THE JOURNAL OF PHYSICAL CHEMISTRY C

Article

Learning from the Machine: Uncovering Sustainable Nanoparticle Design Rules

Clyde A. Daly, and Rigoberto Hernandez

J. Phys. Chem. C, **Just Accepted Manuscript** • DOI: 10.1021/acs.jpcc.0c01195 • Publication Date (Web): 20 May 2020

Downloaded from pubs.acs.org on May 22, 2020

Learning from the Machine: Uncovering Sustainable Nanoparticle Design Rules

Clyde A. Daly Jr. and Rigoberto Hernandez*

Department of Chemistry, Johns Hopkins University, Baltimore, MD 21218

Abstract

Machines consisting of bags of artificial neural networks (ANNs) have been constructed to connect nanoparticle features to the viability of a broad class of organisms upon exposure. The optimization of these machines is based on a relatively small data set but, through consensus across a bag of ANNs, predicts at a level of confidence comparable to the experiment and performs better than chance. The mining of the machine across the feature space allows for the discovery of design rules for nanoparticles with increased viability. As such, we demonstrate the efficacy of inversion as an approach to learn from the machine in the context of designing sustainable nanoparticles. For example, we find that in lithium NiMnCo oxide nanoparticles that increased manganese content is associated with greater viability, carbon dots reduce viability less than quantum dots, and gold nanoparticle coatings can significantly affect viability at high concentration.

I. Introduction

Machine learning (ML) techniques have helped revolutionize a number of scientific fields in the past decade because of their power in identifying relationships between data.[1-6] The massive data troves available from consumers and social media have enabled the development of new tools

* E-Mail: r.hernandez@jhu.edu

that are increasingly available for commercial and academic applications in other fields, including biology, computer science, and medicine.[1-2, 7-12] ML has also been successfully used in chemistry in at least two significant and complementary ways:[13] (i) Computational and theoretical chemists have used ML to build fast, accurate models of atomic and electronic behavior.[14-35] While these models are not directly constructed from the underlying physical laws, they appear to infer them through data mining of high-accuracy computational data and have been seen to be powerful and accurate predictors of chemical behavior. (ii) Chemists have also used ML to find relationships in observed data sets of varied sizes and to plan synthetic routes for making new molecules, especially through models based on quantitative structure-activity relationships.[36-51]

Machine learning spans a broad set of available methods, from the relatively simple least squares linear regression to significantly more complex methods like random forest classification and neural network regression.[52-54] While there are a large number of rules for distinguishing among ML methods, perhaps one of the most useful (and simple) relies on addressing how the underlying data is accessed by the machine. Each datum in a data set typically contains two types of entries: features and labels. Features are the entries that define the state—evoking the use of facial *features* in facial recognition. They are associated with entries called labels—evoking the names of individuals *labeling* faces in facial recognition. A particular set of features and labels (i.e. a single datum) is called an example. In least squares regression, the domain variables x are features and the range variables y are labels. While in the facial recognition problem, it is fairly clear which entries in an example are features and which is the label, in general, there is wide flexibility in the partitioning of the entries between features and labels. If the ML method requires the dataset to have both features and labels, it is *supervised*. If the machine learns from data consisting only of features, then the method is *unsupervised*. Clustering algorithms fall into this second category while least squares regression falls into the first category. There also exist a number of semi-supervised methods which can work with or without labels.[55] In this work, we use a supervised machine learning method, neural network regression, to find connections between nanomaterial properties as features and viability as the label. The generic structure of the artificial neural networks (ANNs) we employ is shown in Fig. 1, indicating the use of bagging and other ANN-optimization techniques described below.

There is precedent for applying machine learning techniques to address problems in environmental and biomedical nanotechnology.[12, 48, 56-64] Findlay *et al.* [65] created a model, based on random forest classification, that was able to predict the types of proteins present in Ag nanoparticle protein coronas. Recently, Jones *et al.* [66] successfully used data mining and decision trees to identify features predicting nanoparticle toxicity that remain relevant across multiple

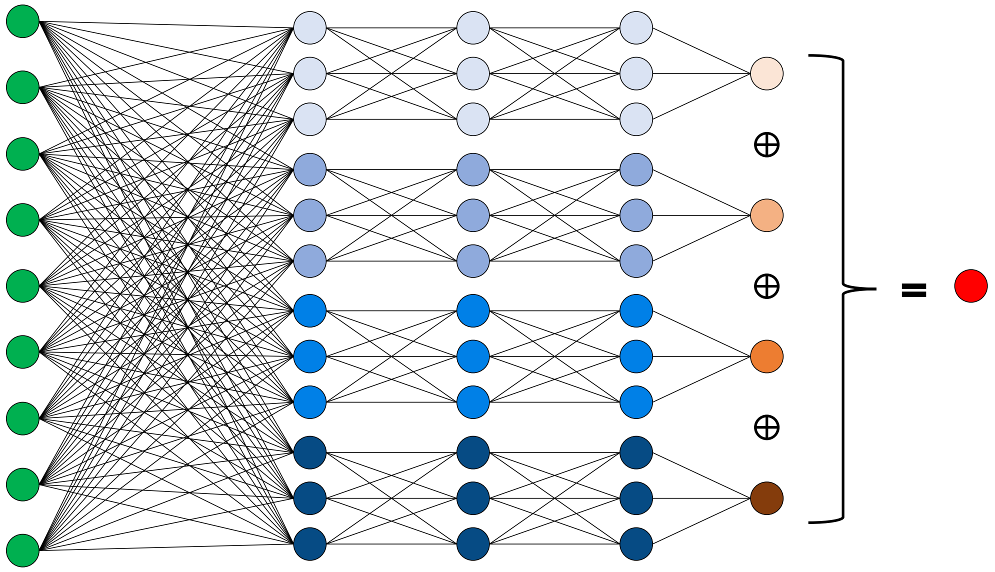


Figure 1: Schematic of the bagged artificial neural network (ANN) structure used in this work. Green nodes are the features in the input layer, blue nodes are elements within the hidden layers of each ANN, orange nodes are the output layer of each ANN, and the red node is the predicted label calculated through some kind of consensus of the orange nodes. The hidden layer nodes in each individual ANN are colored differently to emphasize that each is trained separately and thus they have generally different parameterizations and outputs. In this structure, processing of all inputs occurs separately within each ANN prior to recombination at the end.

unrelated studies. Applying ML to environmental nanotechnology remains challenging because such implementations tend to suffer from low generalizability due to the size and diversity of the nanomaterial datasets being limited by the high cost of the requisite experiments. Here, data from the Center for Sustainable Nanotechnology (CSN), primarily mined from the literature, has been used exclusively.[67-76] While this is a relatively small data set—with just over 200 examples—it has the advantage of consistency across all of the samples providing a complete set of the desired features.

The disadvantage of choosing a small data set to represent such complex quantitative structure-activity relationships is that many approaches, such as perturbation theory,[59, 62-63, 77] similarity modeling,[40, 51, 58] and single ANNs, do not readily span the corresponding wide domain of features. Instead, throughout this work, groups of related ANNs are created and their predictions are averaged to give a final consensus prediction and uncertainty. There is precedent for this in the ML literature, where it is commonly termed "bagging" (**b**ootstrap **agg**regat**ing**).[78-82] Together, these networks form an aggregated machine called a "bag" of ANNs. We optimize and then implement such a bag of ANNs for the prediction of reduced nanoparticle toxicity, where this label is quantified in terms of organismal viability upon nanoparticle exposure. This is not an exclusive figure of merit and is mainly used here to provide a clear metric for the training, validation and prediction of the ANN machine. It is also an observable that is typically measured, and therefore readily available in the data.

We find an ANN bag that makes better-than-chance predictions and which has uncertainty comparable to the experimental uncertainty. We have also partially inverted the ANN's feature-to-label paradigm by determining regions of the feature space resulting in increased viability. Identification of such inversion has been a goal of this field for some time, and several possible solutions are available.[28, 38, 83-85] Overall, the expected mean absolute error (MAE) of predictions produced by our ANN is somewhere between 0.2 and 0.3 on a viability scale in which 1.0 refers to no change in population upon exposure. Through our inversion, we learn from the machine in the sense that we uncover patterns or design rules for nanoparticle properties that lead to increased bacterial viability. It suggests subdomains in the space of composition and structure of lithium nickel magnesium cobalt oxide (NMC) nanosheets linked to targeted levels of viability, especially as regards the composition of Mn and Co. Gold nanoparticle (AuNP) coatings are found to influence their viability, with poly(allylamine hydrochloride) in particular leading to strong cellular responses. Finally, carbon dots are found to be strongly non-toxic regardless of precursor while quantum dots are found to be highly toxic, an effect that is worsened by the presence of a ZnS coating. These trends agree with available experimental results and suggest new nanoparticle compositions and features which should be fully characterized by experiment either to extend the database or to identify desired nanoparticle targets.

II. Materials and Methods

A. ANNs, parameters and hyperparameters

Each node in each layer of an ANN is connected to the nodes in the layers before and after it (Fig. 1). There are three types of layers – input, output, and hidden. The input layer receives the features from each example. The output layer produces the final result of the ANN which corresponds to the label of a given example. Between them, there are some number of hidden layers. These are termed “hidden” because they do not interact with objects outside of the network, unlike the input and output nodes. The output value n_{ij} represented by each node at layer i and row j is

$$n_{ij} = \alpha\left(\sum_{l=1}^{\rho_{i-1}} w_{ij(i-1)l}\, n_{(i-1)l} + b_{ij}\right) \tag{1}$$

where the links between nodes are identified with weights w_{ijkl} and each node is associated with a bias b_{ij}. For the input nodes, the n_{kl} values are provided by the example being considered and the biases are always zero (there are no weights since the input node is the first layer). The nonlinear "activation function," $\alpha(\cdot)$, increases the complexity of the representations that the ANN can produce. The complex set of transformations supplied by the layer structure combined with the non-linearity supplied by the activation functions allows ANNs to learn (or represent) extremely complex patterns and relationships between input and output data.[3-5] Finally, ρ_{i-1} represents the number of nodes in the prior layer. In this work, ANNs have been realized using Keras and Tensorflow version 1.14.0 in python.[4, 86-87] The python libraries numpy, scipy, pandas, scikit-learn, python-ternary, and matplotlib were extensively used to process and visualize data, and Jupyter Notebooks were also employed.[4, 88-94] Categorical features were transformed into numerical ones using one-hot encoding, as described in more detail below.[3, 95]

The *parameters* of the ANN are precisely the weights and biases. When an ANN is being trained, its parameters are typically updated through a method known as backpropagation. Briefly, the neural network is initialized with random values for its weights and biases. Then it is applied to some number of examples in sequence and produces output values that are generally different from the associated labels. The mean squared error between the set of ANN outputs and actual labels is used to define a loss (or cost) function,

$$C(p) = \langle \Delta F^2 \rangle_E = \langle (F - L)^2 \rangle_E , \qquad (2)$$

where L is the actual label for a particular example, F is the output value of the ANN, and the average is taken over examples E in the training set. In this work, the loss function is always the mean squared error over the specified data set. However, the mean absolute error (MAE) is often the reported metric of model performance and is also used here. This first moment function is sometimes preferred to functions of the second moment —such as the cost function in Eq. 2— because its linearity allows averages of MAEs to also be a MAE. During training, backpropagation is directed by gradient descent: the gradient of the cost function with respect to the parameters is computed and used to calculate the new weights and biases. This process is generally repeated until the value of the cost function is static, at which point the ANN is considered to be optimized with respect to its parameters. The final parameter values depend on a number of factors. First, they depend on the specific set of training data used for optimization since the loss function is defined as the mean squared error over its examples. Second, they depend on the choice of the so-called "hyperparameters" which comprise two broad classes of descriptors defined below.

