Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/compbiomed

AweGNN: Auto-parametrized weighted element-specific graph neural networks for molecules

Timothy Szocinski [a], Duc Duy Nguyen [b], Guo-Wei Wei [a,c,d,*]

[a] Department of Mathematics, Michigan State University, MI, 48824, USA
[b] Department of Mathematics, University of Kentucky, KY, 40506, USA
[c] Department of Biochemistry and Molecular Biology, Michigan State University, MI, 48824, USA
[d] Department of Electrical and Computer Engineering, Michigan State University, MI, 48824, USA

ARTICLE INFO

Keywords:
Automated feature extraction
Deep neural network
Mathematical representation
Toxicity
Solvation

ABSTRACT

While automated feature extraction has had tremendous success in many deep learning algorithms for image analysis and natural language processing, it does not work well for data involving complex internal structures, such as molecules. Data representations via advanced mathematics, including algebraic topology, differential geometry, and graph theory, have demonstrated superiority in a variety of biomolecular applications, however, their performance is often dependent on manual parametrization. This work introduces the auto-parametrized weighted element-specific graph neural network, dubbed AweGNN, to overcome the obstacle of this tedious parametrization process while also being a suitable technique for automated feature extraction on these internally complex biomolecular data sets. The AweGNN is a neural network model based on geometric-graph features of element-pair interactions, with its graph parameters being updated throughout the training, which results in what we call a network-enabled automatic representation (NEAR). To enhance the predictions with small data sets, we construct multi-task (MT) AweGNN models in addition to single-task (ST) AweGNN models. The proposed methods are applied to various benchmark data sets, including four data sets for quantitative toxicity analysis and another data set for solvation prediction. Extensive numerical tests show that AweGNN models can achieve state-of-the-art performance in molecular property predictions.

1. Introduction

Automated feature extraction techniques, like those used in convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been very successful in deep learning applications [17,32]. They have made inroads in a variety of fields now, producing state-of-the-art results in signal and information processing fields [10], such as speech recognition, image recognition [8], and natural language processing [35]. These kinds of automated feature extraction techniques, however, are only highly successful on data that is relatively simple in structure, such as with images, text, etc. For data sets with complex internal structures, such as molecules or macro-molecules, hand-crafted descriptors, or representations, are indispensable for developing top-quality predictive models [25].

There are many physical, biological and man-made objects that have intricate internal structures. For example, proteins, chromosomes, human bodies, and cities have very complex structures. Tools from abstract mathematics such as algebraic topology, differential geometry, and combinatorics can be utilized to simplify the structural complexities of data [23,25,39]. For molecules and macro-molecules, molecular fingerprint representations obtained from persistent homology, curvature analysis of surface electrostatic potentials, and eigenvalues of weighted adjacency matrices derived from atomic distances have all be used for this endeavor [25]. From the point view of machine learning, molecular representations which include detailed chemical and physical information can be extremely high-dimensional, especially when the biomolecules in question are made of thousands of atoms, such as in the case of protein-ligand complexes [2,5,6,9]. By using lower dimensional mathematical representations to train simple machine learning models, such as gradient-boosting trees (GBT) and random forest (RF) models, one can achieve stellar performance.

Some of our hand-crafted mathematical features are somewhat limited in their scope [40], in which we mean they are generated by a fixed procedure, with little possibility for making fine adjustments to

* Corresponding author. Department of Mathematics, Michigan State University, MI, 48824, USA.
E-mail address: wei@math.msu.edu (G.-W. Wei).

https://doi.org/10.1016/j.compbiomed.2021.104460
Received 11 February 2021; Received in revised form 23 April 2021; Accepted 26 April 2021
Available online 12 May 2021

produce more favorable representations. Other mathematical features that we have used are based on a choice of kernel function or functions along with adjustable parameters, in which a careful tuning of these parameters can deliver optimal performance [26,28]. This process of tuning, however, can be very time-consuming and requires experience. Although the kernel parameters introduce new dimensions to enhance our models, the cost of tweaking becomes tedious when more than a small handful are introduced.

In this work, motivated by the automation of simple feature-extraction techniques as in CNNs, we seek to automate the parameter optimization for our mathematical representations. Specifically, we automate the selection process of kernel parameters either partially or fully, meaning that we would like to have some or all of the parameters automatically chosen for us during training. Neural networks are trained in a series of epochs where the weights of the network are updated at each step through gradient descent, and so they are a perfect starting point for exploring this idea of automated parameter selection. In our approach, back propagation is extended further to include the kernel parameters of our molecular representations, leading to the simultaneous update of our kernel parameters and neural weights at each batch of training. To this end, we introduce auto-parametrized weighted element-specific graph neural networks (AweGNNs). The AweGNN implements the above scheme of parameter updates by using a representation based on kernel-weighted geometric subgraphs, with features similar to the correlation functions of flexibility-rigidity index (FRI) used in a previous work [28,41]. The representations resulting from updating the kernel parameters through training the network are then referred to as network-enabled automatic representations (NEARs). To validate these NEARs, we implement them on RF and GBT models, leading to accurate predictions. We further show that ensemble learning can in some cases bolster our network predictions through consensus.

To test our AweGNN models, we employ four toxicity data sets of various sizes. Toxicity is a measure of the degree to which a chemical can harm an organism [40]. The harmful effects can be measured qualitatively or quantitatively. A qualitative approach only categorizes chemicals as toxic or nontoxic, while quantitative toxicity data records the minimal amount of a substance that would produce lethal effects. Experimental measurement of toxicity is expensive, time-consuming, and subject to ethical constraints. In this light, machine learning models are extremely useful in that they do not have these same challenges. The working principal of the machine learning approach for toxicity analysis is that molecules with similar structures have similar activities, which is called the quantitative structure-activity relationship (QSAR) approach. By analyzing the relationship between molecular structures, one can predict their biological functions. We can create AweGNN models that predict these toxicity endpoints without having to conduct any lab experiments [3,14,19,33]. To further improve our AweGNN predictions on the sparsity of data [13], we construct multi-task (MT) AweGNNs. MT learning or transfer learning [7] learns from related tasks to improve the performance on the smaller data sets in particular. It is frequently used to compensate if there are similar data sets at hand. We find that AweGNN models perform well on these data sets when we compare our results to state-of-the-art QSAR techniques, such as the ones that were pursued by our group [27,40] previously and by others [21,22], with particularly excellent performance on the larger data sets.

To further showcase our AweGNN, we apply our models on a different data set; a small data set concerning the solvation of molecules. Solvation free energy is a very important quantity in solvation analysis, which can be very beneficial in studying other complex or biological processes [18,20,24,37]. This data set is relatively small compared to the toxicity data sets, but we found that the AweGNN was able to perform extremely well, beating all other methods in the literature [27]. This proves that the AweGNN is useful in application to molecular problems that are not just restricted to toxicity analysis. In addition to this, we can find that the concept of the AweGNN can be extended further to other kernel-based methods, in which we can automate the kernel parameters as we train a network, such as with the differential geometry work done by Nguyen et al. [27]. Finally, we note that although we have not used multi-scale approaches in this work, we have seen them perform well in previous work [27,28], so it is likely to be worthy of exploration in further works.

2. Theory and methods

In this section, we will briefly review single and multi-task neural networks, then describe our AweGNN models. We discuss AweGNN through the frameworks of element-specific geometric graph representations, dynamic normalization functions, explicit derivative calculations, and instructions on how to update the parameters. Some variations on the generation of features are included even though these variations might have not been used to train the final models.

2.1. Neural networks

Neural networks are predictive models that consist of layers which are made up of neurons in which each neuron is connected to every other neuron in the next layer of the layer sequence. A weight is associated with each connection that determines how much a neuron contributes to the input of the neuron in the next layer, and a bias term is introduced to shift the activations. Activation functions, such as the logistic sigmoid, grant the network a level of non-linearity for additional complexity. A feature vector, in which each feature corresponds to a neuron in the input layer, can be pushed through the layers of the network all the way to the output layer in a process called forward propagation. Neural networks can be trained as classification models that categorize data, or regression models that predict continuous quantities. In this work, we only develop regression models.

