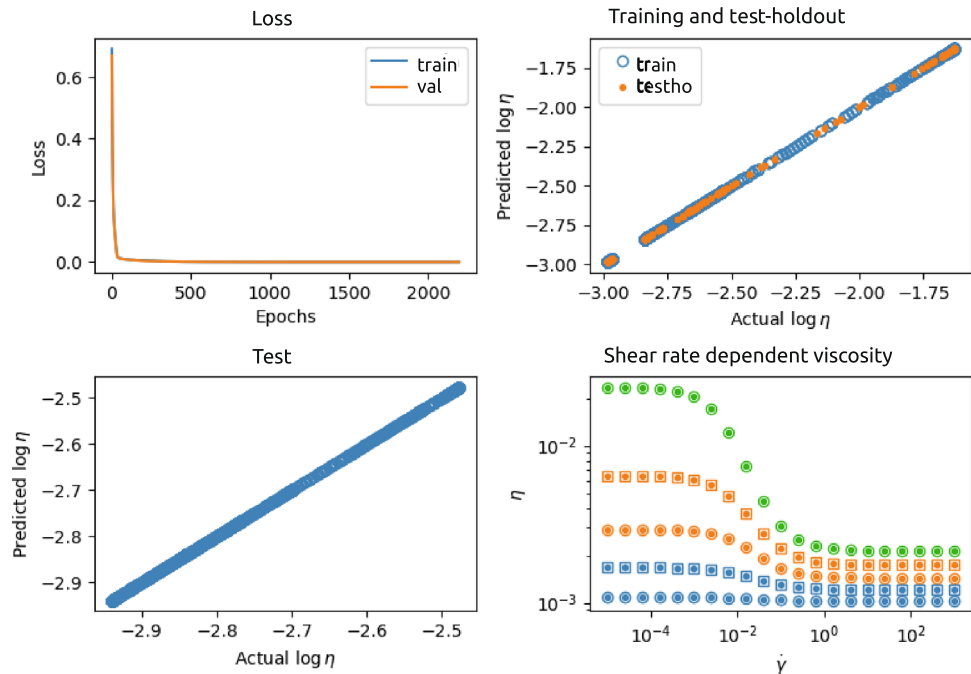


Figure 5. Results of the PeNN 2-1-1-1 [physical-encoded neural network (see Fig. 3)]. This NN was trained and tested using a data set generated from $n = 3$ different ϕ_p and for each ϕ_p $N = 300$ different $\dot{\gamma}$. Top-left: loss as a function of number of epochs; Top-right predicted versus actual values of the training (blue) and test-holdout (orange) set; Bottom-left: predicted versus actual values of the test set containing unseen ϕ_p ; Bottom-right: examples of actual shear rate dependent viscosity (\circ (seen ϕ_p) and \square (unseen ϕ_p)) and predicted shear rate dependent viscosity (\bullet). The colors indicate different ϕ_p .