One class of hyperparameters describes the structure of the ANN. It includes the choice of the activation function $\alpha(\cdot)$, the number of layers, and the number of nodes in each layer. The number of nodes per layer and the number of layers together determine the number of ANN parameters. Among the most commonly used ANN activation functions is the rectified linear unit (ReLU) which is

$$\alpha_{\mathrm{ReLU}}(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{3}$$

There are a number of functions that act as improvements on the ReLU function, most of which seek to address the absence of a gradient in the function for values less than zero which can lead to neurons that "die" *i.e.* stop learning. Many of these functions also contain pre-specified or sub-parameters. For instance, the Leaky ReLU function is

$$\alpha_{\mathrm{Leaky\ ReLU}}(x) = \begin{cases} x, & x > 0 \\ e\,x, & x \leq 0 \end{cases} \tag{4}$$

where *e* is normally set to 0.01. These sub-parameters rarely need to be changed. There are also a number of activation functions that have generally sigmoidal behavior, but they have fallen out of favor due to relatively poor performance.[3]

A second class of hyperparameters are those related to the training of the network parameters. Among these are the optimization algorithm, the batch size, and the number of training epochs. In most cases, datasets are too large to efficiently sample every example before taking a single gradient descent step. Instead, datasets are often broken up into smaller sections called "batches" during ANN parameter training. When every example in the dataset has been visited, an epoch is said to have passed. Together, the epoch number and batch size control the completeness of the ANN training. The optimization algorithm controls how the training steps are taken (i.e. how parameters are changed during training). As noted above, most of these algorithms are based on gradient descent, but are implemented in different ways. One of the most commonly used optimization algorithms is called Adaptive Moment Estimation (Adam), which modifies the distance of upcoming gradient steps depending on the recent history of the optimization.[96] Like activation functions, optimization algorithms often contain pre-specified or sub- parameters that are intentionally not varied during training or optimization without loss of generality in the model's function.

B. Default hyperparameters for divide-and-conquer optimization

Altogether, the parameters of an ANN depend on the specific set of examples used for training, the hyperparameters of the network related to its structure and the process for optimizing its parameters, expressed as

$$F = f_{\mathrm{A}}(I;p, E,h_p) \qquad (5)$$

where f_{A} is the map representing the action of the ANN on the domain of features I, and F is the output value of the ANN. The map is specified according to the training parameters p of the ANN, the set E of examples used to train the network, and the chosen hyperparameters h_p. The efficiency and utility of an ANN with respect to its trainability and predictability is strongly dependent on the choice of hyperparameters, and so they too require optimization. As our output label is a measure of nanotoxicity and our inputs are, in part, nanomaterial properties, this mapping bares similarity to quantitative structure activity/toxicity relationship modeling.[36, 40, 58, 77] By adding information about the biological systems and environment to our feature set, we are able to cover cases where the structure and activity relationship is altered by the surrounding system. The features are described in more detail in section F below.

Here, ANNs are built across a range of hyperparameter values so as to optimize the overall performance on a validation set (that was not used in the training of the ANNs.) Since the dataset was small, ANNs were generally validated using *k*-fold validation (kFV).[97-100] In *k*FV, a set of *k* networks is created as in *k*-fold structuring (see below). Each ANN then predicts the outputs for the group of examples it was not trained on (its validation set). The MAE between the network outputs and the actual labels is recorded; this is then repeated for each network in the set. After *k* rounds, the mean of the model performance measures is taken, providing an estimate of the performance of any particular model having the same hyperparameters. As such, the final MAE provided by *k*-fold validation is

$$\langle|\Delta F|\rangle_{k\mathrm{FV}} = \frac{1}{k}\sum_{i=1}^{k}\langle|f_A(I_i;p_i,E_i,h_p) - L_i|\rangle \tag{6}$$

where I_i and L_i are the features and labels of the group of examples that network i was not trained on, E_i are all other examples, and p_i are the parameters acquired by the network through training on E_i with hyperparameters h_p. We applied four-fold validation (4FV) throughout this work. For k less than 4, we would not have obtained sufficient statistics between the training sets to establish confidence. In practice, we did not find a need to move beyond k=4.

Because the hyperparameters are selected according to validation performance, information about the validation set "leaks" into the model. A "test set" is used to provide an estimate of model performance on truly new data. Prior to the optimization of the hyperparameters of our model, a test set of 20 examples was selected randomly and then removed from the dataset. These examples were excluded from all forms of model validation and hyperparameter selection during training except for their use as a final test set. In our hyperparameter optimization, we tested the effectiveness of four or five possible values for each of the six hyperparameters. A complete combinatoric optimization across these values would require the creation of 40,000 individual ANNs. In order to eliminate some of these combinations, a layered grid search was performed.[4-5] In this procedure, pairs of hyperparameter values were examined together, while the values of the remaining hyperparameters were held fixed at default values. We chose these values either because they are commonly used in neural network regression or because they maximize model complexity without egregiously overfitting the data. Our default hyperparameters are: 5 hidden layers, 5 nodes in each hidden layer, ReLU activation, the Adam optimizer, 1000 training epochs, and a batch size of 8. Once hyperparameter values resulting in excessive damage to model performance were eliminated, the remaining hyperparameter choices were optimized together to find the best hyperparameter combination. In addition, all ANNs in this work limit the range of output node values to be greater than or equal to zero, as viability values cannot be negative.

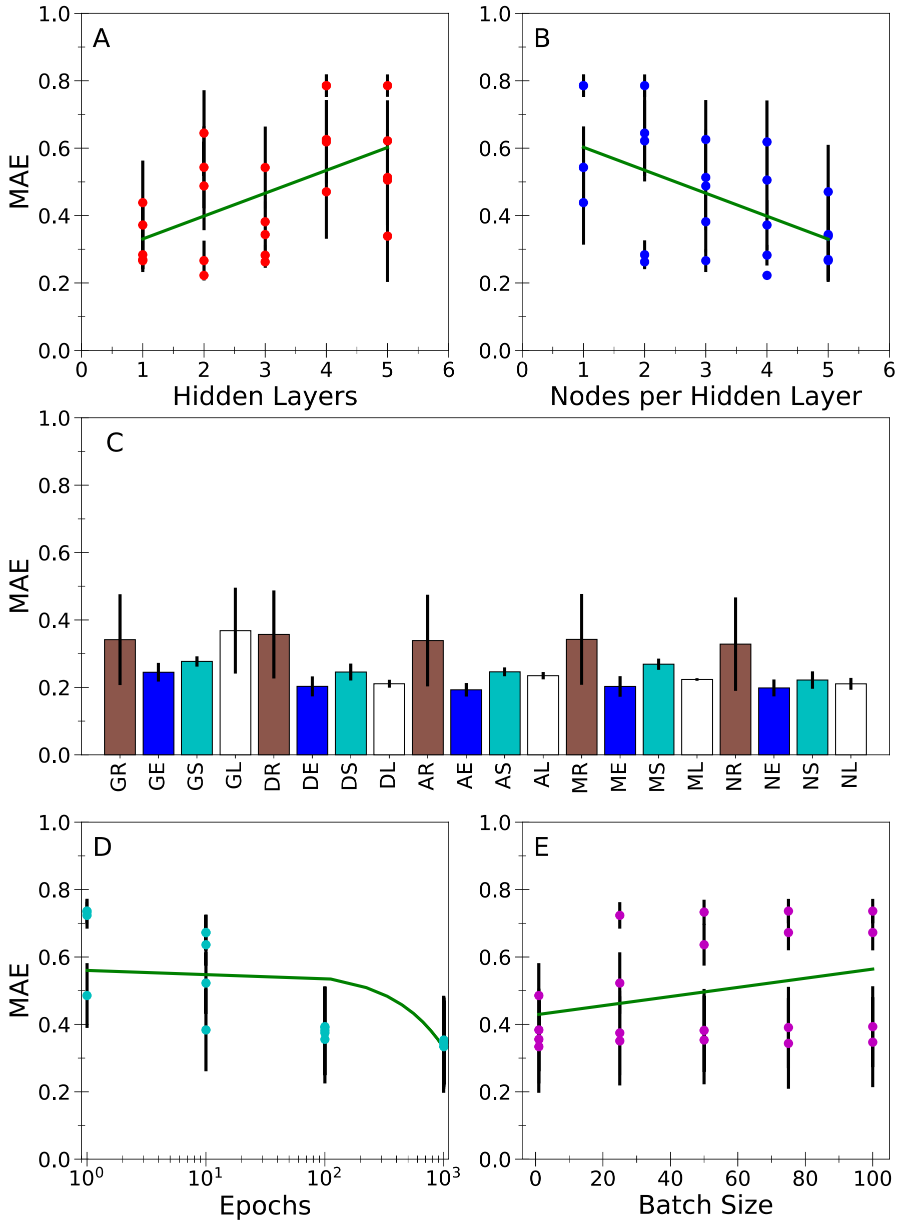


Figure 2: *(A) The relationship between error in viability predictions and the number of hidden layers in the model. (B) The relationship between error in viability predictions and the number of nodes per hidden layer in the model. (C) The relationship between optimization algorithm and activation function combinations and error in viability predictions. G, D, A, M, and N are the Adagrad, Adadelta, Adam, Adamax, and Nadam optimization algorithms, respectively. R, E, S, and L are the ReLU, ELU, SeLU, and Leaky ReLU activation functions, respectively. (D) The relationship between the number of training epochs and viability prediction error. Note that this is an x-log plot. (E) The relationship between the batch size and viability prediction error. In all plots, green lines are linear fits.*

C. Divide-and-conquer optimization of the hyperparameters

First, the number of hidden layers and the number of nodes per hidden layer were optimized by applying a grid search across both hyperparameters over the integer values between 1 and 5. The searched values were limited to this range in order to minimize the number of parameters in the model, which would otherwise increase quickly. With only 5 layers and 5 nodes in each layer, the ANN would contain 555 adjustable parameters. 4FV was performed on each of the ANNs corresponding to the 25 combinations of layer number and node number pairs leading to the results shown in Figs. 2A and 2B, respectively. Figure 2A displays $\langle|\Delta F|\rangle_{4\mathrm{FV}}$ as a function of the number of hidden layers. Here, there is a general increase in the MAE as a function of the number of hidden layers (p = 4.4 x 10^{-3}, R = 0.55). Figure 2B presents the relationship between $\langle|\Delta F|\rangle_{4\mathrm{FV}}$ and the number of nodes in each hidden layer. There is an inverse relationship between these values (p = 4.2 x 10^{-3}, R = -0.55). These results lead us to favor restricting the number of layers to lower values (≤ 3) and the number of nodes per layer to higher values (≥ 3).

A grid search was then performed for each of 20 combinatorial choices of the parameter optimization algorithm and node activation function. The results are shown in Fig. 2C. Of the activation functions, only ReLU has consistently bad performance, though Leaky ReLU does not perform well when paired with Adagrad. Comparison across the activation functions suggests that Elu (average MAE of 0.21), has the best performance of any of those tested (average MAE of 0.26 across all activation functions). Nadam (0.24) has the best performance of the optimization algorithms tested, followed by Adadelta (0.25) and Adam (0.25).

Finally, the effect of changing the number of training epochs and the batch size was evaluated. Epoch numbers between 1 and 1,000 and batch sizes between 1 and 100 were explored. The results from 4FV on these ANNs is shown in Figs. 2D and 2E. The number of epochs (p = 4.4

x 10^{-3} , R = -0.61) had a significant effect on the resulting error, but the batch size (p = 0.20, R = 0.30) did not. This effect from the epoch number should be expected: more training should generally result in a better model. However, one should still be wary of overfitting, particularly when smaller datasets are used. In section III.A, we use several tests to ensure that we have not overfit our data and that our models are trustworthy.