Neural networks are trained by updating the weights and biases of the network in a series of steps through gradient descent, in which an appropriate loss function is minimized. The process of calculating the errors of a neural network is called back propagation, in which derivatives are calculated in the backwards direction starting from the output layer, where the loss (or error) is calculated, and propagating backward through each layer all the way back to the input layer, after which the weights are then updated by following the negative direction of the gradient. The gradient descent process finds a local minimum of the loss function when viewed with respect to the weights. This process *fits* the network to the data set in question.

A general neural network consists of an input layer, an output layer, and any number of hidden layers in between. Deep neural networks are characterized by having many more layers and neurons, allowing the network to be more complex. There are many additions to the neural network architecture that can be included for training a model. Activation functions bring non-linearity to a network, and a proper choice of activation function can have an effect on training and performance of a network. Dropout [34] layers prevent over-fitting by dropping out random neurons during each step of training. Weight decay is another technique that regularizes the network to prevent over-fitting by decreasing the magnitude of each weight throughout the training. Batch normalization [12] layers normalizes the outputs of each layer to speed up the training time, improve performance, and allow deep networks to be trained with stability by reducing internal covariant shift. Adaptive learning rates are learning rates that change during the training of the network, such as in a momentum approach to gradient descent that “gains speed as it descends down a valley”, that can find local minimums faster and sometimes have the ability to jump out of “undesirable” local minimums. We use batch normalization, ReLU activation, and the AMSGrad [31] variant of the popular Adam [15] optimization adaptive learning rates in our work. Dropout and weight decay increased the training time dramatically and so were not used in our final models, but may be explored further in the future.

Single-task (ST) neural networks are networks trained on a single data set and have a single output (at least in the regression case). An ST network is trained by the minimization problem:

$$\min_{\mathbf{W},\mathbf{b}}\left[\alpha\cdot\left|\left|\mathbf{W}\right|\right|_2^2+\sum_i^N L(y_i,f(\mathbf{x}_i;\{\mathbf{W},\mathbf{b}\}))\right], \tag{1}$$

where $||\cdot||_2$ denotes the L_2 norm, α represents the regularization constant, f is a function parametrized by the weights, **W**, and biases, **b**, that represents the output of the network, and $\mathbf{x}_i$ and y_i are the feature vector and the label of the i^{th} data point respectively of a data set with N samples.

Single-task networks are, however, limited in their usefulness when being trained on small data sets. Multi-task (MT) networks have been developed to take advantage of large data sets to bolster the training of a network on a smaller data set. The idea is to train the models on multiple tasks simultaneously by sharing weights, thus highlighting relevant features that are important across all the related tasks. The details of their training are as follows:

If we let T be the number of tasks and $\{(\mathbf{x}_i^t,y_i^t)\}_{i=1}^{N_t}$ is the training data for the t^{th} task where N_t is the number of samples of the t^{th} task, $\mathbf{x}_i^t$ is the feature vector of the i^{th} sample of the t^{th} task, and y_i^t is the label for the i^{th} sample of the t^{th} task. The training strategy is to minimize the chosen loss function, L, for all tasks simultaneously. We define the loss for task t, L_t, below.

$$L_t=\sum_{i=1}^{N_t}L\left(y_i^t,f^t\left(f^s\left(\mathbf{x}_i^t;\{\mathbf{W}^s,\mathbf{b}^s\}\right);\{\mathbf{W}^t,\mathbf{b}^t\}\right)\right), \tag{2}$$

where f^t is the part of the network that predicts the labels for the t^{th} task parametrized by weights, $\mathbf{W}^t$, and bias, $\mathbf{b}^t$; f^s is the part of the network that has shared weights, $\mathbf{W}^s$, and bias, $\mathbf{b}^s$. To train all models simultaneously, we provide a single loss function to minimize so our problem is then:

$$\text{argmin}\left[\beta\cdot\left|\left|\mathbf{W}^s\right|\right|_2^2+\frac{1}{T}\sum_{t=1}^{T}\left(L_t+\alpha\cdot||\mathbf{W}^t||_2^2\right)\right], \tag{3}$$

where $||\cdot||_2$ denotes the L_2 norm and α and β represent the regularization constants that set the magnitudes of the weight decay during training. In practice, if weight decay is used, we generally set $\alpha = \beta$.

2.2. Overview of auto-parametrized weighted element-specific graph neural networks (AweGNNs)

We now wish to describe the concept of AweGNNs applied to molecular data sets. As in most machine learning endeavors, we seek to frame the task at hand as a minimization problem. To detail our supervised learning algorithm, we first start with a biomolecular data set, χ, where χ_i will represent the i^{th} element of the training data set. We want a function $\mathbf{F}(\chi_i;\{\boldsymbol{\eta},\boldsymbol{\kappa}\})$ that encodes the geometric and chemical information into an abstract representation parametrized by a set of parameters, $\{\boldsymbol{\eta},\boldsymbol{\kappa}\}$. Then, the minimization problem becomes:

$$\min_{\boldsymbol{\eta},\boldsymbol{\kappa},\mathbf{W},\mathbf{b}}\alpha\cdot\left|\left|\mathbf{W}\right|\right|_2^2+\sum_{i\in I}L(\mathbf{y}_i,f(\mathbf{F}(\chi_i;\{\boldsymbol{\eta},\boldsymbol{\kappa}\});\{\mathbf{W},\mathbf{b}\})), \tag{4}$$

where L is the chosen scalar loss function to be minimized, α is the regularization constant that determines the magnitude of the weight decay, and $\mathbf{y}_i$ is the label of the i^{th} data point in our biomolecular data set. The function, f, is the function parametrized by the weights, **W**, and biases, **b**, that represent the output of the neural network. Notice the difference with the formulations in the previous section. The output of the function, **F**, on the data χ_i acts as the feature vector, $\mathbf{x}_i$, given previously. The parameters, {**η**, **κ**}, that parametrize **F** are included in the minimization process. This simultaneous update of the parameters of the representation function is the novel idea driving this work.

The AweGNNs are trained for the regression task of predicting the toxicity endpoints and the solvation free energy of various small molecules, with the standard choice of mean squared error (MSE) as our loss function, L. The regularization constant, α, as noted before, is omitted in our work for both the single and multi-task models due to a lack of noticeable increase in performance, but with a moderate increase in training time. **F** will be based on element-specific calculations that are controlled by pairs of tunable kernel parameters that capture different geometric features of the molecules based on the values of those kernels chosen. Each element-specific group will be assigned a pair of these parameters and will be updated throughout the training by back propagation. The representations generated by **F** after the training of the AweGNN are called network-enabled automatic representations (NEARs), and will be used later for training other models to demonstrate their performance. The procedure is summarized in Fig. 1 below.

More specifically, we wish to understand how to back propagate through **F**. This will require us to, at each batch or epoch, back propagate throughout the entire network, then use that information to back propagate through **F** all the way to the kernel parameters that **F** is dependent on. In reality, **F**, should be a composition of a function that outputs vector representations of biomolecules along with the normalization of the output of that function. Thus, we can represent **F** as $\mathbf{F} = N\circ R$, where N is the normalization function, and R is the raw representation function. The normalization function is important because it avoids instability in the network. Now when considering back propagation through the function, **F**, we have to keep in mind the chain rule, $\frac{\partial\mathbf{F}}{\partial x} = \frac{\partial N}{\partial R}\frac{\partial R}{\partial x}$, as we calculate our partial derivatives.

If we want to be able to calculate these derivatives, then we must make sure that they are differentiable functions. Some normalization functions, however, will not be differentiable everywhere, but will usually still be differentiable *almost* everywhere and continuous *almost* everywhere, so they will be practical functions to use. Representation functions themselves can be very discontinuous and non-differentiable depending on the method of calculating features, so we must be especially careful in selecting the function, R. We would also like both the representation function and normalization function to be easy to calculate since we must calculate the normalized features at each epoch of training. We will later describe in detail the geometric graph representation function, R, that we used for our AweGNN that is continuous and differentiable everywhere, and easy to calculate.