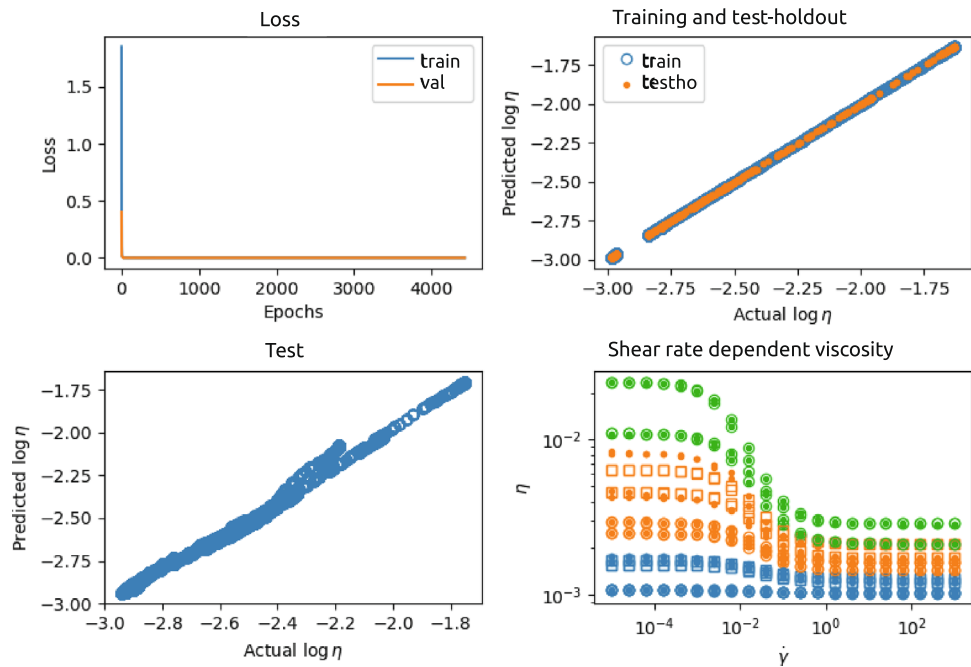


Figure 6. Results of the NN 4-32-8-1 neural network (see Fig. 3). The NN was trained and tested using a data set generated from $n = 3$ different ϕ_p and for each ϕ_p $N = 300$ different $\dot{\gamma}$. Top-left: loss as a function of number of epochs; Top-right predicted versus actual values of the training and test-holdout set; Bottom-left: predicted versus actual values of the test set; Bottom-right: examples of actual shear rate dependent viscosity (\circ (seen ϕ_p) and \square (unseen ϕ_p)) and predicted shear rate dependent viscosity (\bullet). The colors indicate different ϕ_p .

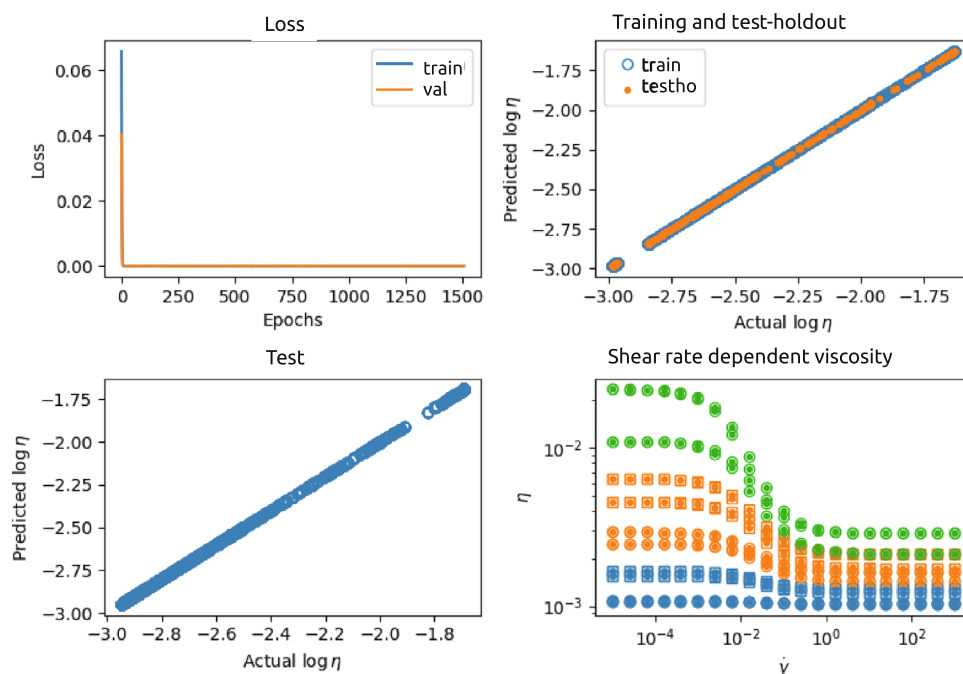


Figure 7. Results of the PeNN 4-6-2-1-1-1 physical-encoded neural network (see Fig. 3). The PeNN was trained and tested using a data set generated from $n = 3$ different ϕ_p and for each ϕ_p $N = 300$ different $\dot{\gamma}$. Top-left: loss as a function of number of epochs; Top-right: predicted versus actual values of the training (orange) and test-holdout (blue) set; Bottom-left: predicted versus actual values of the test set ϕ_p ; Bottom-right: examples of actual (○) and predicted (●) flow curves (colors indicate ϕ_p).

right-bottom panel of the figure shows the predicted and ground truth viscosity as function of shear rate and for different ϕ_p , seen, as well as unseen during training.

We conclude also for case 2 that replacing neural nodes with physics does improve the NNs significantly. The viscosity as function of shear rate and primary particle volume fraction are predicted very well, also for values of input parameters which the model has not been trained for. Thus interpolation and extrapolation are captured well for the PeNNs.