A limited grid search of all hyperparameters, informed by these findings, was then performed. ANNs with between 1 and 3 hidden layers, 3 and 5 nodes per hidden layer, and the Elu activation function were trained with Adam, Adadelta, or Nadam, with batch sizes of 25 or 100 examples and for either 100 or 1000 epochs. This search required the training and evaluation of 432 individual ANNs. The ANN structure with the best performance (MAE of 0.21 ± 0.02) had 3 hidden layers, 5 nodes in each hidden layer, the Elu activation function, and was trained with the Adadelta optimizer for 1000 epochs with a batch size of 25.

D. Viability

Organism viability was chosen as a target for ANN training as it is an anti-correlated proxy for nanoparticle toxicity. Viability is a measure of the fraction of organisms that survive upon exposure to a substance, such as nanoparticles. The number or density of survivors is compared to a control where no foreign substance was introduced. Thus, viability normally deviates from 1 (i.e. the same number of organisms are present in both the experiment and control) and can even be greater than 1 if the relative survival is somehow enhanced by exposure to the substance. The minimum value of viability is 0, representing total organism death. The dataset used in this work contains 200 examples of organismal viability experiments extracted from prior publications of the Center for Sustainable Nanotechnology.[67-76] While this dataset is relatively small, it allows us to make inferences about the best model hyperparameters, test methods of model validation, and establish a framework for making predictions about nanoparticle viability from a given model. Viability, whether measured by colony counting or growth-based viability, is treated as the same variable, though it should be noted that colony counting has a higher intrinsic error.[70]

E. Bagging

A common method for increasing the power of machine learning predictions is to train collections of individual machines and group or combine their results in some way. This method is called "bagging," short for "**b**ootstrap **agg**regat**ing**."[78, 82, 99] In the context of ANNs, the networks are trained such that they have different parameters, and the outputs are averaged together. Together, these networks form a larger machine learning object called a "bag" of ANNs. The most direct way to obtain distinct parameters for the individual ANNs is to use distinct subsets of the training examples to train the networks, while leaving the hyperparameters constant. Thus, the output of a bag of ANNs is

$$F = \frac{1}{N}\sum_{i=1}^{N} f_A(I; p_i, E_i, h_p) \tag{7}$$

where N is the number of ANNs in the bag and other variables retain their meanings from Eq. 6. In this work, three methods are employed to produce the distinct example sets that are used to create the bags: (i) In k-fold structuring (kFS), the dataset is shuffled and split evenly into k groups. Each of k networks is then trained using k-1 of these groups such that no two networks share the same training dataset. The collection of the resulting k models is the bag. The average of the values across the bag is its prediction, and an uncertainty can be obtained from the deviation between the prediction of the bag and the outputs from each of its models. (ii) In i iterated k-fold structuring (i-kFS), k-fold structuring is performed i times, shuffling the data each time before it is split into k groups. This produces a bag containing $i \times k$ distinct networks. A useful feature of bags created using k-fold structuring and i iterated k-fold structuring is that the bags are guaranteed to have been collectively trained on the whole dataset. (iii) In so-called random-fold structuring (rFS), distinct networks are trained on a random fold —viz, a randomly selected portion— of the dataset. This can be done as many times as desired without duplication (given a large enough dataset), producing very large bags.

F. Data Preprocessing

The full dataset prior to pre-processing is provided in the Supporting Information, and the full list of the features accounted for in the data set are listed in Table 1. Prior to use, the data set was transformed in a number of ways. All feature values were normalized to have a standard deviation of 1 and a mean of 0. The base 10 logarithm of the concentration was used, rather than the concentration itself, as this improved model performance markedly. In order to operate on values of zero, a small value (10^{-9} ppm) was added to all concentration values. This did not alter the concentration of non-zero values (ranging from 10^{-4} to 10^{4} ppm) in an appreciable way. NMC nanoparticle examples were characterized with a feature reporting the surface area of exposed NMC per liter as it approximately reflects the surface area exposed to solvent which is a characteristic that was found to be germane to viability in prior CSN work.[72] For non-NMC examples, this value is by definition zero. In several cases, it was not possible to determine the full chemical makeup of a nanoparticle, and this is noted in the database and considered by the ANNs through a simple true/false variable noting whether the composition features reflected the complete makeup of the nanoparticle. Because ANNs can only interact with numerical features, non-numerical (i.e. categorical) features were one-hot encoded. Consider a categorical feature *I*. In one-hot encoding, the feature *I* is eliminated and new numerical features I_a are created for each category *A*. For each example *E* with feature value *E(I)*, the value of I_a is

$$I_a = \begin{cases} 1, & E(I) = A \\ 0, & E(I) \neq A \end{cases} \tag{8}$$

resulting in a set of new numerical features with the full information content of *I*. The new features I_a are called one-hot vectors or one-hot features. Nanoparticle composition is treated in a semi-one-hot manner. Each element present in a nanoparticle in the database is given a separate feature I_a which is equal to the mol % of the given element in that nanoparticle. These feature engineering processes resulted in a total of 82 numerical features that were used by the ANNs for prediction.

Table 1: The features used to categorize the viability experiments in training the neural networks are listed and sorted according to the type of values they exhibit (as noted in the columns) and the types of characteristics they describe (as noted in the rows).

Features	Numerical	Categorical	Mixed
Nanomaterial Properties	*Particle Diameter Per Dimension (nm), Concentration (mg/L), Exposed NMC Surface Area (m^2/g)*	*Type, Total Composition Knowledge, Capping Agent, Shape*	*Elemental Composition (mol %)*
Organism Characteristics		*Bacteria, Gram, Identity, Mutation, Viability Method*	
Experimental Conditions	*pH, Natural Organic Matter (mg/L), Total Centrifugation Steps, Exposure Time (min)*	*Medium, Purification Method*	

III. Results and Discussion

A. Validation of the Machine: Better than Chance

The optimal ANN hyperparameterization (see Methods) was evaluated in its ability to lead to ANNs (or bags of ANNs) which can learn to predict viability values at a better than chance rate, its behavior when confronted with falsified data, and its performance on validation and test sets. To define "better than chance" predictions on the dataset, we find the MAE value that would be produced if the model prediction was performed according to two null hypotheses: N_1 defined such that the model knows nothing about the data except the range of its labels, and N_2 defined such that the model knows both the label range and the frequency of label values. In both of these cases, the model would have no other information about how features are connected to labels, and its outputs can be represented by random selections from an appropriate probability distribution without direct reference to the input features. These outputs are generated using a random number generator and the corresponding MAE for a given realization of the model is

$$\langle|\Delta x_{N_H}|\rangle = \frac{1}{M}\sum_{i=1}^{M}|x_i(f_i) - \xi_i(p_H)| \quad (9)$$

where $\{x_i(f_i)\}$ is a given data set with M examples consisting of labels x_i corresponding to features f_i, respectively, and ξ_i is a random number chosen according to the p_H distribution consistent with the N_H hypothesis. The uniform chance distribution p_1 represents the values generated by a uniform random number generator producing values between 0 and the maximum value in *L*. The frequency-weighted chance distribution, p_2, is the frequency with which the labels appear in the data set with no reference to the underlying features. p_2 can be generated via random permutations of the dataset's labels. We computed the average MAE for a given null hypothesis by averaging the absolute differences between the true viability dataset and an ensemble of one million realizations of the "predictions" from the associated chance distribution. The resulting $\langle|\Delta x_{N_1}|\rangle$ was 0.87 and $\langle|\Delta x_{N_2}|\rangle$ was 0.49. The difference between $\langle|\Delta x_{N_1}|\rangle$ and $\langle|\Delta x_{N_2}|\rangle$ reflects the fact that the label —namely, viability— of the examples in our system is not uniform between 0 and the maximum value in *L*.

With the results of the null hypotheses in hand, a relevant figure of merit is the error of the ANN converged over possible training sets—viz. possible parameter sets for the ANN. In order to reach convergence, 25-4FV (*i*-4FV for $i = 25$ different arbitrary four-fold partitionings of the examples; see Methods) was used and the resulting $\langle|\Delta F|\rangle_{25-4\mathrm{FV}}$ was found to be 0.27 ± 0.01, substantially smaller than the values seen for the null hypotheses for chance. Thus, we can

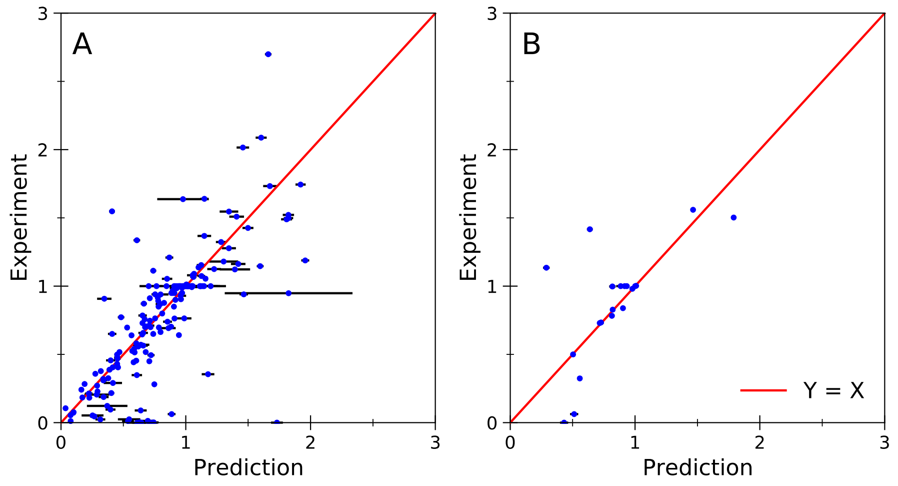


Figure 3: (A) Comparison of the viability predictions of 100 ANNs with the experimental values. (B): Model performance of 100 ANNs on the test set. Better predictions are closer to the red Y = X line.

conclude that these ANNs are able to learn real connections between features and labels in the dataset.

An additional test was performed in which an ensemble of fictitious data was generated by randomly producing feature values between 0 and 1 and label values between 0 and the maximum value in L such that the final data set had the same number of examples as the original dataset. There is little to no information content in this dataset, and thus it can be used to examine the performance of the ANN in the case of a null dataset. 25-4FV was used to obtain a bag of ANNs (as defined in the Methods section) whose regression represents the fictitious data. It led to a converged MAE $\langle|\Delta F(\mathrm{null})|\rangle_{25-4\mathrm{FV}}$ equal to 0.98 ± 0.01. As hypothesized, this is much higher than the MAE $\langle|\Delta F|\rangle_{25-4\mathrm{FV}}$ seen for the actual examples. Thus, we conclude that the true dataset contains correlated information, not available in the null dataset, that can be learned by the ANN to predict viability.

B. Validation of the Machine: Accuracy Comparable to Experiment

Because of our smaller dataset, single models are highly susceptible to bias in their predictions. Ensembles of ANNs can minimize the effect of individual model bias by cancelling opposing biases. The collection of such an ensemble is then a new model called a bag. Here, we validate bagged ANNs producing predictions of viability values and use a previously sequestered test set to obtain a final model performance estimate. In validation, one determines if the parameters that a particular model has been trained to are accurate through the quality of its predictions on the validation set. However, the validation data is also used to evaluate ANN structure (*i.e.* hyperparameters; see Methods) optimization and thus cannot be used as a representation of model performance on data the model has no direct information about. In testing, one examines the quality of hyperparameters by evaluating prediction quality on data excluded from the hyperparameter optimization process. 100 ANNs were trained using random-fold sampling (rFS, see Methods), training on 75 % of the dataset examples. The rFS bag has the same number of ANNs as the 25-4FV bag used above, and hence serves as a useful reference to whether the latter has any significant bias due to the less-random partitioning of the training examples. After each ANN was trained in the rFS set, it was used to predict values for the remaining quarter of the examples (used as a validation set) resulting in predictions of each label by a unique ensemble (bag) of approximately 25 ANNs. The results of this analysis are shown in Fig. 3A. This plot compares the ensemble viability predictions to the true viability values. The MAE between ensemble predictions and experimental values was 0.21, comparable to the accuracy found from 25-4FV.