We will be testing both single-task (ST) and multi-task (MT) AweGNN models that follow much the same structure, again as outlined in Fig. 1. The ST model takes batches and from a data set, generates a representation, then normalizes it with batch normalization before feeding it into the artificial neural network portion of the AweGNN and then updating the parameters as above. The MT-AweGNN consists of a common set of parameters that are used for all 4 toxicity data sets, shared weights for the hidden layers of the artificial neural network portion, but separate weights for the 4 output layers. The normalization is done also in batches, by placing a number of data points into the batch from each data set proportional to the size of that data set with respect to all the data sets combined. For example, if the batch size is 100, and there are data sets with size 200, 400, 600, and 800, then each batch would have 10 samples from the first data set, 20 from the second, 30 from the third, and 40 from the last data set. This MT structure is illustrated below in Fig. 2.

2.3. Normalization function

For the normalization function, we choose the standard normalization function, N_σ, which centers each feature about the mean and scales by the standard deviation of the training samples. More precisely, if x is the feature, μ the mean, and σ the standard deviation, then $N_\sigma(x) = \frac{x-\mu}{\sigma}$.

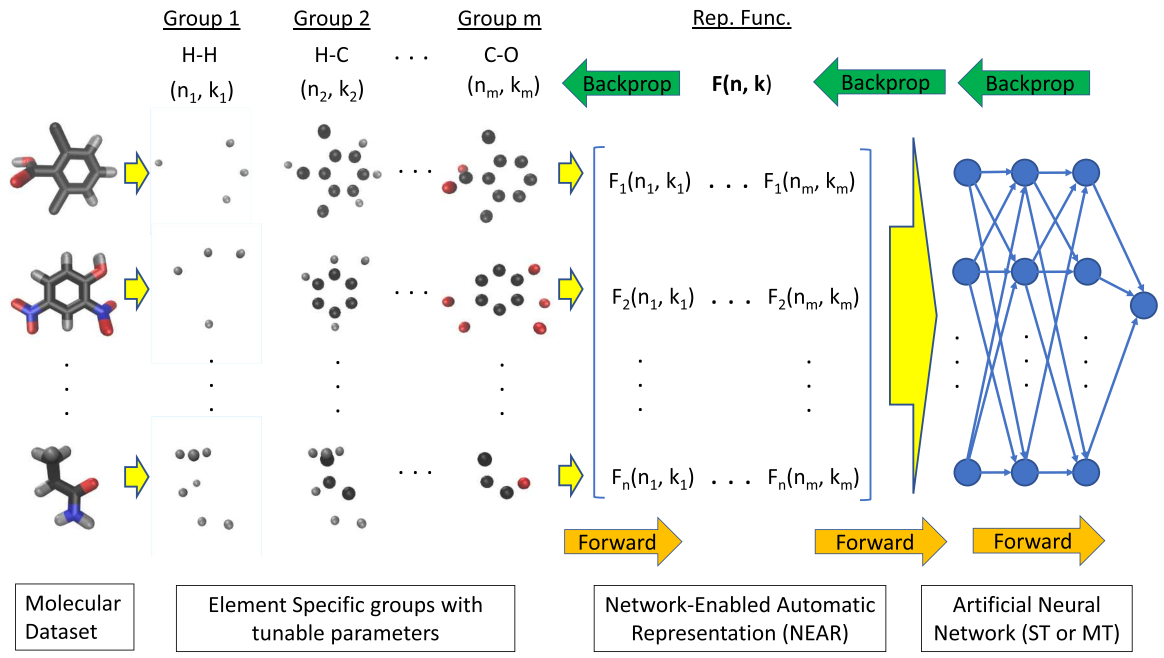

Group 1
H-H
(n1, k1)
Group 2
H-C
(n2, k2)
Group m
C-O
(nm, km)
Rep. Func.
F(n, k)
Backprop
Backprop
Backprop
F1(n1, k1) . . . F1(nm, km)
F2(n1, k1) . . . F2(nm, km)
Fn(n1, k1) . . . Fn(nm, km)
Forward
Forward
Forward
Molecular Dataset
Element Specific groups with tunable parameters
Network-Enabled Automatic Representation (NEAR)
Artificial Neural Network (ST or MT)

Fig. 1. Pictorial visualization of the AweGNN training process. The first stage is splitting up the molecules in the data set into the element-specific groups, with initialized η and κ kernel parameters for each group. Then, a molecular representation is generated based on the kernel parameters, given by **F** in the diagram. This representation is fed into the neural network, then the kernel parameters are updated by back propagation through the neural network and through the representation function, **F**.

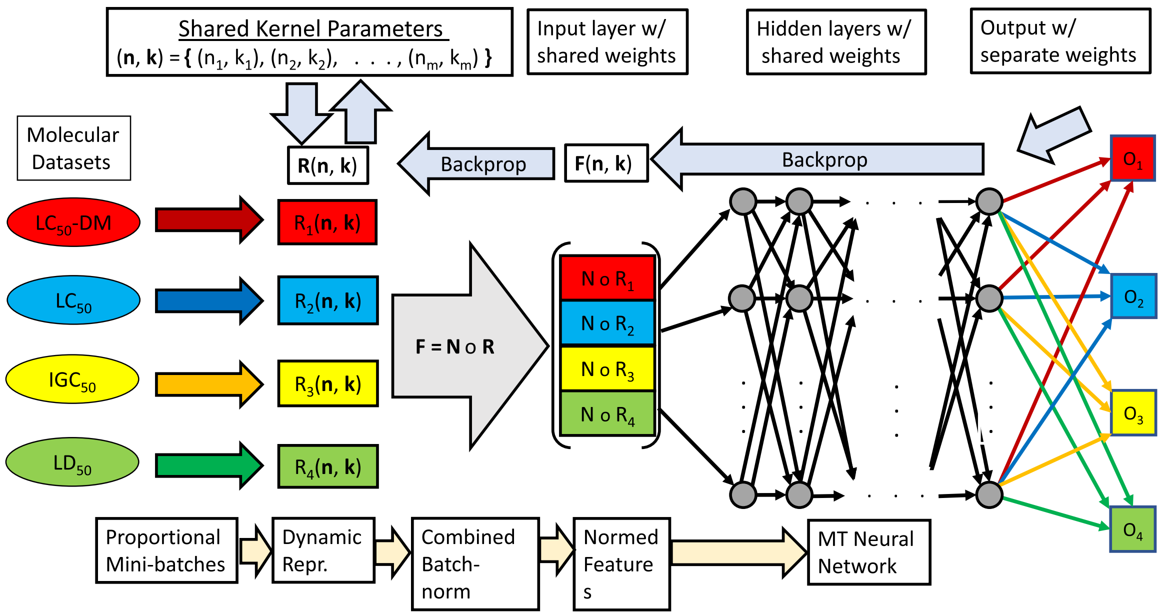


Fig. 2. Diagram depicting the structure and automation of the MT-AweGNN. The model is trained by collecting samples from each data set proportionally and then generating their representations according to shared parameters. The representations are normalized together, then pushed through the artificial neural network and then to their corresponding outputs. Back propagation goes through the feature matrix, normalization function, and finally the representation function to update the kernel parameters.

This function is continuous and differentiable everywhere except when the standard deviation is zero. By a slight modification, we can add an error constant for numerical stability and to make N_σ continuous and differentiable everywhere, where $N_\sigma(x) = \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}}$ with ε being a very small number such as 10^{-5}. When training in batches, we normalize each batch individually with separate batch mean, $\mu_{\mathscr{B}}$, and batch standard deviation, $\sigma_{\mathscr{B}}$. These statistics are tracked and a batch momentum value of 0.1 is used to record accumulated batch statistics to obtain a more stable reading of the distribution of the features in the whole data set, as detailed in Ioffe and Szegedy [12]. During evaluation time, the recorded statistics are used to transform the output of the representation function, *R*.