The observation that PeNNs perform better than NNs is also nicely illustrated in Fig. 8, showing the mean squared error (mse) for the NNs and PeNN for the test set. The figure shows that the mse of the PeNN is always at least about a factor 100 to 1000 smaller than that of the NNs. This indicates that the PeNN's performs significantly better than the NN's. This is also confirmed by the Friedman test performed on the data displayed in Fig. 8. The Friedman test resulted in rejection of the H_0 hypotheses (there is no performance differences between the models) with a p-value smaller than 0.0002. A Wilcoxon signed-rank test shows that the performances of all NNs are equal (not significantly different, with p-values larger than 0.6) while the PeNN performs significantly better than the NNs, with according p-values smaller than 0.002. We refer to Supplementary Information for more details.

We also studied the effect of NN's hyperparameters (different architectures with different complexities, optimization algorithms, learning rates) as well as different performance metrics and have found that these have no effect on our main conclusion about the differences between performance of NN's and PeNN's that PeNN's outperform NN's. We refer to Supplementary Information for more details.

Discussion, conclusions, and perspective

We have illustrated for a complex multi-scale problem how physics information can be encoded into a neural network and how that leads to improvement of the predictability and robustness of the network. Similar results have been obtained in another field giving better predictions and with a smaller number of data⁴⁻⁷.

Our work uses physics to improve a NN model. The incorporation of physics into the architecture of the NN is such that the resulting architecture of the NN has the same architecture of the physics-based hierarchical architecture. It is this architectural hierarchical feature that is important in realizing the so-called Physics-encoded NN (PeNN). The term physics-encoded has some specific implications, which are clearly different from the features of other types of hybrid NNs, as also discussed by Faroughi *et al.*⁸. Advantages of PeNNs above other forms of

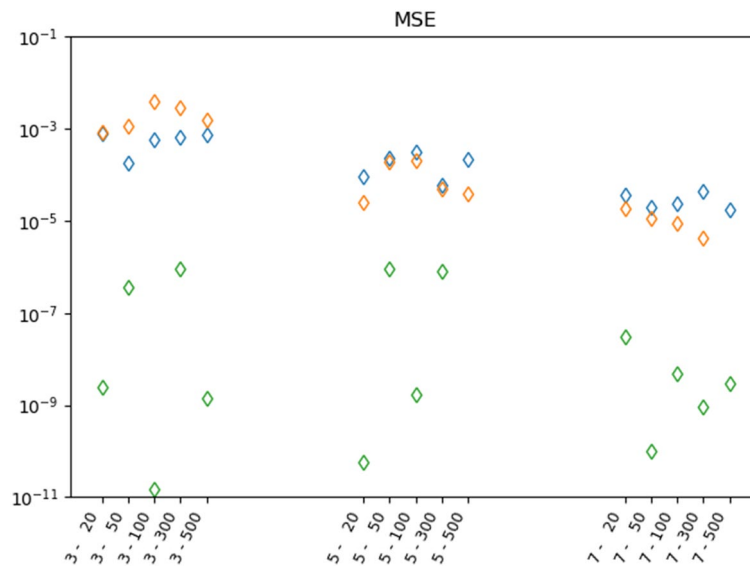


Figure 8. Mean squared error (mse) of the different NN's and PeNN between predicted and ground truth values for the test set. The colors correspond to NN 4-32-8-1 (blue), NN 4-128-32-8-1 (orange), and PeNN 4-6-2-1-1-1 (green). The horizontal axis correspond to different data sets generated with $n \in [3, 5, 7]$ different ϕ_p 's and for each ϕ_p , $N \in [20, 50, 100, 300, 500]$ different $\log \gamma$.

neural networks that are enriched with physics information are the efficiency of algorithms in finite dimension settings, robustness against data scarcity, and their modular transferability into other areas. Similarly, neural-mechanistic hybrid approaches have been utilized in genome-scale metabolic models recently by Faure *et al.*³⁴