During hyperparameter optimization, 20 randomly selected examples, termed a “test set,” were excluded from the dataset. Because these examples were not used to develop the model structure, no information about them was incorporated into the model. As such, the model performance on these values is taken to represent the performance of the model on entirely new data and is used to evaluate the quality of the hyperparameters. The procedure used above for validation examples was repeated to evaluate the performance of a bag of 100 distinctly trained ANNs on this test set. The predicted values are compared to the experimental values in Fig. 4B, and the MAE is 0.23. ANN performance is good and likely not due to chance (p = 0.001, R = 0.67).

It is notable that the uncertainty reported in viability experiments using both growth-based viability and colony counting methods can range from 0 to 0.3.[70] As reported above, the expected MAE of predictions of viability produced by our bags of ANNs is somewhere between 0.2 and 0.3. Thus, an ANN bag produces predictions with a comparable level of confidence to the ground truth in the data set it has learned from.

C. Learning from the Machine

ANNs are typically used to predict outputs given a set of inputs. However, it is tempting to invert them by way of determining the set of inputs —viz. features— that result in a given desired output —viz label. When using ANNs, there are several difficulties involved. First, a given output is quite likely to have many possible associated sets of inputs – ANNs often represent many-to-one relationships.[83] The fundamental complexity of ANNs is also challenging because they generally do not admit to a simple method for inverting the functional form of the ANN. There are several complex methods that have been developed for the inversion of ANNs. These include methods that directly invert backpropagation and algorithms that learn an inverse mapping from labels to features.[28, 38, 83-85] Here, we use a scan across dynamic input variables because of its simplicity.

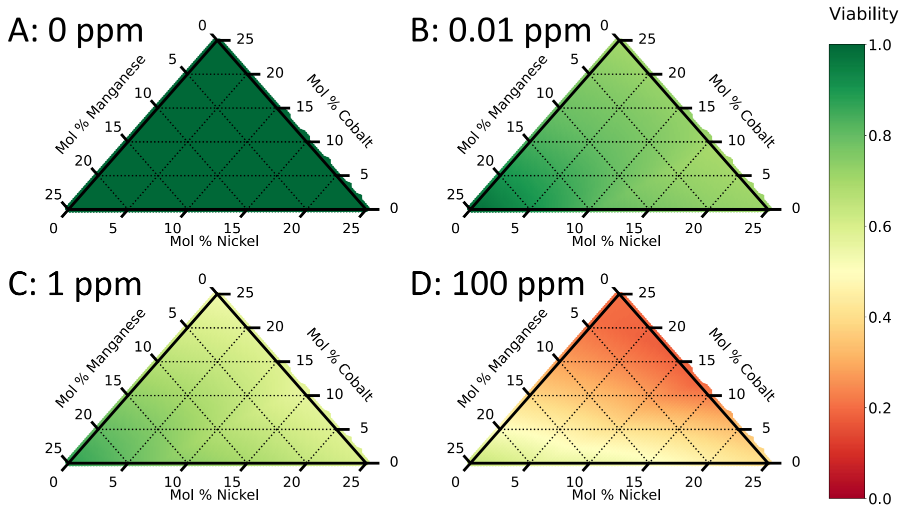


Figure 4: Predicted viability as a function of NMC nanosheet composition at 4 different total concentrations, as indicated. Grid lines reference the tick marks they are connected and parallel to.

A small bag of four ANNs was created using 4FS. Viability predictions were taken to be the average of the predictions of these distinct ANNs. Model inversion was attempted for several classes of nanoparticles — NMC nanosheets, gold nanospheres, and carbon and quantum dots. All predictions in this section are for Gram negative *Shewanella oneidensis* MR-1 bacteria (note that the dataset includes several different types of organisms). As “the dose makes the poison” — i.e. any material is toxic or nontoxic as some exposure level— concentration dependence is evaluated for all considered nanomaterials.[101]

For NMC nanoparticles, the effect of composition is shown in Fig. 4 across four representative cases, varying in concentration of nanoparticles. This scan includes only nanosheets as they have the highest surface area to volume ratio and thus the most dynamic viability response of NMC nanoparticles in the database. The viability is 1.0 at a concentration of 0 ppm regardless of composition, as expected. As the concentration increases, the region towards the right of the graph, corresponding to lower compositions of Mn and mixed compositions of Ni and Co,

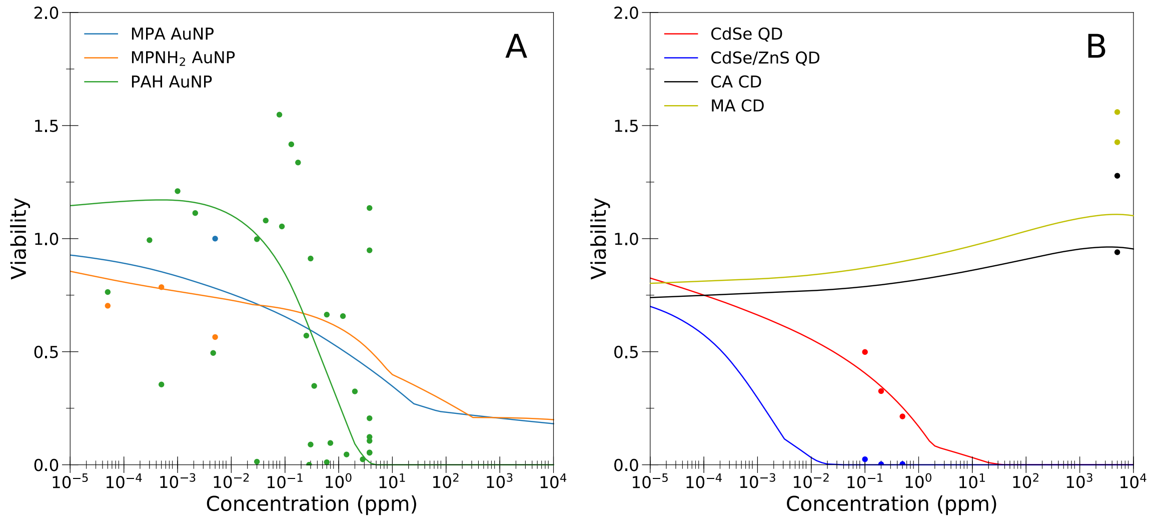


Figure 5: (A) Viability as a function of AuNP coating and concentration. (B) Comparison of concentration dependence of carbon and quantum dots.

decreases in viability most quickly. At 100 ppm, high Mn nanosheets are still clearly preferred. However, mixed Mn/Ni nanoparticles show less toxicity than mixed Mn/Co nanoparticles. In terms of composition, the ANNs predicts toxicity as Mn $<$ Ni $<$ Co. Prior experiments have observed no Ni composition dependent effect on toxicity, which in Fig. 4 would appear as a color gradient along the Ni axis. At the Mn compositions examined in that study (~3 to 10 mol %), the ANN predicts weak Ni dependent effects unless the Ni mol % is particularly high or the NMC concentration is particularly large.[76] This shows that our model is able to make testable viability predictions outside of the range of previously performed experiments, leading to proposed nanotechnology design rules. Other work has shown that increasing the relative composition of Mn and lowering that of Co or Ni decreases the toxicity of NMC nanoparticles, in agreement with the current work.[102] These experiments measured oxygen consumption by bacteria, a type of data that was not considered by this ANN. Thus, our ANN is in agreement with experimental data beyond what it has been trained on.

For gold nanospheres, the effect of nanoparticle coating was also determined from the optimal bag of ANNs. Specifically, the viability in response to several spherical nanoparticles with 4.5 nm (core) diameters was determined as a function of concentration. These values are plotted in Fig. 5A. Shaded areas represent standard error across the four ANNs. The 3-mercaptopropionic acid (MPA) and 3-mercaptopropylamine (MPNH2) nanoparticles are predicted to have largely similar dose-response effects, while the poly(allylamine hydrochloride) (PAH) nanoparticles are expected to be much more toxic to the bacteria at high concentration; these trends are largely in agreement with experiment.[67]

Finally, we compare the concentration-dependent effect on viability of carbon and quantum dots in Fig. 5B. Carbon dots (without phosphorus doping) made with citric acid (CA) or malic acid (MA) precursors are compared to CdSe quantum dots with and without an approximately 2 nm ZnS shell. While the error bars are generally fairly large due to the limitations of the training dataset, it is clear that carbon dots are predicted to have much less toxicity than quantum dots at equivalent concentrations. For carbon dots, changing the precursor material does not have a large effect on viability. For quantum dots, adding a ZnS shell strongly increases the nanoparticle toxicity. These trends are in overall agreement with prior experimental work.[74-75] Moreover, we can now compare between the viability in response to carbon dots and quantum dots as predicted by the bag of ANNs at the same concentrations. Such a direct comparison was not available in the original data and illustrates the interpolative and extrapolative power of the bag of ANNs.

IV. Conclusion

Using a diverse, yet limited database of experimental results, we developed a bag of ANNs for prediction of organism viability upon exposure to nanomaterials. Rigorous validation is undertaken to ensure that the predictions of the ANN families are accurate. The working machine is a bag of ANNs whose consensus value is the prediction. It performs better than chance as quantified against null hypotheses in which the labels are randomly associated with features either for training or evaluation. Its uncertainty is comparable, but no worse, to that of the experiments in the underlying database used to generate it.

The ANNs have been inverted to uncover relationships between the properties of nanoparticles and their toxicity to Gram negative bacteria that agree with prior experimental trends and imply new areas for experimental exploration. These properties —whether previously known or not— were not provided to the ANNs directly. Instead, they were encoded only through whatever correlations exist within the data set provided to the ANN, and hence the use of the ANN to learn them represents a pathway for their discovery. First, we found complex variation in organism viability in response to NMC nanosheets of varying composition. We also found that MPA and $MPNH_2$ coated AuNPs have similar dose-dependent viability profiles, and PAH coated nanoparticles are predicted to lead to more dynamic viability response than either of the other two coated AuNPs. Finally, carbon dots are found to support much higher viability than quantum dots, regardless of concentration or carbon dot composition. Quantum dots with ZnS shells are found to lead to very low viability. Thus, ANNs can be used to uncover rules or principles connecting nanoparticle properties and their effects on viability.

In summary, in this work, we have achieved at least three novel aims: (1) We applied a neural net approach---including bagging---to connect molecular properties to macroscopic observables in a chemical context not attempted before. (ii) We addressed a class of problems for which the data set is often “small” and hence requires some care in applying machine learning to resolve it. (iii) We introduced an approach to use a bagged-neural net approach to both recover prediction and a kind of inversion.

Supporting Information

The Supporting Information is available free of charge at TK.

The Excel file, CSN_DataBase2020.xlsx, includes all of the data in the Center for Sustainable Nanotechnology (CSN) Nanoparticle Viability Database used in this work to construct the ANNs, and cites the corresponding papers from which it was drawn.

Acknowledgements

Data used in this study for the training of ANNs was graciously provided by the CSN through deposits in an internal data repository called the NanoDatabase, from work performed by, but not limited, to Thu Nguyen, Sunipa Pramanik, Peter Clement, Benjamin Frank, Eric Melby, Bo Zhi, Joseph Buchman, Denise N. Williams, Tian Autumn Qiu, Natalie Hudson-Smith, Galya Orr, Howard Fairbrother, Z. Vivian Feng, Rebecca Klaper, and Christy Haynes. They are also acknowledged for providing interpretation and understanding of the diverse data streams. This work was supported by National Science Foundation under the NSF Center for Sustainable Nanotechnology (CSN), CHE-1503408. The CSN is part of the NSF Centers for Chemical Innovation Program.