The normalization procedure is similar to batch normalization in neural networks, except that the affine transformation is omitted, and the expression for $\frac{\partial N}{\partial R}$ can be found in the paper written by Ioffe and Szegedy [12], along with the rest of the details for batch normalization. Although the above procedure omits the affine transformations, we chose to apply the affine transformations in our work because the models performed well with them. In practical terms, the PyTorch [29] library is utilized for applying this batch normalization. A BatchNorm1d layer is placed in front of the molecular representation layer with parameters: eps = 1e-05, momentum = 0.1, affine = True, and track_-running_stats = True, which are the default settings in PyTorch [29].

2.4. Biomolecular geometric graph representations

We seek a general formulation of a geometric graph representation of biomolecular structures from which we can extract features to create our sought after representation function, *R*. We use this method to generate descriptors for the toxicity analysis of small molecules, but the idea can extend beyond this concept quite naturally to provide useful representations. In our work, we want to use a weighted and vertex-labeled graph in which we can generate descriptors from special subgraphs of the graph representation of each molecular structure. We begin by looking for a set, $\mathscr{T}$, of element types that are suitable for our analysis. Generally the choice of element types will be chosen by looking at the commonly occurring element types found in a given data set, or we might choose to omit certain element types to avoid elements that have uncertain positioning or negligible influence in molecular interactions. Usually, we will have at least C, N, O, S $\in \mathscr{T}$, but in most cases we will also include H, P, Cl, and Br. We define

$$\mathscr{V} = \left\{ \left(\mathbf{r}_j, \alpha_j\right) \in \mathbb{R}^3 \times \mathscr{T} \mid \mathbf{r}_j;\ \alpha_j \in \mathscr{T};\ j = 1, 2, \ldots, N \right\}$$

to be the vertex set, in which *N* denotes the number of atoms in the molecule that are of an element type that is a member of $\mathscr{T}$. This vertex set has a labeling that describes the atom coordinates and the element type of the atom at each vertex. We will use these labels to determine element specific subgraphs from which to extract features.

Our edge set, $\mathscr{E}$, is described by a choice of parameters, $\{\eta_{kk'}\}$ and $\{\kappa_{kk'}\}$, and a choice of a type of kernel function, $\Phi : \mathbb{R} \to \mathbb{R}$, which is parametrized by them

$$\mathscr{E} = \left\{ \Phi\left(\left|\left|\mathbf{r}_i - \mathbf{r}_j\right|\right|; \eta_{kk'}, \kappa_{kk'}\right) \mid \alpha_i = k \in \mathscr{T}, \alpha_j = k' \in \mathscr{T}\ ;\ i, j = 1, 2, \ldots, N\ ; \\ \left|\left|\mathbf{r}_i - \mathbf{r}_j\right|\right| > v_i + v_j + \sigma \right\},$$

where $||\cdot||$ denotes the Euclidean distance, v_i and v_j denote the Van der Waals radius of the i^{th} and j^{th} atoms respectively, $\mathbf{r}_i$ and $\mathbf{r}_j$ represent the i^{th} and j^{th} atom coordinates, and σ is the average of the standard deviations of v_i and v_j in the data set. Notice, the distance constraint eliminates covalent interactions from being used in our calculations. We remove the covalent interactions because many biomolecular properties are known to be determined by non-covalent interactions, As noted in Nguyen et al. [27], many biomolecular properties are known to be determined by non-covalent interactions, so we remove the covalent interactions because we do not want their contribution to overwhelm or otherwise interfere with the contributions of the non-covalent interactions. The parameters, $\eta_{kk'}$ and $\kappa_{kk'}$, are values tied to the element specific pair, kk', which can be tied to the properties of the element types in question. The types of kernels that determine the weights of our edges are always chosen to be decreasing functions that have the properties:

$$\Phi(x) \to 1 \quad \text{as} \quad x \to 0 \tag{5}$$

and

$$\Phi(x) \to 0 \quad \text{as} \quad x \to \infty. \tag{6}$$

This characterization gives the property that atoms that are closest will contribute the most, and atoms that are too far away, will contribute almost nothing. Most radial basis functions can be used, but the most popular choice of kernel types are the generalized exponential and the generalized Lorentz functions, given by

$$\Phi^E(x;\ \eta, \kappa) = e^{-\left(\frac{x}{\eta}\right)^\kappa} \tag{7}$$

and

$$\Phi^L(x;\ \eta, \kappa) = \frac{1}{1 + \left(\frac{x}{\eta}\right)^\kappa}, \tag{8}$$

respectively. These kernel types have been used to create successful models for protein-ligand binding prediction, predicting toxicity endpoints, and many other applications [25,27,28]. They act as low-pass filters that capture the most important element interactions, where the η value determines the cutoff point, and the κ value determines the sharpness of the cutoff. In summary, we have procured a vetex-labeled, weighted graph, $G(\mathscr{V}, \mathscr{E})$, in which we can extrapolate features from special element specific subgraphs.

Let $G_{kk'}$ be the subgraph of *G* whose vertex set, $\mathscr{V}_{kk'}$, contains the vertices that are labeled element type *k* or k'; and the edge set, $\mathscr{E}_{kk'}$, contain only the edges connecting a vertex labeled with element *k* to a vertex labeled with element type k'. This is called the element specific subgraph generated by the element pair kk'. For a given element specific subgraph, $G_{kk'}$, we define the following descriptor associated with that subgraph:

$$\mu_{G_{kk'}, \Phi} = \sum_{(i,j) \in \mathscr{E}_{kk'}} w_j \cdot \Phi\left(||\mathbf{r}_i - \mathbf{r}_j||;\ \eta_{kk'}, \kappa_{kk'}\right) \tag{9}$$

where the w_j term is usually a constant 1 or a value associated with the j^{th} atom, such as a partial charge. The $\eta_{kk'}$ and $\kappa_{kk'}$ values represent the respective η and κ values for the kk' group, while $\mathbf{r}_i$ and $\mathbf{r}_j$ represent the atom coordinates as before. We get a sum over the weights of the subgraph in which additional atom specific weights can be applied to enhance the meaning of the feature. This abstractly defines the total strength of the non-covalent interactions between the atoms of element

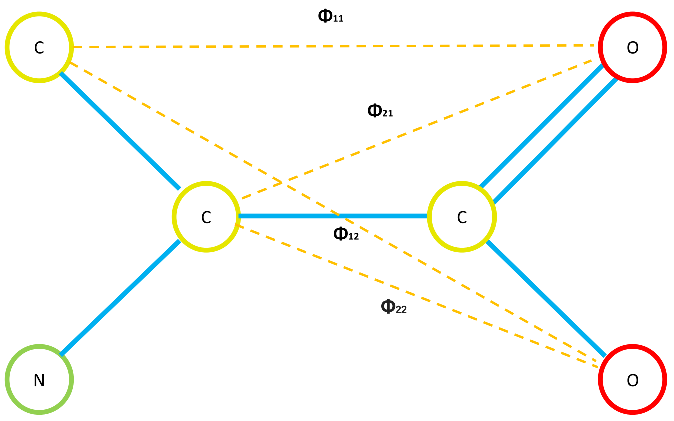


Fig. 3. This shows an element specific subgraph corresponding to the element types C and O of aminopropanoic acid, $C_3H_6NO_2$ (with the hydrogens omitted). The dashed orange edges represent the edges of the subgraph that are weighted by the chosen kernel function, Φ, while the solid blue lines represent covalent bonds. Notice the 3rd carbon atom does not connect to the oxygen atoms since it is covalently bonded to both of them.

type k and the atoms of element type k'. This is illustrated in Fig. 3 below.