The physics-encoded NNs allows continued learning as opposed to instance learning. Instance learning implies that it is required to retrain an entire network for predicting outcomes in a new setting for that network. Continued learning resembles a feature of intelligence. In this view, physics-encoded neural networks should not be viewed as an “AI” model but an “I” model, i.e. leaving out the term “Artificial”. In respect to learning, it is noteworthy that recently Zador³⁵ has argued that one may distinguish several levels of learning within animals, as opposed to the technical usage of the term learning within NN's. Within animals, there exist learned and innate mechanisms in executing functions. The innate mechanisms are encoded in the genome, which provides the rules that wire up the brain, for the behavioral programs for many functions (walking, swimming, etc.). The wiring of the brain is not explicitly programmed but will evolve during development, on the basis of a set of rules given by the coding. Interestingly, artificial NNs have to be optimized according to what is learned during their evolution and the learning during their functioning. In contrast, animals learn only in their functioning, as the innate part has been encoded. In analogy to the above, in physics-encoded NNs, the physics provides the set of rules, that allow for the explicit efficient wiring of the NNs during their training period. In this sense, one always should use NNs that build on previous solutions (use their learning). This represents a “real” learning phase where information is stored in a structured manner, suitable for building new information on top of that information, in a congruent manner, instead of building a structure all the time on the basis of data available, en re-iterating this latter learning. In fact, the physical information encoding implies a form of “real” intelligence. This explains the advantage of using physics to encode the NN wiring. This wiring is not random but instead is containing information, obtained from exposure to its surroundings and subsequently storing that information, which, if used again, explicates learning. This view is extended into a formal theory of clever computing of systems as expressed by Jaeger *et al.*³⁶, where the ideal computing of systems occurs as a bottom-up activity that structures the processes along which the computing takes place (cybernetic), using physically observables (physics-encoded), instead of the classic computing systems that describe the processing along structures that are present (algorithmic).

Our work is straightforward and uses essential physics to improve a NN model. The NNs can predict the viscosity curves very well, when the input parameters are part of the training set. Predictions are significantly worse for parameters not in the training set. Thus a NN does not sufficiently capture interpolation and extrapolation. The physics model may not be entirely covering all aspects, but the physics will be directing the number of possibilities for the NN while optimizing during its learning stage. The incorporation of the physics into the architecture of the NN such that the resulting architecture of the NN has the same architecture of the physics-based hierarchical architecture. It is this architectural characteristic that is important in being embedded into the NN, which is different from what has been reported until now.

Main algorithm
Define data $[n], [N], \dot{\gamma}_{min}, \dot{\gamma}_{max}, \phi_{pmin}, \phi_{pmax}, S_0, S_{\infty}, C, \eta_f, a, T, p_1$ and p_2
[NNarchitectures, PeNNarchitecture], [hyper parameters]
Loop over $[n]$ and $[N]$
Generate list of particle volume fractions $[\phi_p]$ of length n with $\phi_{pmin} < \phi_p < \phi_{pmax}$
Generate list of shear rates logarithms $[\log(\dot{\gamma})]$ of length N with $\dot{\gamma}_{min} < \dot{\gamma} < \dot{\gamma}_{max}$
Generate list of Quemada viscosities logarithms $[\log \eta([\dot{\gamma}], [\phi_p], S_0, S_{\infty}, C, \eta_f, a, T)]$ of length nN
Construct input data tensor X from $[\phi_p]$ and $[\log(\dot{\gamma})]$ with shape equals $(nN, 4)$
Construct output data tensor Y from $[\log \eta]$ with shape equals $(nN, 1)$
Split X and Y randomly in training set X_{train} and Y_{train} (80%)
and test-holdout set $X_{test-holdout}$ and $Y_{test-holdout}$
Generate list $[\phi_p]_{test}$ of randomly chosen $\phi_{pmin} < \phi_p < \phi_{pmax}$
Generate list $[\log \eta([\dot{\gamma}]_{test}, [\phi_p]_{test}, S_0, S_{\infty}, C, \eta_f, a, T)]_{test}$
Construct input data test tensor X_{test} from $[\phi_p]_{test}$ and $[\log(\dot{\gamma})]_{test}$
Construct output data test tensor Y_{test} from $[\log \eta]_{test}$
Generate list $[\phi_p]_{t2} = [\phi_p]_1 + [\phi_p]_2$ with list $[\phi_p]_2$ containing ϕ_p 's not in $[\phi_p]_1$
Generate list $[\log \eta([\dot{\gamma}], [\phi_p]_{t2}, S_0, S_{\infty}, C, \eta_f, a, T)]_{t2}$
Construct input data test2 tensor X_{t2} from $[\phi_p]_{t2st}$ and $[\log(\dot{\gamma})]$
Construct output data test2 tensor Y_{t2} from $[\log \eta]_{t2}$
Loop over network models $list$ [NNarchitectures, PeNNarchitecture]
Build model
Compile model with using hyper parameters for loss function and optimization algorithm
Train (Fit) network model using training set and hyper parameters for validation
and early-stopping callback giving trained model \mathcal{M}
Calculate predicted outcome of training set $\hat{Y}_{train} = \mathcal{M}(Y_{train})$
Calculate predicted outcome of test-holdout set $\hat{Y}_{test-holdout} = \mathcal{M}(Y_{test-holdout})$
Calculate predicted outcome of test set $\hat{Y}_{test} = \mathcal{M}(Y_{test})$
Calculate predicted outcome of test2 set $\hat{Y}_{t2} = \mathcal{M}(Y_{t2})$
Calculate performance metrics $mse_{\mathcal{M}}$ and $R^2_{\mathcal{M}}$ from \hat{Y}_{test} and Y_{test}
Plot loss function as function of epochs
Plot \hat{Y}_{train} vs. Y_{train} and $\hat{Y}_{test-holdout}$ vs $Y_{test-holdout}$
Plot \hat{Y}_{test} vs. Y_{test}
Plot \hat{Y}_{t2} vs. Y_{t2}
Plot performance metrics for all models and all test sets like $mse_{\mathcal{M}, n, N}$
Perform Friedman and Wilcoxon test