References

1. Mjolsness, E.; DeCoste, D. Machine Learning for Science: State of the Art and Future Prospects. *Science* **2001,** *293*, 2051-2055.
2. Jordan, M. I.; Mitchell, T. M. Machine Learning: Trends, Perspectives, and Prospects. *Science* **2015,** *349*, 255-260.
3. Chollet, F., *Deep Learning with Python*. 1st ed.; Manning Publications Co.: 2017.
4. Géron, A., *Hands-on Machine Learning with Scikit-Learn and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 1st ed.; O'Reilly Media, Inc.: 2017.
5. Bonaccorso, G., *Machine Learning Algorithms*. Packt Publishing: 2017.

6. Carleo, G.; Cirac, I.; Cranmer, K.; Daudet, L.; Schuld, M.; Tishby, N.; Vogt-Maranto, L.; Zdeborová, L. Machine Learning and the Physical Sciences. *Rev. Mod. Phys.* **2019,** *91*, 045002.
7. Barrington, L.; Turnbull, D.; Lanckriet, G. Game-Powered Machine Learning. *Proc. Natl. Acad. Sci. USA* **2012,** *109*, 6411-6416.
8. Schmuker, M.; Schneider, G. Processing and Classification of Chemical Data Inspired by Insect Olfaction. *Proc. Natl. Acad. Sci. USA* **2007,** *104*, 20285-20289.
9. Belabbas, M.-A.; Wolfe, P. J. Spectral Methods in Machine Learning and New Strategies for Very Large Datasets. *Proc. Natl. Acad. Sci. USA* **2009,** *106*, 369-374.
10. Mellit, A.; Pavan, A. M. A 24-H Forecast of Solar Irradiance Using Artificial Neural Network: Application for Performance Prediction of a Grid-Connected Pv Plant at Trieste, Italy. *Sol. Energy* **2010,** *84*, 807-821.
11. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum Machine Learning. *Nature* **2017,** *549*, 195-202.
12. Adir, O.; Poley, M.; Chen, G.; Froim, S.; Krinsky, N.; Shklover, J.; Shainsky-Roitman, J.; Lammers, T.; Schroeder, A. Integrating Artificial Intelligence and Nanotechnology for Precision Cancer Medicine. *Adv. Mater.* **2019**, 1901989.
13. Butler, K. T.; Davies, D. W.; Cartwright, H.; Isayev, O.; Walsh, A. Machine Learning for Molecular and Materials Science. *Nature* **2018,** *559*, 547-555.
14. Behler, J. Constructing High-Dimensional Neural Network Potentials: A Tutorial Review. *Int. J. Quantum Chem.* **2015,** *115*, 1032-1050.
15. Faber, F.; Lindmaa, A.; Lilienfeld, O. A. v.; Armiento, R. Crystal Structure Representations for Machine Learning Models of Formation Energies. *Int. J. Quantum Chem.* **2015,** *115*, 1094-1101.
16. Ramakrishnan, R.; Dral, P. O.; Rupp, M.; von Lilienfeld, O. A. Big Data Meets Quantum Chemistry Approximations: The Δ-Machine Learning Approach. *J. Chem. Theory Comput.* **2015,** *11*, 2087-2096.
17. Fernandez, M.; Bilić, A.; Barnard, A. S. Machine Learning and Genetic Algorithm Prediction of Energy Differences between Electronic Calculations of Graphene Nanoflakes. *Nanotechnology* **2017,** *28*, 38LT03.
18. Bartók, A. P.; De, S.; Poelking, C.; Bernstein, N.; Kermode, J. R.; Csányi, G.; Ceriotti, M. Machine Learning Unifies the Modeling of Materials and Molecules. *Sci. Adv.* **2017,** *3*, e1701816.
19. Chmiela, S.; Tkatchenko, A.; Sauceda, H. E.; Poltavsky, I.; Schütt, K. T.; Müller, K.-R. Machine Learning of Accurate Energy-Conserving Molecular Force Fields. *Sci. Adv.* **2017,** *3*, e1603015.
20. Carrasquilla, J.; Melko, R. G. Machine Learning Phases of Matter. *Nat. Phys.* **2017,** *13*, 431-434.
21. Ward, L.; Liu, R.; Krishna, A.; Hegde, V. I.; Agrawal, A.; Choudhary, A.; Wolverton, C. Including Crystal Structure Attributes in Machine Learning Models of Formation Energies Via Voronoi Tessellations. *Phys. Rev. B* **2017,** *96*, 024104.
22. Gómez-Bombarelli, R.; Aspuru-Guzik, A., Machine Learning and Big-Data in Computational Chemistry. In *Handbook of Materials Modeling: Methods: Theory and Modeling*, Andreoni, W.; Yip, S., Eds. Springer International Publishing: Cham, 2018; pp 1-24.

23. Wu, Z.; Ramsundar, B.; Feinberg, Evan N.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, K.; Pande, V. Moleculenet: A Benchmark for Molecular Machine Learning. *Chem. Sci.* **2018,** *9*, 513-530.

24. Sifain, A. E.; Lubbers, N.; Nebgen, B. T.; Smith, J. S.; Lokhov, A. Y.; Isayev, O.; Roitberg, A. E.; Barros, K.; Tretiak, S. Discovering a Transferable Charge Assignment Model Using Machine Learning. *J. Phys. Chem. Lett.* **2018,** *9*, 4495-4501.

25. Smith, J. S.; Nebgen, B.; Lubbers, N.; Isayev, O.; Roitberg, A. E. Less Is More: Sampling Chemical Space with Active Learning. *J. Chem. Phys.* **2018,** *148*, 241733.

26. Afzal, M. A. F.; Cheng, C.; Hachmann, J. Combining First-Principles and Data Modeling for the Accurate Prediction of the Refractive Index of Organic Polymers. *J. Chem. Phys.* **2018,** *148*, 241712.

27. Ribeiro, J. M. L.; Bravo, P.; Wang, Y.; Tiwary, P. Reweighted Autoencoded Variational Bayes for Enhanced Sampling (Rave). *J. Chem. Phys.* **2018,** *149*, 072301.

28. Blaschke, T.; Olivecrona, M.; Engkvist, O.; Bajorath, J.; Chen, H. Application of Generative Autoencoder in De Novo Molecular Design. *Mol. Inform.* **2018,** *37*, 1700123.

29. Lamim Ribeiro, J. M.; Tiwary, P. Toward Achieving Efficient and Accurate Ligand-Protein Unbinding with Deep Learning and Molecular Dynamics through Rave. *J. Chem. Theory Comput.* **2019,** *15*, 708-719.

30. Smith, J. S.; Nebgen, B. T.; Zubatyuk, R.; Lubbers, N.; Devereux, C.; Barros, K.; Tretiak, S.; Isayev, O.; Roitberg, A. E. Approaching Coupled Cluster Accuracy with a General-Purpose Neural Network Potential through Transfer Learning. *Nat. Commun.* **2019,** *10*, 2903.

31. Wang, Y.; Ribeiro, J. M. L.; Tiwary, P. Past–Future Information Bottleneck for Sampling Molecular Reaction Coordinate Simultaneously with Thermodynamics and Kinetics. *Nat. Commun.* **2019,** *10*, 3573.

32. Häse, F.; Fdez. Galván, I.; Aspuru-Guzik, A.; Lindh, R.; Vacher, M. How Machine Learning Can Assist the Interpretation of Ab Initio Molecular Dynamics Simulations and Conceptual Understanding of Chemistry. *Chem. Sci.* **2019,** *10*, 2298-2307.

33. Tuckerman, M. E. Machine Learning Transforms How Microstates Are Sampled. *Science* **2019,** *365*, 982-983.

34. Afzal, M. A. F.; Haghighatlari, M.; Ganesh, S. P.; Cheng, C.; Hachmann, J. Accelerated Discovery of High-Refractive-Index Polyimides Via First-Principles Molecular Modeling, Virtual High-Throughput Screening, and Data Mining. *J. Phys. Chem. C* **2019,** *123*, 14610-14618.

35. Noe, F.; Olsson, S.; Kohler, J.; Wu, H. Boltzmann Generators: Sampling Equilibrium States of Many-Body Systems with Deep Learning. *Science* **2019,** *365*, eaaw1147.

36. Roy, K.; Kar, S.; Das, R. N., Chapter 1 - Background of QSAR and Historical Developments. In *Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment*, Roy, K.; Kar, S.; Das, R. N., Eds. Academic Press: Boston, 2015; pp 1-46.

37. Raccuglia, P.; Elbert, K. C.; Adler, P. D. F.; Falk, C.; Wenny, M. B.; Mollo, A.; Zeller, M.; Friedler, S. A.; Schrier, J.; Norquist, A. J. Machine-Learning-Assisted Materials Discovery Using Failed Experiments. *Nature* **2016,** *533*, 73-76.

38. Miyao, T.; Kaneko, H.; Funatsu, K. Inverse QSPR/QSAR Analysis for Chemical Structure Generation (from y to x). *J. Chem. Inf. Model.* **2016,** *56*, 286-299.

39. Chowdhury, A. J.; Yang, W.; Walker, E.; Mamun, O.; Heyden, A.; Terejanu, G. A. Prediction of Adsorption Energies for Chemical Species on Metal Catalyst Surfaces Using Machine Learning. *J. Phys. Chem. C* **2018,** *122*, 28142-28150.

40. Simón-Vidal, L.; García-Calvo, O.; Oteo, U.; Arrasate, S.; Lete, E.; Sotomayor, N.; González-Díaz, H. Perturbation-Theory and Machine Learning (Ptml) Model for High-Throughput Screening of Parham Reactions: Experimental and Theoretical Studies. *J. Chem. Inf. Model.* **2018,** *58*, 1384-1396.

41. Bartel, C. J.; Millican, S. L.; Deml, A. M.; Rumptz, J. R.; Tumas, W.; Weimer, A. W.; Lany, S.; Stevanović, V.; Musgrave, C. B.; Holder, A. M. Physical Descriptor for the Gibbs Energy of Inorganic Crystalline Solids and Temperature-Dependent Materials Chemistry. *Nat. Commun.* **2018,** *9*, 4168.

42. Coley, C. W.; Green, W. H.; Jensen, K. F. Machine Learning in Computer-Aided Synthesis Planning. *Acc. Chem. Res.* **2018,** *51*, 1281-1289.

43. Sanchez-Lengeling, B.; Aspuru-Guzik, A. Inverse Molecular Design Using Machine Learning: Generative Models for Matter Engineering. *Science* **2018,** *361*, 360-365.

44. Ahneman, D. T.; Estrada, J. G.; Lin, S.; Dreher, S. D.; Doyle, A. G. Predicting Reaction Performance in C–N Cross-Coupling Using Machine Learning. *Science* **2018,** *360*, 186-190.

45. Sanchez-Lengeling, B.; Roch, L. M.; Perea, J. D.; Langner, S.; Brabec, C. J.; Aspuru-Guzik, A. A Bayesian Approach to Predict Solubility Parameters. *Adv. Theory Simul.* **2019,** *2*, 1800069.

46. Hachmann, J.; Afzal, M. A. F.; Haghighatlari, M.; Pal, Y. Building and Deploying a Cyberinfrastructure for the Data-Driven Design of Chemical Systems and the Exploration of Chemical Space. *Mol. Simulat.* **2018,** *44*, 921-929.

47. Popova, M.; Isayev, O.; Tropsha, A. Deep Reinforcement Learning for De Novo Drug Design. *Sci. Adv.* **2018,** *4*, eaap7885.

48. Aykol, M.; Hegde, V. I.; Hung, L.; Suram, S.; Herring, P.; Wolverton, C.; Hummelshøj, J. S. Network Analysis of Synthesizable Materials Discovery. *Nat. Commun.* **2019,** *10*, 2018-2018.

49. Zhang, Y.; Mesaros, A.; Fujita, K.; Edkins, S. D.; Hamidian, M. H.; Ch'ng, K.; Eisaki, H.; Uchida, S.; Davis, J. C. S.; Khatami, E., et al. Machine Learning in Electronic-Quantum-Matter Imaging Experiments. *Nature* **2019,** *570*, 484-490.