With these descriptors detailed above, we can describe a representation function by obtaining one or more features from each element specific subgraph. We can extract multiple features per subgraph by considering different statistics of our features such as the mean of the sum above, the standard deviation of the terms in the sum, choosing various values of w_j above according to chemical or physical properties of specific atoms, etc. In summary, the features that we generate using this approach measures the aggregate interaction strength between groups of atoms of specific element types, which we call element-specific groups. This gives us a representation of the internal structure of a molecule that fits into the QSAR paradigm, which promotes the idea that similar molecular structures have similar physical properties.

The representation function used in our work for the toxicity data sets consisting of small molecules considers the element type set $\mathscr{T} = \{H,C,N,O,S,P,F,Cl,Br,I\}$. For each element specific pair, we calculate 4 descriptors: The first is calculated with each $w_j = 1$, the next with the w_j's set to be equal to the partial charge of the j^{th} atom, and the remaining 2 descriptors are generated by dividing the first 2 descriptors by the number of edges in the element specific group, giving us a measure of the mean interaction of the elements. Since we have 10 different element types, we can form 100 element specific groups; and since we have 4 descriptors per element specific group, we have 400 descriptors for every set of parameters and choice of kernel. In our work, we only use the Lorentz functions, $\Phi^L_{\eta,\kappa}$, for our feature generation and results.

2.5. Parameter adjustment and initialization

When considering our molecular representation function in the context of the parameter automation task, we can make a few choices in terms of the parameters that we will be adjusting at each epoch or batch. Below are 3 possible avenues to consider:

η-adjustable Here, we set the η values for each element specific pair randomly within some range. We note that these assignments that are made for each element specific pair are common for all of the biomolecular structures in your data set. We can combine several sets of features to make multi-scale models that would have multiple initializations of the η values, but we will be assuming a 1-scale model at present. If we set the η values this way, these will be the parameters that we will be updating in our gradient descent.

τ-adjustable For an element specific pair, kk', we set the characteristic value to be $\eta_{kk'} = \tau(\nu_k + \nu_{k'})$, where ν_k is the Van der Waals radius of element type, k, and $\nu_{k'}$ is the Van der Waals radius of element type, k'. Similar to the previous situation, the τ value assigned determines the element specific η values for the entire data set. Again, if we consider multi-scale models, then we may choose different values of τ for each set. In the τ-adjustable case, we will be seeking to update the τ value at each step of the training.

κ-adjustable In the previous two situations, we were assuming a fixed κ value, but we can also update this parameter in conjunction with the others. There are two ways to introduce this technique. The first way is to introduce one κ and update that with respects to all of the element specific groups. The second way is to introduce a κ value for all the element specific groups and update those separately. Multi-scale models are handled in a similar way as above.

Aside from choosing which parameters to update throughout the training of the AweGNNs, we also must consider how we want to initialize each parameter before training. We notice that τ, κ, and η all have to take values greater than zero to maintain continuity and/or satisfy the conditions of a radial basis function. Also, we want to make certain that the values are not too large. A large η value will capture all of the atoms in a molecule and the kernel value will be effectively a constant 1 at all measured distances. A large κ value will make the kernel function approach the form of an ideal low-pass filter, where it is either a constant 0 or constant 1. In both cases, the magnitude of the derivatives will be extremely small and the model will not be able to update effectively.

In our experimentation, we use 100 η-adjustable and 100 κ-adjustable parameters per scale (1 η and 1 κ value per element specific group) giving us 200 total adjustable variables per scale for our molecular representation function. The η-initialization for each scale is chosen by a single random tau value from a uniform distribution with a range of 0.5–1.5, in which all the η values are set according to the Van der Waals radii as described above. The κ-initialization for each scale is done by chosing a value of κ for each element-specific group from a uniform distribution with a range of 5–8.

The adjustment of parameters throughout the training of our AweGNN will be like an evolving kernel function at each element-specific group that attempts to create an optimal filter that provides the best representation for the network. The η values determine the center of the cutoff region and the κ values control the sharpness of the cutoff. In Fig. 4, we see the evolution of the kernel function of a MT-AweGNN model, where the kernel functions are graphed at various choices of the number of epochs of training, so that the transformation of the kernel function can be clearly seen.

2.6. Derivatives of the representation function

Our main goal is to update the parameters of our feature representation during the training of our neural network. We must first show how to calculate the derivatives of the representation functions with respect to parameters κ, η, and τ values. We will then use these derivatives to show how to complete the back propagation through all the way to the parameters. We begin with a representation function, $R:\mathbb{R}^n \rightarrow \mathbb{R}^{d\times m}$, and focus on a given element-specific subgraph; where this could be a single or multi-scale representation function.

Suppose that $\boldsymbol{\nu}$ is the set of update-able parameters for R. Then we can represent R by:

$$R(\boldsymbol{\nu})_{i,j} = \sum_{(a,b)\in G} q_b \cdot \Phi(||\mathbf{r}_a - \mathbf{r}_b||;\boldsymbol{\nu}_G), \tag{10}$$

Where G is the element specific subgraph which is used in the calculation of the j^{th} descriptor of the i^{th} sample, q_b represents a value that may be associated with the atom labeled b, $\mathbf{r}_a$ and $\mathbf{r}_b$ represent the coordinates of the atoms labeled a and b respectively, and $\boldsymbol{\nu}_G$ is the subset of $\boldsymbol{\nu}$ that corresponds to the calculations that are associated with G.

Now, since differential operators are linear and our features are calculated by a summation of differentiable functions, we need only know how to calculate the derivative of each term, i.e., for any variable, x, we have:

$$\left(\frac{\partial R(\boldsymbol{\nu})}{\partial x}\right)_{i,j} = \sum_{(a,b)\in G} q_b \cdot \frac{\partial}{\partial x}\Phi(||\mathbf{r}_a - \mathbf{r}_b||;\boldsymbol{\nu}_G). \tag{11}$$

Thus, we want to find out the derivatives of the functions, $\Phi^E_{\eta,\kappa}$ and $\Phi^L_{\eta,\kappa}$, with respect to parameters η, τ, and κ. We start by finding the derivative with respect to η.

The derivative of the generalized exponential function is:

$$\frac{\partial \Phi^E_{\eta,\kappa}(r)}{\partial \eta} = \left(\frac{\kappa}{\eta}\right)\left(\frac{r}{\eta}\right)^{\kappa} e^{-(r/\eta)^{\kappa}}, \tag{12}$$

and for the Lorentz generalized function, we have the derivative:

$$\frac{\partial \Phi^L_{\eta,\kappa}(r)}{\partial \eta} = \frac{\left(\frac{\kappa}{\eta}\right)\left(\frac{r}{\eta}\right)^{\kappa}}{\left(1+\left(\frac{r}{\eta}\right)^{\kappa}\right)^2} \tag{13}$$