Table 1. Pseudo code of the program used The parameters and symbols are also used and explained in the manuscript. With [] a list is indicated.

Using physics-based information introduces information in NN's that has an experimental basis. The information added incorporates the existing and known natural sequence of events and their hierarchy, i.e. it incorporates the way that nature shows itself to us. In the same spirit as encoding physics information into NN's (PeNN's), one can encode information from other disciplines into NN's, creating in general "Information encoded NN's, or IeNN's. Subsequently, one can imagine building different IeNN's on top of one another, addressing even more complex systems.

Data availability

The datasets used and/or analysed during the current study is available from the corresponding author on reasonable request.

Code availability

The code used during the current study is available from the corresponding author on reasonable request.

Appendix

The pseudo code of the program can be found in Table 1

Received: 5 February 2024; Accepted: 19 June 2024

Published online: 01 July 2024

References

- van Mil, H. G., Foegeding, E., Windhab, E. J., Perrot, N. & Van Der Linden, E. A complex system approach to address world challenges in food and agriculture. *Trends Food Sci. Technol.* **40**(1), 20–32 (2014).
- Perrot, N. *et al.* Some remarks on computational approaches towards sustainable complex agri-food systems. *Trends Food Sci. Technol.* **48**, 88–101 (2016).
- Weinan, E., Han, J. & Zhang, L. *et al.* Integrating machine learning with physics-based modeling. [arXiv:2006.02619](https://arxiv.org/abs/2006.02619) (2020).
- Jia, X. *et al.* Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles. *ACM/IMS Trans. Data Sci.* **2**(3), 1–26 (2021).
- Daw, A., Karpatne, A., Watkins, W., Read, J., Kumar, V. Physics-guided neural networks (pgnn): An application in lake temperature modeling. [arXiv:1710.11431](https://arxiv.org/abs/1710.11431) (2017).
- Read, J. S. *et al.* Process-guided deep learning predictions of lake water temperature. *Water Resour. Res.* **55**(11), 9173–9190 (2019).
- Hanson, P. C. *et al.* Predicting lake surface water phosphorus dynamics using process-guided machine learning. *Ecol. Model.* **430**, 109136 (2020).
- Faroughi, S. A., Pawar, N., Fernandes, C., Das, S., Kalantari, N. K. & Mahjour, S. K. Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing. [arXiv:2211.07377](https://arxiv.org/abs/2211.07377) (2022).
- Alber, M. *et al.* Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. *NPJ Digit. Med.* **2**(1), 115 (2019).
- Mahmoudabadbozchelou, M. & Jamali, S. Rheology-informed neural networks (rhinns) for forward and inverse metamodelling of complex fluids. *Sci. Rep.* **11**(1), 12015 (2021).
- Saadat, M., Mahmoudabadbozchelou, M. & Jamali, S. Data-driven selection of constitutive models via rheology-informed neural networks (rhinns). *Rheol. Acta* **61**(10), 721–732 (2022).
- Young, C. D., Corona, P. T., Datta, A., Helgeson, M. E. & Graham, M. D. Scattering-informed microstructure prediction during lagrangian evolution (simple)—a data-driven framework for modeling complex fluids in flow. [arXiv:2305.03792](https://arxiv.org/abs/2305.03792) (2023).
- Dabiri, D., Saadat, M., Mangal, D. & Jamali, S. Fractional rheology-informed neural networks for data-driven identification of viscoelastic constitutive models. *Rheologica Acta* **62**, 557–568 (2023).
- Willard, J., Jia, X., Steinbach, M., Kumar, V. & Xu, S. Integrating physics-based modeling with machine learning: A survey. [arXiv:2003.04919](https://arxiv.org/abs/2003.04919), vol. 1, p. 34 (2020).
- Takeishi, N. & Kalousis, A. Physics-integrated variational autoencoders for robust and interpretable generative modeling. *Adv. Neural. Inf. Process. Syst.* **34**, 14809–14821 (2021).