50. Varsou, D.-D.; Tsoumanis, A.; Afantitis, A.; Melagraki, G., Enalos Cloud Platform: Nanoinformatics and Cheminformatics Tools. In *Ecotoxicological Qsars*, Roy, K., Ed. Springer US: New York, NY, 2020; pp 789-800.

51. Toyao, T.; Maeno, Z.; Takakusagi, S.; Kamachi, T.; Takigawa, I.; Shimizu, K.-i. Machine Learning for Catalysis Informatics: Recent Applications and Prospects. *ACS Catal.* **2020,** *10*, 2260-2297.

52. Dreiseitl, S.; Ohno-Machado, L. Logistic Regression and Artificial Neural Network Classification Models: A Methodology Review. *J. Biomed. Inform.* **2002,** *35*, 352-359.

53. Svetnik, V.; Liaw, A.; Tong, C.; Culberson, J. C.; Sheridan, R. P.; Feuston, B. P. Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. *J. Chem. Inf. Comput. Sci.* **2003,** *43*, 1947-1958.

54. Grömping, U. Variable Importance Assessment in Regression: Linear Regression Versus Random Forest. *Am. Stat.* **2009,** *63*, 308-319.

55. Weston, J.; Ratle, F.; Mobahi, H.; Collobert, R., Deep Learning Via Semi-Supervised Embedding. In *Neural Networks: Tricks of the Trade: Second Edition*, Montavon, G.; Orr, G. B.; Müller, K.-R., Eds. Springer Berlin Heidelberg: Berlin, Heidelberg, 2012; pp 639-655.
56. Liu, R.; Cohen, Y. Nanoinformatics for Environmental Health and Biomedicine. *Beilstein J. Nanotechnol.* **2015,** *6*, 2449-2451.
57. Jones, D. E.; Ghandehari, H.; Facelli, J. C. A Review of the Applications of Data Mining and Machine Learning for the Prediction of Biomedical Properties of Nanoparticles. *Comput. Meth. Prog. Bio.* **2016,** *132*, 93-103.
58. Kleandrova, V. V.; Feng, L.; Speck-Planche, A.; Cordeiro, M. N. D. S., QSAR-Based Studies of Nanomaterials in the Environment. In *Pharmaceutical Sciences: Breakthroughs in Research and Practice*, IGI Global: Hershey, PA, USA, 2017; pp 1339-1366.
59. Concu, R.; Kleandrova, V. V.; Speck-Planche, A.; Cordeiro, M. N. D. S. Probing the Toxicity of Nanoparticles: A Unified in Silico Machine Learning Model Based on Perturbation Theory. *Nanotoxicology* **2017,** *11*, 891-906.
60. Findlay, M. R.; Freitas, D. N.; Mobed-Miremadi, M.; Wheeler, K. E. Machine Learning Provides Predictive Analysis into Silver Nanoparticle Protein Corona Formation from Physicochemical Properties. *Environ. Sci. Nano* **2018,** *5*, 64-71.
61. Ha, M. K.; Trinh, T. X.; Choi, J. S.; Maulina, D.; Byun, H. G.; Yoon, T. H. Toxicity Classification of Oxide Nanomaterials: Effects of Data Gap Filling and Pchem Score-Based Screening Approaches. *Sci. Rep.* **2018,** *8*, 3141.
62. González-Durruthy, M.; Manske Nunes, S.; Ventura-Lima, J.; Gelesky, M. A.; González-Díaz, H.; Monserrat, J. M.; Concu, R.; Cordeiro, M. N. D. S. MitoTarget Modeling Using ANN-Classification Models Based on Fractal SEM Nano-Descriptors: Carbon Nanotubes as Mitochondrial F0F1-ATPase Inhibitors. *J. Chem. Inf. Model.* **2019,** *59*, 86-97.
63. Halder, A. K.; Melo, A.; Cordeiro, M. N. D. S. A Unified in Silico Model Based on Perturbation Theory for Assessing the Genotoxicity of Metal Oxide Nanoparticles. *Chemosphere* **2020,** *244*, 125489.
64. Afantitis, A.; Melagraki, G.; Isigonis, P.; Tsoumanis, A.; Varsou, D. D.; Valsami-Jones, E.; Papadiamantis, A.; Ellis, L.-J. A.; Sarimveis, H.; Doganis, P., et al. NanoSolveIT Project: Driving Nanoinformatics Research to Develop Innovative and Integrated Tools for in Silico Nanosafety Assessment. *Comput. Struct. Biotechnol. J.* **2020,** *18*, 583-602.
65. Findlay, M. R.; Freitas, D. N.; Mobed-Miremadi, M.; Wheeler, K. E. Machine Learning Provides Predictive Analysis into Silver Nanoparticle Protein Corona Formation from Physicochemical Properties. *Environ. Sci.: Nano* **2018,** *5*, 64-71.
66. Labouta, H. I.; Asgarian, N.; Rinker, K.; Cramb, D. T. Meta-Analysis of Nanoparticle Cytotoxicity Via Data-Mining the Literature. *ACS Nano* **2019,** *13*, 1583-1594.
67. Feng, Z. V.; Gunsolus, I. L.; Qiu, T. A.; Hurley, K. R.; Nyberg, L. H.; Frew, H.; Johnson, K. P.; Vartanian, A. M.; Jacob, L. M.; Lohse, S. E., et al. Impacts of Gold Nanoparticle Charge and Ligand Type on Surface Binding and Toxicity to Gram-Negative and Gram-Positive Bacteria. *Chem. Sci.* **2015,** *6*, 5186-5196.
68. Mensch, A. C.; Hernandez, R. T.; Kuether, J. E.; Torelli, M. D.; Feng, Z. V.; Hamers, R. J.; Pedersen, J. A. Natural Organic Matter Concentration Impacts the Interaction of Functionalized Diamond Nanoparticles with Model and Actual Bacterial Membranes. *Environ. Sci. Technol.* **2017,** *51*, 11075-11084.

69. Qiu, T. A.; Meyer, B. M.; Christenson, K. G.; Klaper, R. D.; Haynes, C. L. A Mechanistic Study of Tio2 Nanoparticle Toxicity on Shewanella Oneidensis MR-1 with Uv-Containing Simulated Solar Irradiation: Bacterial Growth, Riboflavin Secretion, and Gene Expression. *Chemosphere* **2017,** *168*, 1158-1168.

70. Qiu, T. A.; Nguyen, T. H. T.; Hudson-Smith, N. V.; Clement, P. L.; Forester, D.-C.; Frew, H.; Hang, M. N.; Murphy, C. J.; Hamers, R. J.; Feng, Z. V., et al. Growth-Based Bacterial Viability Assay for Interference-Free and High-Throughput Toxicity Screening of Nanomaterials. *Anal. Chem.* **2017,** *89*, 2057-2064.

71. Qiu, T. A.; Torelli, M. D.; Vartanian, A. M.; Rackstraw, N. B.; Buchman, J. T.; Jacob, L. M.; Murphy, C. J.; Hamers, R. J.; Haynes, C. L. Quantification of Free Polyelectrolytes Present in Colloidal Suspension, Revealing a Source of Toxic Responses for Polyelectrolyte-Wrapped Gold Nanoparticles. *Anal. Chem.* **2017,** *89*, 1823-1830.

72. Hang, M. N.; Hudson-Smith, N. V.; Clement, P. L.; Zhang, Y.; Wang, C.; Haynes, C. L.; Hamers, R. J. Influence of Nanoparticle Morphology on Ion Release and Biological Impact of Nickel Manganese Cobalt Oxide (NMC) Complex Oxide Nanomaterials. *ACS Appl. Nano Mater.* **2018,** *1*, 1721-1730.

73. Melby, E. S.; Cui, Y.; Borgatta, J.; Mensch, A. C.; Hang, M. N.; Chrisler, W. B.; Dohnalkova, A.; Van Gilder, J. M.; Alvarez, C. M.; Smith, J. N., et al. Impact of Lithiated Cobalt Oxide and Phosphate Nanoparticles on Rainbow Trout Gill Epithelial Cells. *Nanotoxicology* **2018,** *12*, 1166-1181.

74. Williams, D. N.; Pramanik, S.; Brown, R. P.; Zhi, B.; McIntire, E.; Hudson-Smith, N. V.; Haynes, C. L.; Rosenzweig, Z. Adverse Interactions of Luminescent Semiconductor Quantum Dots with Liposomes and Shewanella Oneidensis. *ACS Appl. Nano Mater.* **2018,** *1*, 4788-4800.

75. Zhi, B.; Gallagher, M. J.; Frank, B. P.; Lyons, T. Y.; Qiu, T. A.; Da, J.; Mensch, A. C.; Hamers, R. J.; Rosenzweig, Z.; Fairbrother, D. H., et al. Investigation of Phosphorous Doping Effects on Polymeric Carbon Dots: Fluorescence, Photostability, and Environmental Impact. *Carbon* **2018,** *129*, 438-449.

76. Buchman, J.; Bennett, E.; Wang, C.; Abbaspour Tamijani, A.; Bennett, J.; Hudson, B.; Green, C.; Clement, P.; Zhi, B.; Henke, A., et al. Nickel Enrichment of Next-Generation NMC Nanomaterials Alters Material Stability, Causing Unexpected Dissolution Behavior and Observed Toxicity to S. Oneidensis MR-1 and D. Magna. *Environ. Sci. Nano* **2020,** *7*, 571-587.

77. Ambure, P.; Halder, A. K.; González Díaz, H.; Cordeiro, M. N. D. S. QSAR-Co: An Open Source Software for Developing Robust Multitasking or Multitarget Classification-Based QSAR Models. *J. Chem. Inf. Model.* **2019,** *59*, 2538-2544.

78. Breiman, L. Bagging Predictors. *Mach. Learn.* **1996,** *24*, 123-140.

79. Opitz, D. W.; Maclin, R. F. In *An Empirical Evaluation of Bagging and Boosting for Artificial Neural Networks*, Proceedings of International Conference on Neural Networks (ICNN'97), 12-12 June 1997; 1997; pp 1401-1405 vol.3.

80. Cunningham, P.; Carney, J.; Jacob, S. Stability Problems with Artificial Neural Networks and the Ensemble Solution. *Artif. Intell. Med..* **2000,** *20*, 217-225.

81. Moretti, F.; Pizzuti, S.; Panzieri, S.; Annunziato, M. Urban Traffic Flow Forecasting through Statistical and Neural Network Bagging Ensemble Hybrid Modeling. *Neurocomputing* **2015,** *167*, 3-7.

82. Khwaja, A. S.; Naeem, M.; Anpalagan, A.; Venetsanopoulos, A.; Venkatesh, B. Improved Short-Term Load Forecasting Using Bagged Neural Networks. *Electr. Pow. Syst. Res.* **2015,** *125*, 109-115.

83. Jensen, C. A.; Reed, R. D.; Marks, R. J.; El-Sharkawi, M. A.; Jae-Byung, J.; Miyamoto, R. T.; Anderson, G. M.; Eggen, C. J. Inversion of Feedforward Neural Networks: Algorithms and Applications. *Proc. IEEE* **1999,** *87*, 1536-1549.

84. Mousavi, A.; Baraniuk, R. G. In *Learning to Invert: Signal Recovery Via Deep Convolutional Networks*, 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 5-9 March 2017; 2017; pp 2272-2276.

85. Shrikumar, A.; Greenside, P.; Kundaje, A. Learning Important Features through Propagating Activation Differences. *PMLR* **2017,** *70*, 3145-3153.

86. Chollet, F. *Keras*, GitHub: https://github.com/fchollet/keras, 2015.

87. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M., et al. Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* **2016,** *arXiv:1603.04467*.

88. Oliphant, T. Python for Scientific Computing. *Comput. Sci. Eng.* **2001,** *9*, 10-20.

89. Hunter, J. D. Matplotlib: A 2d Graphics Environment. *Comput. Sci. Eng.* **2007,** *9*, 90-95.

90. McKinney, W. In *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, Walt, S. v. d.; Millman, J., Eds. 2010; pp 51-56.