Also, we calculate the derivatives with respect to κ parameters

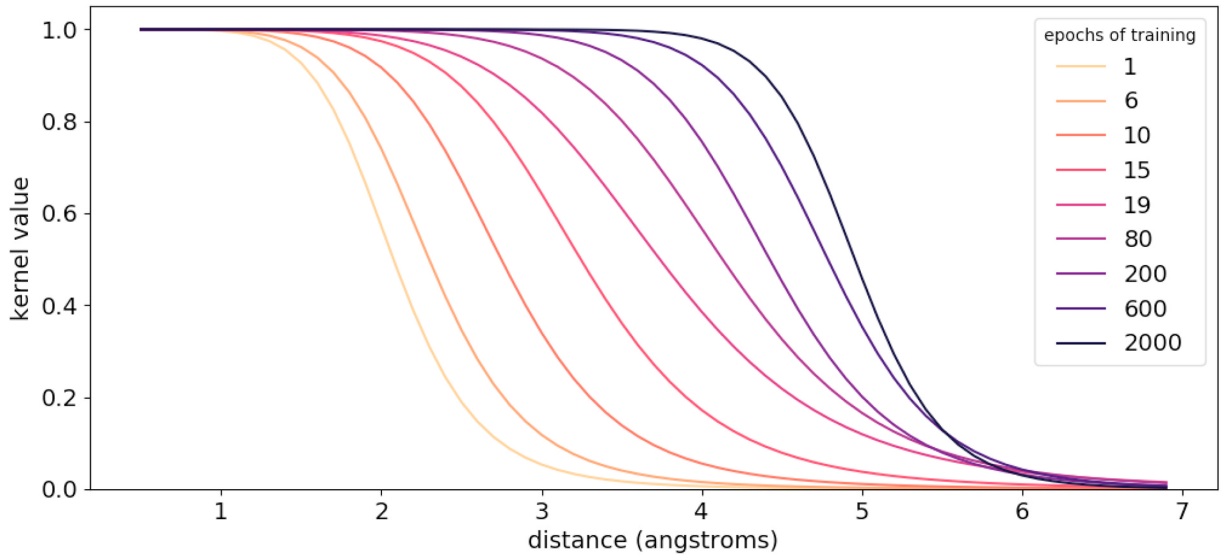


Fig. 4. Evolution of the kernel function of the C–N group for a MT-AweGNN. The picture below shows a series of kernel functions that were used to evaluate the 4 features corresponding to the C–N element-specific group at different points in the training of an MT-AweGNN. We choose specific snapshots during training that show a smooth change in the kernel function as the η-κ pair is updated.

starting with the generalized exponential function:

$$\frac{\partial\Phi^{E}_{\eta,\kappa}(r)}{\partial\kappa}=\left(\frac{r}{\eta}\right)^{\kappa}\ln\left(\frac{\eta}{r}\right)e^{-(r/\eta)^{\kappa}}, \tag{14}$$

and for the Lorentz generalized function, we have the derivative:

$$\frac{\partial\Phi^{L}_{\eta,\kappa}(r)}{\partial\kappa}=\frac{\left(\frac{r}{\eta}\right)^{\kappa}\ln\left(\frac{\eta}{r}\right)}{\left(1+\left(\frac{r}{\eta}\right)^{\kappa}\right)^{2}}. \tag{15}$$

Now, for the derivative with respect to the τ parameter, we can just apply the chain rule with $\eta=\tau(\nu_1+\nu_2)$ being a function of τ. Note that $\frac{\partial\eta}{\partial\tau}=(\nu_1+\nu_2)=\frac{\tau(\nu_1+\nu_2)}{\tau}=\frac{\eta}{\tau}$. Then, for the exponential derivative, we have:

$$\frac{\partial\Phi^{E}_{\eta,\kappa}}{\partial\tau}=\frac{\partial\eta}{\partial\tau}\cdot\frac{\partial\Phi^{E}_{\eta,\kappa}}{\partial\eta}=\left(\frac{\eta}{\tau}\right)\left(\frac{\kappa}{\eta}\right)\left(\frac{r}{\eta}\right)^{\kappa}e^{-(r/\eta)^{\kappa}}=\left(\frac{\kappa}{\tau}\right)\left(\frac{r}{\eta}\right)^{\kappa}e^{-(r/\eta)^{\kappa}}, \tag{16}$$

and for the lorentz function, we have:

$$\frac{\partial\Phi^{L}_{\eta,\kappa}}{\partial\tau}=\frac{\partial\eta}{\partial\tau}\cdot\frac{\partial\Phi^{L}_{\eta,\kappa}}{\partial\eta}=\left(\frac{\eta}{\tau}\right)\frac{\left(\frac{\kappa}{\eta}\right)\left(\frac{r}{\eta}\right)^{\kappa}}{\left(1+\left(\frac{r}{\eta}\right)^{\kappa}\right)^{2}}=\frac{\left(\frac{\kappa}{\tau}\right)\left(\frac{r}{\eta}\right)^{\kappa}}{\left(1+\left(\frac{r}{\eta}\right)^{\kappa}\right)^{2}} \tag{17}$$

Thus, we now know the derivatives $\frac{\partial R}{\partial\eta}$, $\frac{\partial R}{\partial\tau}$, and $\frac{\partial R}{\partial\kappa}$; and we can use them to calculate the gradient for our gradient descent when training our neural network.

2.7. How to update the parameters

Now that we have the derivatives of the representation function, we will be able to calculate the derivatives of the loss function that we have chosen, L, with respect to our parameters, η, τ, and κ. We must remember that in our process of feature calculation, we have decided to normalize the features. We chose to use the standard normalization, N_σ, and we denoted the normalized representation function to be $F=N\circ R$, with R being the un-normalized representation function. By the chain rule, we obtain $\frac{\partial F}{\partial x}=\frac{\partial N}{\partial R}\frac{\partial R}{\partial x}$, where $x=\eta$, κ, or τ. We have also discussed what $\frac{\partial N}{\partial R}$ looks like, so in fact we should now have a full description of the derivatives of the normalized representation function with respect to our parameters.

Now, to get our final update rule, we have to extend the back propagation of our neural network to the features, obtaining the derivatives, $\frac{\partial L}{\partial F}$. Then our derivatives that we calculated earlier, $\frac{\partial F}{\partial x}$, will be used in the chain rule again to get: $\frac{\partial L}{\partial x}=\frac{\partial L}{\partial F}\frac{\partial F}{\partial x}$, where $x=\eta$, κ, or τ. This gives us the update rule for our parameters. In more detail, the update rule actually looks like $\frac{\partial L}{\partial x}=\sum_{i,j}\frac{\partial L}{\partial F_{ij}}\frac{\partial F_{ij}}{\partial x}$. Remember that the τ, κ, and η parameters are all constrained to be greater than zero. As a precaution we propose that after every epoch, the parameters be clipped so that they are constrained to values that are 0.01 or higher to avoid forbidden values.

2.8. Multi-scale models

Multi-scale models are models that are trained on features generated by multiple sets of parameters. More specifically, for 2 representation functions, $R_1:\mathbb{R}^n\rightarrow\mathbb{R}^{d\times m}$ and $R_2:\mathbb{R}^n\rightarrow\mathbb{R}^{d\times m}$, we can get the 2-scale representation function, $[R_1,R_2]:\mathbb{R}^n\rightarrow\mathbb{R}^{d\times 2m}$, by concatenating the outputs of R_1 and R_2 so that the features of each data point match up.

We can extend this idea of a 2-scale model to any scale. Let $R_1:\mathbb{R}^n\rightarrow\mathbb{R}^{d\times m}$, …, $R_k:\mathbb{R}^n\rightarrow\mathbb{R}^{d\times m}$ be representation functions with the same parameter types. Then, combining the functions together one at a time as above, then we get a new multi-scale representation function, $[R_1,R_2,\ldots,R_k]:\mathbb{R}^n\rightarrow\mathbb{R}^{d\times(k\cdot m)}$. These multi-scale methods have been shown to be very effective in improving the performance of single-scale models [27,28].

2.9. Model architectures and hyper-parameters

We wish to now describe the specific network architecture of the AweGNN model. For the artificial neural network (ANN) portion of the AweGNN, we choose a very simple 4-layer network with 400 neurons in the first two hidden layers and 20 neurons in the last two hidden layers. The ANN architecture and parameters were chosen through a quick parameter search of the multi-task network where the models were tested on a randomized validation set of 10% of the training data of each toxicity data set to obtain good average predictions across all 4 data sets and was modified to ensure relative convergence for the kernel parameters. The convergence assures that we have stable choices for the parameters of the kernel function, which we will use for later analysis.

Dropout and weight decay are omitted since they did not notably increase performance, but increased the training time significantly.

The AMSGrad [31] variant of the Adam optimizer is used because it has been shown to improve the convergence of the kernel parameters while still achieving large gradients that allow the model to explore a larger parameter space, and allows a more interesting analysis of the parameter trajectories. A large learning rate of 0.1 is also applied with a learning rate decay of 0.999 per epoch of training, for the total of 2000 epochs, leaving the final learning rate at a value of approximately 0.01352 at the end of the training. The decay gives the network and kernel parameters an opportunity to settle down in a local minimum, while the large learning rate gives the models a chance to explore more parameter choices while also regularizing the network. The network was further improved by applying batch normalization, which is known to speed up the training of a network and provides an additional source of regularization that reduces the need for dropout [12]. Finally, the batch size used was 100 for the ST models, while the MT models used a total of 40 batches per epoch, with sizes as even as possible and number of samples from each data set proportional to their respective size.