- Peng, G. C. Y. *et al.* Multiscale modeling meets machine learning: What can we learn?. *Arch. Comput. Methods Eng.* **28**, 1017–1037 (2021).
- Lie-Piang, A. *et al.* Machine learning to quantify techno-functional properties—a case study for gel stiffness with pea ingredients. *Innov. Food Sci. Emerg. Technol.* **83**, 103242 (2023).
- Bhagya Raj, G. & Dash, K. K. Comprehensive study on applications of artificial neural network in food process modeling. *Crit. Rev. Food Sci. Nutr.* **62**(10), 2756–2783 (2022).
- Datta, A. *et al.* Computer-aided food engineering. *Nat. Food* **3**, 894–904 (2022).
- Derkach, S. R. Rheology of emulsions. *Adv. Colloid Interface Sci.* **151**, 1–23 (2009).
- Rao, M. A. *Rheology of Fluid and Semisolid Foods* (Springer, 2014).
- Larson, R. G. & Wei, Y. A review of thixotropy and its rheological modeling. *J. Rheol.* **63**, 477–501 (2019).
- McClements, D. J. Modeling the rheological properties of plant-based foods: Soft matter physics principles. *Sustain. Food Prot.* **1**, 101–132 (2023).
- Quemada, D. Rheology of concentrated disperse systems and minimum energy dissipation principle—I. viscosity-concentration relationship. *Rheol. Acta* **16**, 82–94 (1977).
- Quemada, D. Rheological modelling of complex fluids. I. The concept of effective volume fraction revisited. *Eur. Phys. J.-Appl. Phys.* **1**, 119–127 (1998).
- Berli, C. L. & Quemada, D. Rheological modeling of microgel suspensions involving solid-liquid transition. *Langmuir* **16**, 7968–7974 (2000).
- Berli, C. L., Quemada, D. & Parker, A. Modelling the viscosity of depletion flocculated emulsions. *Colloids Surf. A Physicochem. Eng. Asp.* **203**, 11–20 (2002).
- Quemada, D. & Berli, C. Energy of interaction in colloids and its implications in rheological modeling. *Adv. Colloid Interface Sci.* **98**, 51–85 (2002).
- Jansen, K. M. B., Agterof, W. G. M. & Mellema, J. Viscosity of surfactant stabilized emulsions. *Cit. J. Rheol.* **45**, 1359 (2001).
- Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016).
- Nwankpa, C., Ijomah, W., Gachagan, A. & Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. [arXiv:1811.03378](https://arxiv.org/abs/1811.03378) (2018).
- Mercioni, M. A. & Holban, S. The most used activation functions: Classic versus current. In *2020 International Conference on Development and Application Systems (DAS)*, pp. 141–145, IEEE (2020).
- Zamri, N. E. *et al.* A modified reverse-based analysis logic mining model with weighted random 2 satisfiability logic in discrete hopfield neural network and multi-objective training of modified niched genetic algorithm. *Expert Syst. Appl.* **240**, 122307 (2024).
- Faure, L., Mollet, B., Liebermeister, W. & Faulon, J.-L. A neural-mechanistic hybrid approach improving the predictive power of genome-scale metabolic models. *Nat. Commun.* **14**(1), 4669 (2023).
- Zador, A. M. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nat. Commun.* **10**(1), 3770 (2019).
- Jaeger, H., Noheda, B. & Van Der Wiel, W. G. Toward a formal theory for computing machines made out of whatever physics offers. *Nat. Commun.* **14**(1), 4911 (2023).

Acknowledgements

We acknowledge funding from the Dutch “Sectorplan Techniek II”.

Author contributions

E. van der Linden and M. Meinders conceptualized the idea and approach. Programming has been performed by M. Meinders. All authors discussed the results. E. van der Linden and M. Meinders drafted the first manuscript. All authors contributed to later versions the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1038/s41598-024-65304-w>.

Correspondence and requests for materials should be addressed to M.B.J.M.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2024