91. Walt, S. v. d.; Colbert, S. C.; Varoquaux, G. The Numpy Array: A Structure for Efficient Numerical Computation. *Comput. Sci. Eng.* **2011,** *13*, 22-30.

92. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V., et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011,** *12*, 2825-2830.

93. Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J.; Grout, J.; Corlay, S., et al., Jupyter Notebooks – a Publishing Format for Reproducible Computational Workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, Loizides, F.; Schmidt, B., Eds. IOS Press: 2016; pp 87-90.

94. Harper, M.; Weinstein, B.; tgwoodcock; Simon, C.; chebee7i; Morgan, W.; Knight, V.; Swanson-Hysell, N.; Evans, M.; jl, b., et al. *Marcharper/Python-Ternary: Version 1.0.6*, GitHub: https://github.com/marcharper/python-ternary, 2019.

95. Alaya, M. Z.; Bussy, S.; Gaïffas, S.; Guilloux, A. Binarsity: A Penalization for One-Hot Encoded Features in Linear Supervised Learning. *arXiv* **2017,** *1703.08619*.

96. Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014,** *1412.6980v9*.

97. Fushiki, T. Estimation of Prediction Error by Using K-Fold Cross-Validation. *Stat. Comput.* **2009,** *21*, 137-146.

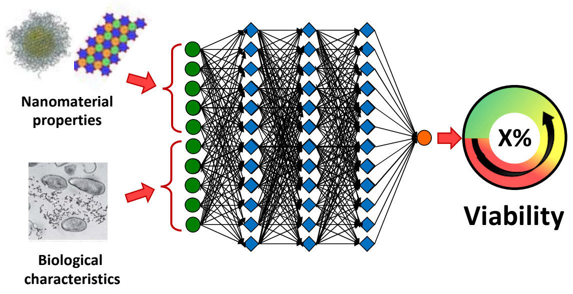
98. Jiang, P.; Chen, J. Displacement Prediction of Landslide Based on Generalized Regression Neural Networks with K-Fold Cross-Validation. *Neurocomputing* **2016,** *198*, 40-47.

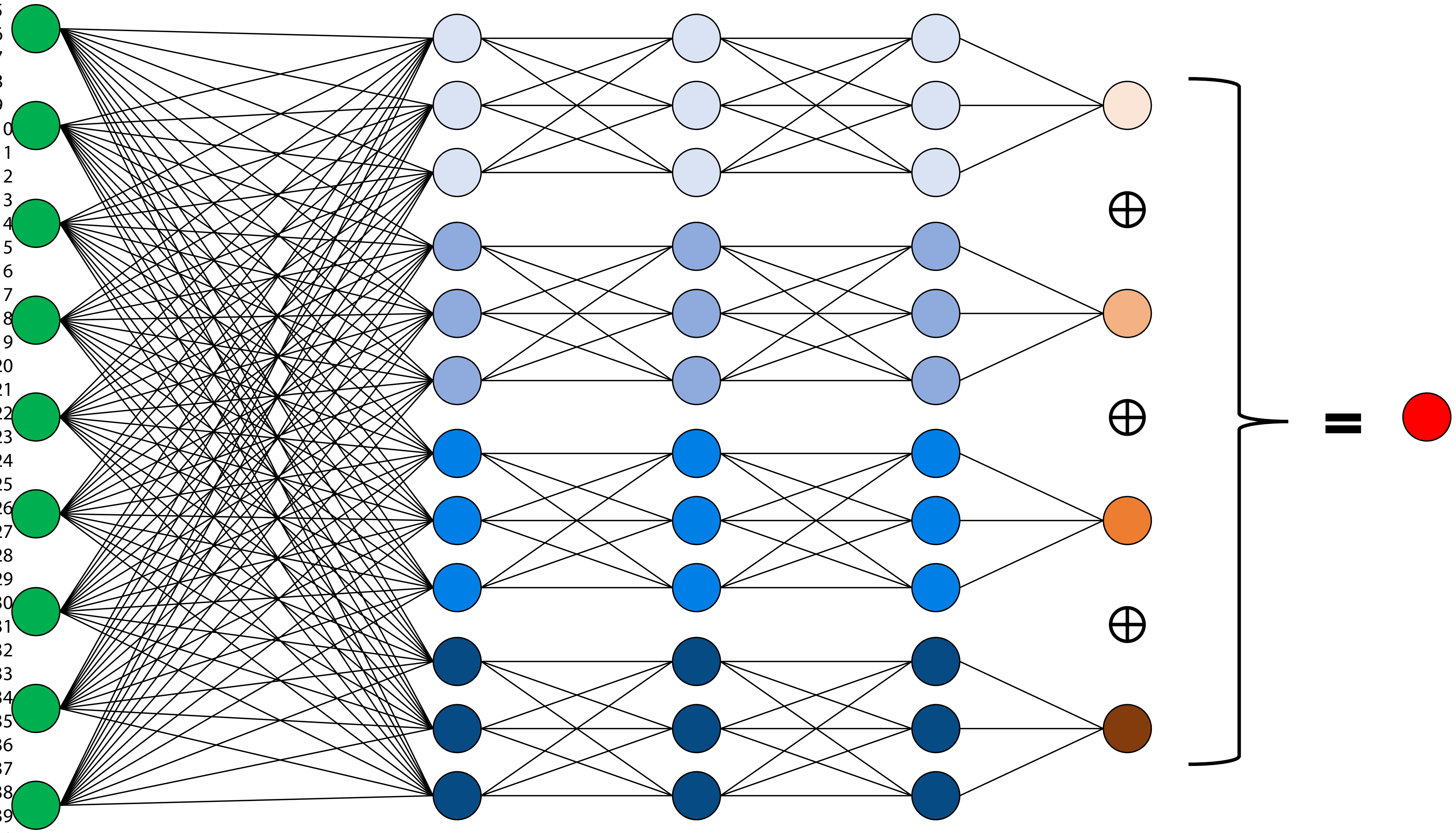
99. Barrow, D. K.; Crone, S. F., Crogging (Cross-Validation Aggregation) for Forecasting — a Novel Algorithm of Neural Network Ensembles on Time Series Subsamples. In *The 2013 International Joint Conference on Neural Networks*, 2013.

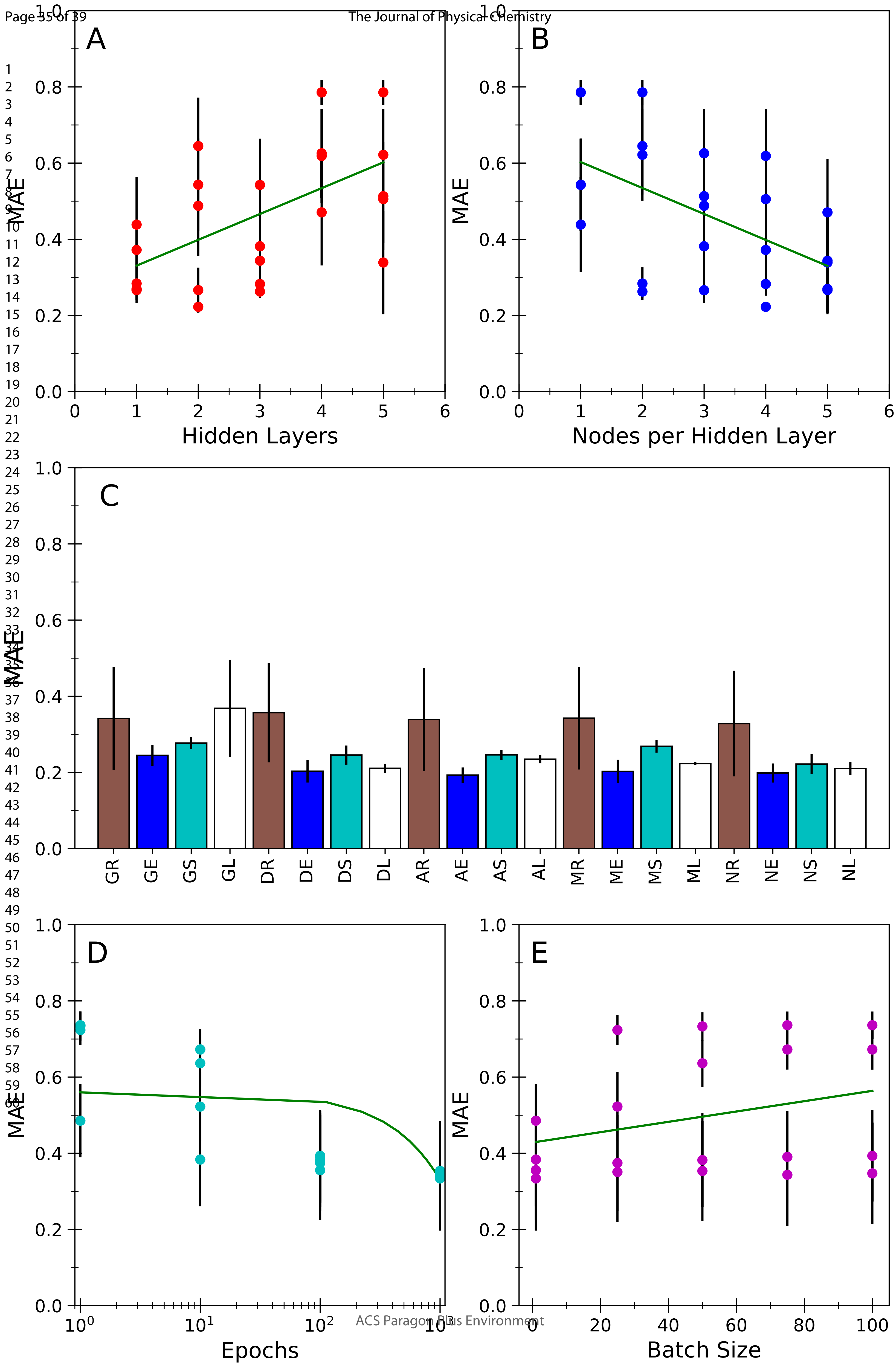
100. Bengio, Y.; Grandvalet, Y. No Unbiased Estimator of the Variance of K-Fold Cross-Validation. *J. Mach. Learn. Res.* **2004,** *5*, 1089-1105.