For all of the network models, we use the PyTorch [29] API in conjunction with cython [4] and Numba [16] to speed up the calculation of the geometric graph descriptors. All hidden layers are actually composed of 3 different layers: a linear unit, followed by a batch normalization layer with affine transformations active and momentum set to 0.1 (default PyTorch setting), then a ReLU activation. The input layer is constructed with 2 layers: a Geometric Graph Representation (GGR) layer followed by a batch normalization layer again with affine transformations and the same momentum setting. The GGR layer is the representation function, R, and the batch normalization layer is the N_σ normalization function, as described in detailed previously to generate our normalized representation function, $F = N_\sigma \circ R$. Finally, the output layer is simply a linear activation into either 1 or 4 outputs, depending on if we are using a multi-task or single-task model. The details of the full AweGNN architecture are illustrated in Fig. 5 below.

As discussed earlier, the network-enabled automatic representation (NEAR) generated from training an AweGNN can be used as inputs for ensemble models. By examining the performance of these models, we can see if the representations generated by AweGNNs can produce meaningful features for a broad range of machine learning models. We use two ensemble models, random forest regressor (RF) and gradient boosting regressor (GBT), from the scikit-learn v0.23.2 package [30]. A search for reasonable parameter choices was made on the same validation sets above. For the RF models, we use the parameters: n_estimators = 8 000, max_depth = 27, min_samples_split = 3, and max_features = sqrt, with any remaining parameters considered to be set at the default setting. For the GBT models, we use the same parameters as the RF models above along with the additional parameters: loss = 'ls', learning_rate = 0.1, and subsample = 0.2, with any remaining parameters set to the default choice.

3. Results

In this section, we give a description of the different quantitative toxicity data sets, and then we compare the results of our best models to the models generated by other published methods [21,22,27,40]. We note that in the work by Nguyen et al. [27], we only report the results from one consensus model trained and tested on the Tetrahymena pyriformis IGC_{50} data set that performed the best, which is labeled as *Nguyen-best*. For the work done by Wu et al. [40], we only report the results for the models that utilized all the descriptors detailed in that paper. The results of the random forest models are referred to as Wu-RF, the gradient boosting models are referred to as Wu-GBT, the ST models are Wu-ST, the MT models are Wu-MT, and the consensus of Wu-GBT and Wu-MT is Wu-consensus. Except in the case of the *Daphnia Magna* LC_{50}, the best scoring Wu models were those that relied on all descriptors, but any such important detail will be mentioned when relevant.

We also introduce a solvation data set, Model III, as in the paper by Nguyen et al. [27]. The performance of our AweGNN models, when trained on this data set, is compared to those of differential geometry based models developed earlier [27], and other state-of-the-art methods [36,38]. The models in this particular section are named as they are in the previously referenced paper [27], where the names refer to various choices of kernel options/kernel parameters used in the calculation of differential geometric descriptors.

As for our own models, we report them in many ways and in many combinations. We train multiple models for each data set and each model type, then we combine those in that instance by averaging their predictions to get one consensus prediction. This is done to reduce variance from random weight initializations and to improve our final results. We also combine the predictions of different model types in consensus to see if those will possibly yield better results, as we see in Wu [40]. We note that models were trained in sets of 42 instead of sets of 40 or 50 models due to the constraints of our computing architecture. If there are computational concerns, we could also significantly reduce the number of models trained (perhaps to 5 or 10 models), and still obtain

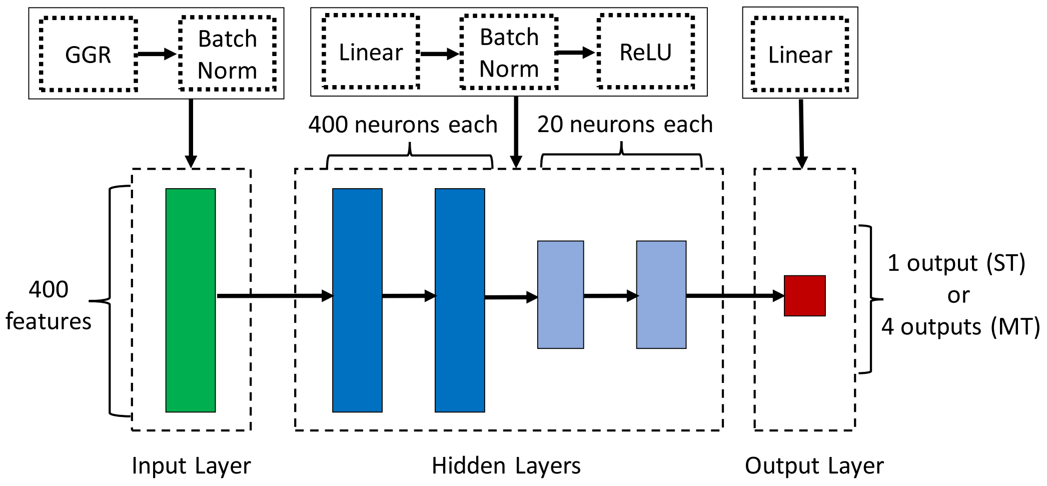


Fig. 5. Internal architecture of our Auto-parametrized weight element-specific Graph Neural Network (AweGNN) model for the toxicity data sets. The input layer consists of our novel Geometric Graph Representation (GGR) layer and a batch norm with no affine transformation. Hidden layers include a linear transformation followed by regular batch normalization and ReLU activation, while the output layer has the standard linear activation.

similar results. For each data set, we train 42 ST and MT AweGNNs with randomized initial weights and obtain a consensus prediction for both types, labeled ST-network and MT-network respectively. Then, we train one ensemble model for each of the NEARs generated by training the networks. Thus, for random forest, we train 42 models based on the NEARs resulting from the ST models and 42 more on the NEARs resulting from the MT models, in which we get the corresponding consensus predictions, ST-RF and MT-RF. Similarly, we get predictions from the GBT models, ST-GBT and MT-GBT. Finally, we obtain various consensus predictions by combining the previous predictions mentioned above. First, we average the prediction ST-network and MT-network to get the consensus, network-consensus. Next, we combine ST-network and ST-GBT to get ST-consensus. Similarly, we obtain MT-consensus by combining MT-network and MT-GBT. Finally, we get the prediction, final-consensus by taking the consensus between ST-consensus and MT-consensus. These predictions and their performances on the test sets are recorded in the tables in section 3.2, and a summary of this entire process is depicted in Fig. 6.

3.1. Evaluation metrics

A protocol was proposed by Golbraikh et al. [11] to determine if a QSAR model has true predictive power, as summarized in the following four points:

1. $q^2 > 0.5$
2. $R^2 > 0.6$
3. $\frac{R^2 - R_0^2}{R^2} < 0.1$
4. $0.85 \leq k \leq 1.15$

Where q^2 is the square of the leave one out correlation coefficient for the training set, R^2 is the square of the Pearson correlation coefficient between the experimental and predicted toxicity endpoints of the test set, R_0^2 is the square of the correlation coefficient between the experimental and predicted toxicity for the test set in which the intercept is set to zero for a linear regression performed on the predicted and experimental labels given by X and Y respectively so that the regression is given by $Y = kX$ (from which the k value is extrapolated).

For each numerical experiment involving toxicity, we record the metrics above, although we omit the q^2 coefficient due to the computational cost. We also record the root-mean-squared error (RMSE) and the mean-absolute error (MAE) since these are standard metrics for evaluating the performance of a model. The final metric to report is coverage, or the fraction of chemicals predicted, which is important because a lower coverage can unfairly improve prediction accuracy. For the solvation data sets, we only report the MAE, RMSE, and the R^2 score as in Nguyen et al. [27].