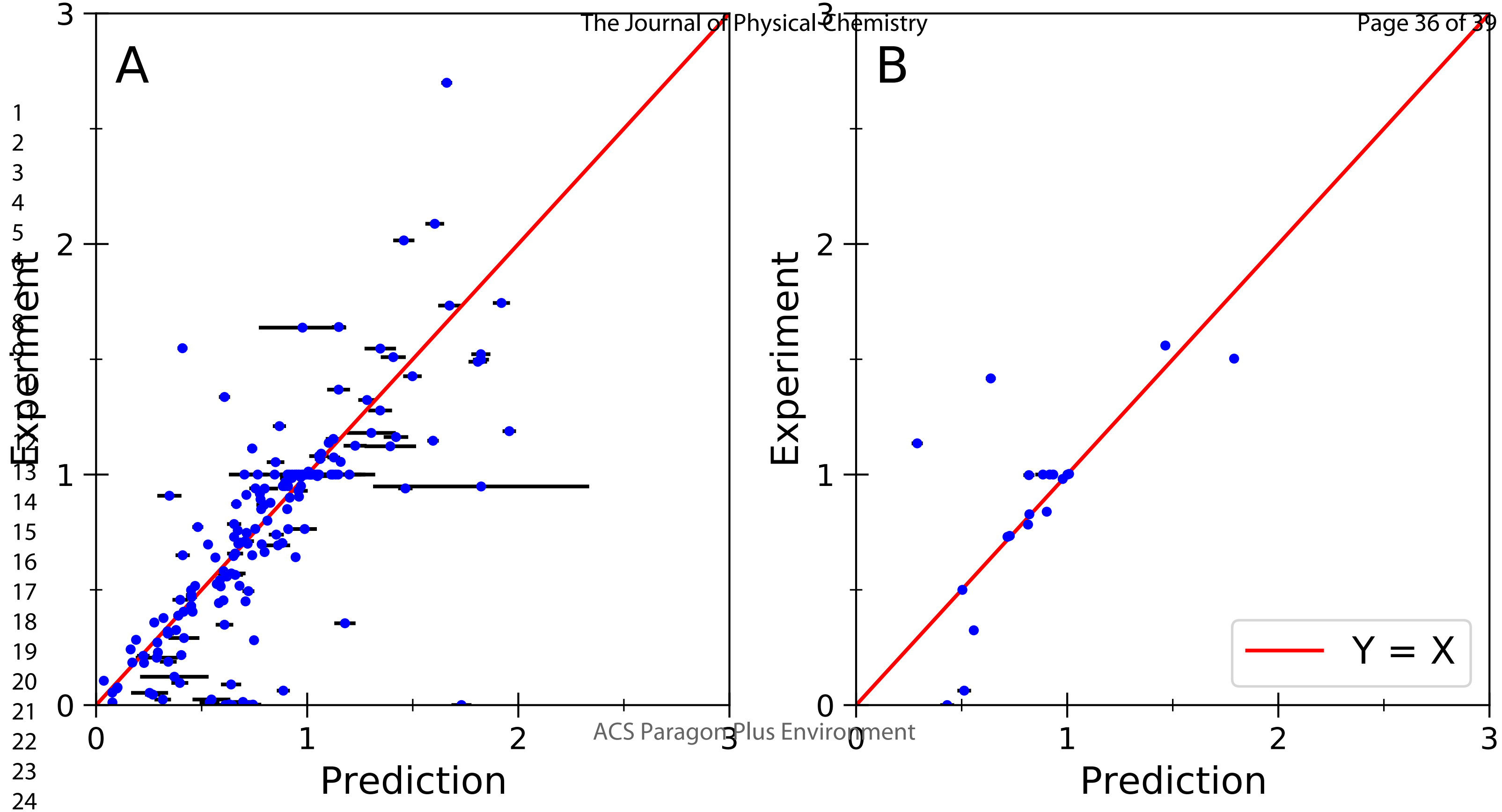
101. Frank, P.; Ottoboni, M. A., *The Dose Makes the Poison: A Plain-Language Guide to Toxicology*. John Wiley & Sons: 2011.
102. Gunsolus, I. L.; Hang, M. N.; Hudson-Smith, N. V.; Buchman, J. T.; Bennett, J. W.; Conroy, D.; Mason, S. E.; Hamers, R. J.; Haynes, C. L. Influence of Nickel Manganese Cobalt Oxide Nanoparticle Composition on Toxicity toward Shewanella Oneidensis MR-1: Redesigning for Reduced Biological Impact. *Environ. Sci. Nano* **2017,** *4*, 636-646.

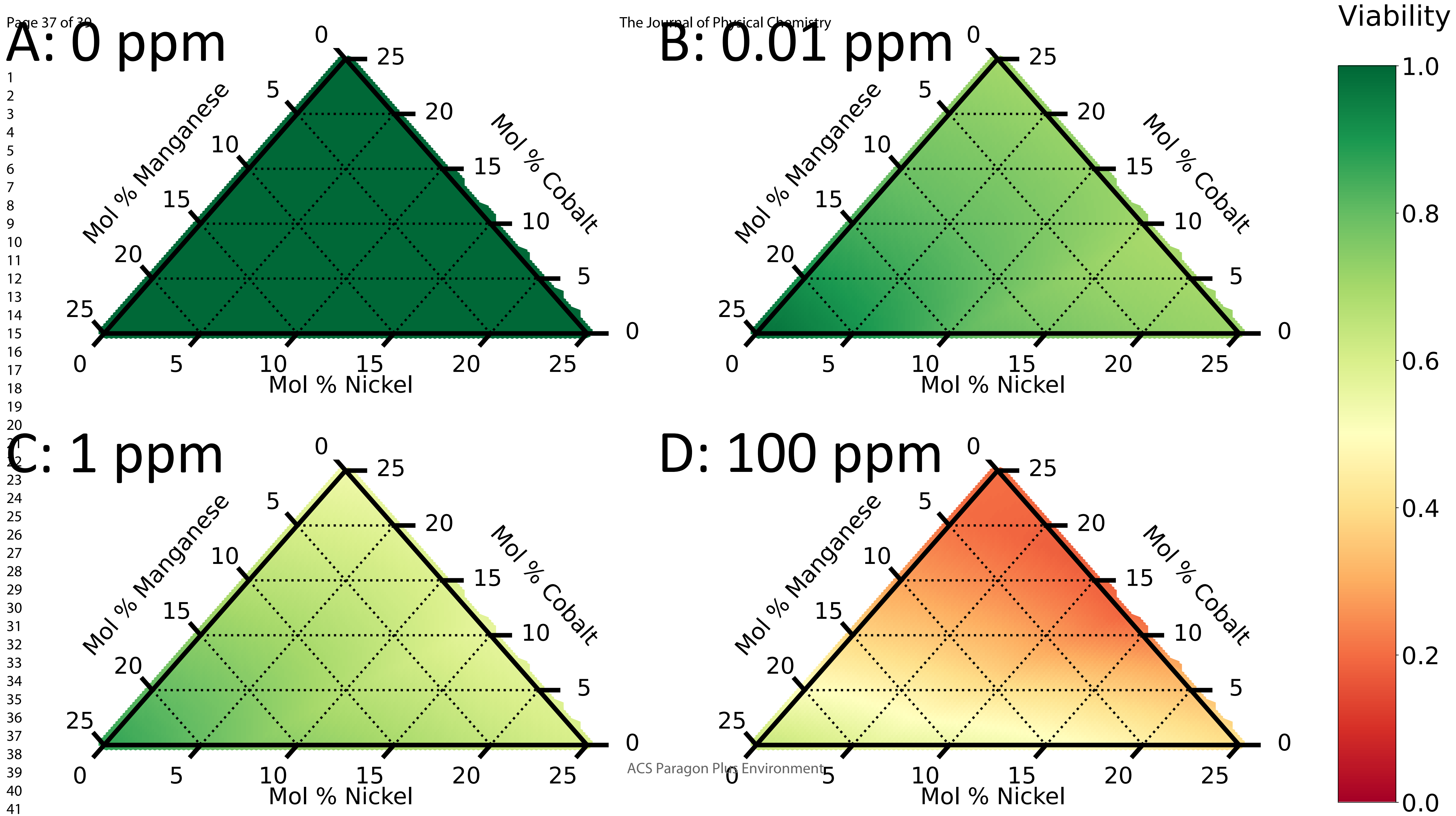
TOC Graphic

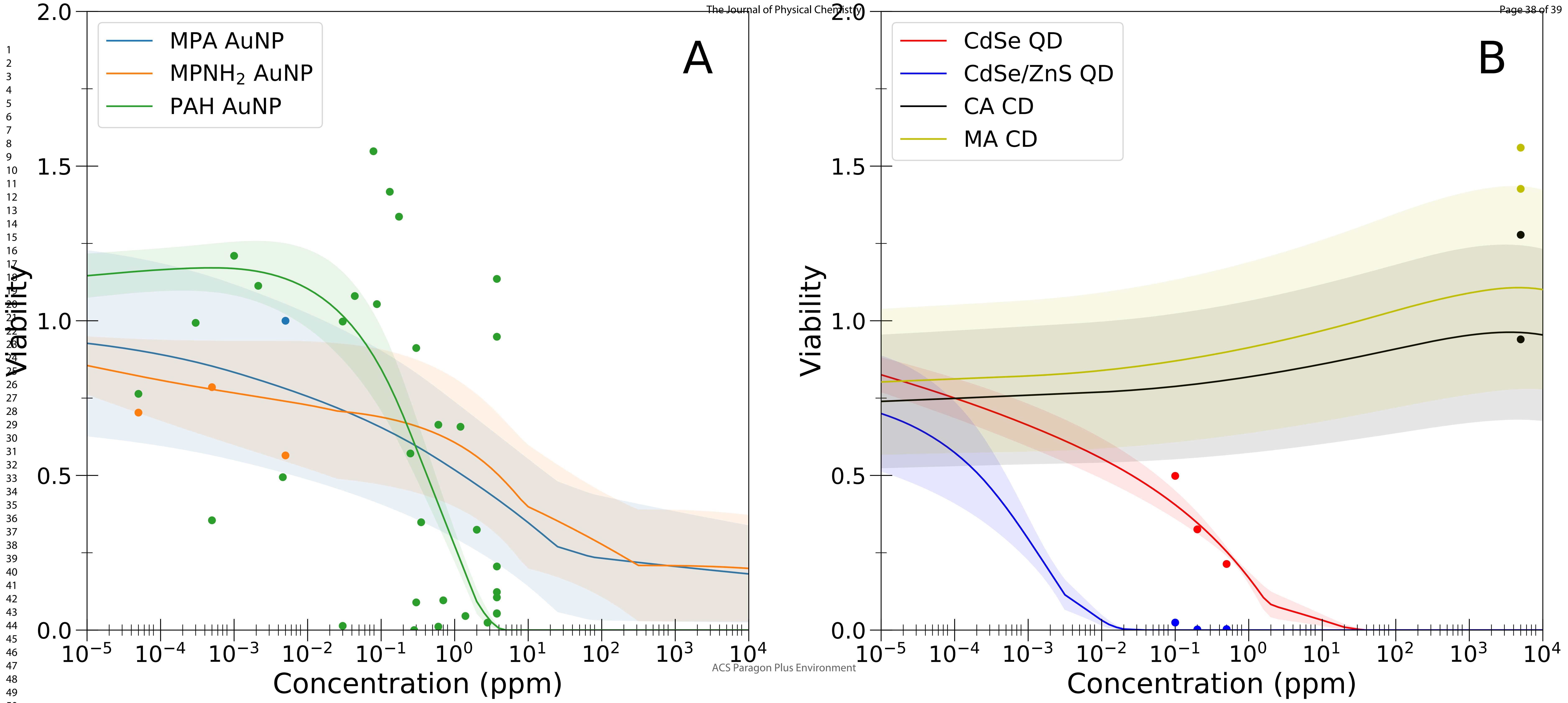

Nanomaterial properties
Biological characteristics
X%
Viability



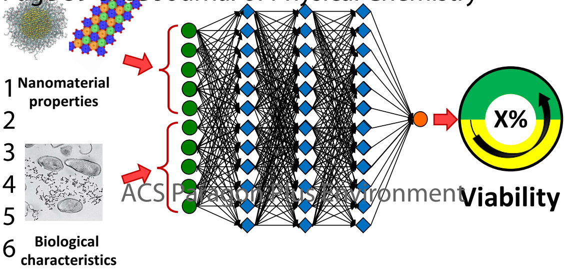

A
B
C
D
E
MAE
Hidden Layers
Nodes per Hidden Layer
Epochs
Batch Size
GR
GE
GS
GL
DR
DE
DS
DL
AR
AE
AS
AL
MR
ME
MS
ML
NR
NE
NS
NL


A
B
Experiment
Prediction
0
1
2
3
Y = X


A: 0 ppm
B: 0.01 ppm
C: 1 ppm
D: 100 ppm
Mol % Manganese
Mol % Cobalt
Mol % Nickel
0
5
10
15
20
25
Viability
1.0
0.8
0.6
0.4
0.2
0.0



A
MPA AuNP
MPNH$_2$ AuNP
PAH AuNP
Viability
Concentration (ppm)
B
CdSe QD
CdSe/ZnS QD
CA CD
MA CD
Viability
Concentration (ppm)


Nanomaterial properties
Biological characteristics
X%
Viability