3.2. Toxicity data sets

This paper analyzes 4 different quantitative toxicity data sets [21]. These are the 96h fathead minnow LC_{50} data set, the 48h Daphnia magna LC_{50}-DM data set, the 40h Tetraphymena pyriformis IGC_{50} data set, and the oral rat LD_{50} data set. The LC_{50} (LC_{50}-DM) set report the concentration in milligrams per liter of test chemicals placed in water that it takes to cause 50% of fathead minnows (Daphnia magna) to die after 96(48) hours. These were downloaded from the ECOTOX aquatic toxicity database via the web site e http://cfpub.epa.gov/ecotox/and were preprocessed using filter criterion including media type, test location, etc. The IGC_{50} data set measures the 50% growth inhibitory concentration of the Tetrahymena pyriformis organism after 40 h and was obtained by Schultz and co-workers [1,43]. The last data set, LD_{50}, represents the concentration of chemicals that kill half of rats when orally ingested. This data set was constructed from the ChemIDplus database (http://chem.sis.nlm.nih. gov/chemidplus/chemidheavy.jsp)

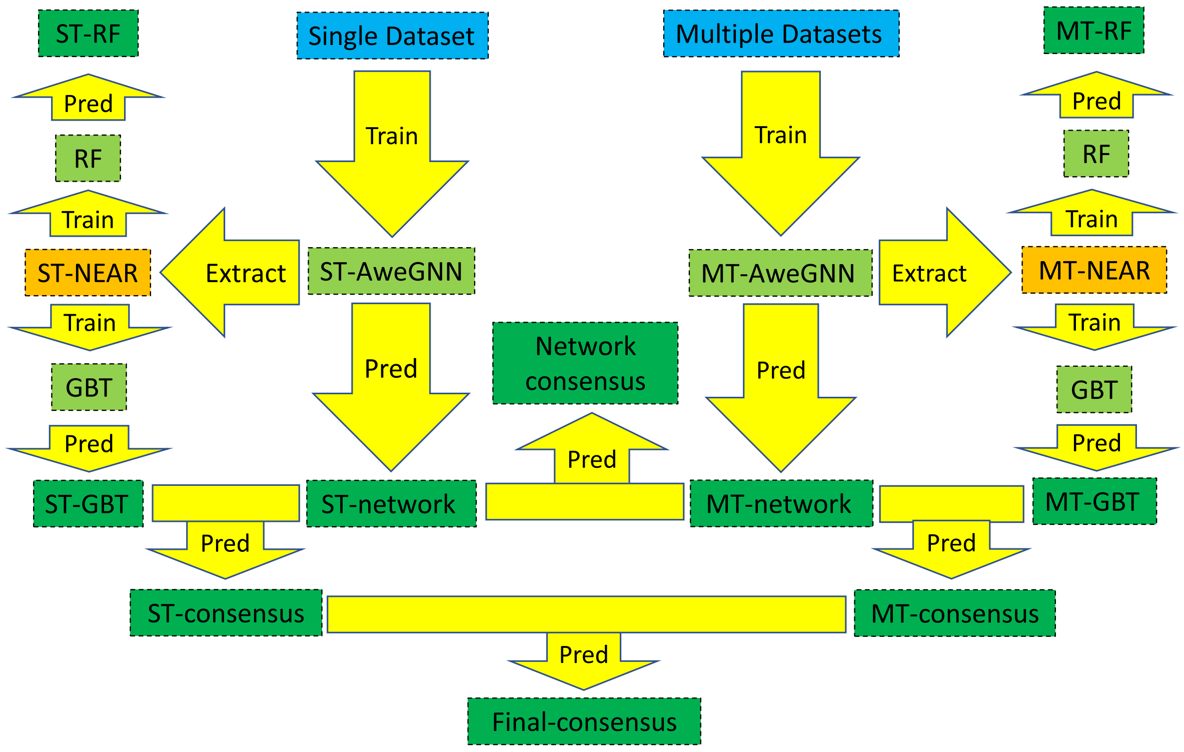


Fig. 6. Pictorial representation of the overall strategy. The arrows show the flow of the processes of training, extracting representations from trained AweGNNs, and making predictions. Blue corresponds to molecular data sets, light green corresponds to machine learning models, orange corresponds to molecular representations, and dark green corresponds to predictions.

and then filtered according to several criteria [21].

The final sets in this work are identical to the sets preprocessed to develop the Toxicity Estimation Software Tool (TEST) [21]. The 2D sdf format molecular structures and toxicity endpoints are available on the TEST website. 3D mol2 format molecular structures were created with the Schrödinger software in an earlier work [40]. We should note that the units of the toxicity endpoints are not uniform between the data sets. The LD_{50} set endpoints are in -$\log_{10}(T\ \ \text{mol}/\text{kg})$ while the endpoints of the remaining sets are in units of -$\log_{10}(T\ \ \text{mol}/\text{L})$. No attempt has been made to rescale the values. Finally, the data sets all have differing sizes and compositions, and so the effectiveness of our method varies greatly between them.

Statistics of each data set are detailed below in Table 1. Numbers inside parentheses indicate the actual number of molecules that were used for training and evaluating models. The first 3 data sets include all available molecules, but for the last data set (LD_{50}), some molecules were dropped out due to force field failures when applying the Schrödinger software. Despite this, our coverage is greater than any of the TEST models and so is more widely applicable in use.

3.2.1. *LC_{50}-DM (Daphnia Magna) set*

The *Daphnia magna* LC_{50} set is the smallest data set with 283 molecules in the training set and 70 molecules in the test set. Given the small size of the data set, it can be difficult to train robust QSAR models, thus multi-scale models are extremely important for obtaining reasonable results. Table 2 shows the results of various QSAR models on the LC50DM data set. The TEST consensus had the highest R^2 score ($R^2 = 0.739$) out of all the models shown, although Wu et al. [40] reported a much higher R^2 score of 0.788, which does not appear in the table, when only using topological descriptors for a multi-task model. This high result with fewer descriptors may be due to the nature of neural networks to overfit when trained on small data sets and many descriptors. When comparing the result of Wu-MT as reported in the table (using all descriptors) to the TEST consensus, we notice that although the R^2 score was lower, the RMSE and MAE are better. The group contribution scored exceptionally well in RMSE and MAE, but doubt was cast on the accuracy of those results [40].

The performance of our ensemble models was fairly poor in comparison to the results from Wu. Our RF-ST and RF-MT models scored 0.439 and 0.443 respectively vs. the 0.460 R^2 score achieved by Wu-RF. Our MT-GBT model was a bit better in performance than the Wu-RF model, but it could not beat the Wu-GBT model. We do see however that our MT-GBT model had outperformed the ST-GBT model noticeably, going from an R^2 score of 0.457–0.471, showing a slight benefit from using the MT-NEAR for GBT models. As for the network models, we see a descent show for our ST-network model vs. the Wu-ST model. Although ST-network has an R^2 score of 0.448 vs. the 0.459 for Wu-ST, we see that the ST-network RMSE of 1.315 beats quite decisively the RMSE of 1.407 for Wu-ST. Our MT-network model had unfortunately performed much worse than the Wu-MT model, even though its results are comparable with many of the TEST models reported. The Wu-MT model R^2 score of 0.726 is significantly higher than the score of 0.664 from MT-network. All of our other models do not do well enough to the comment on.

Table 1
Set statistics for quantitative toxicity data.

data set	# of molecules	train set size	test set size	max value	min value
LC_{50}-DM	353	283	70	10.064	0.117
LC_{50}	823	659	164	9.261	0.037
IGC_{50}	1792	1434	358	6.36	0.334
LD_{50}	7413 (7398)	5931 (5919)	1482 (1479)	7.201	0.291

Table 2
Comparison of prediction results for the LC50DM test set.

model	R^2	$\frac{R^2-R_0^2}{R^2}$	k	RMSE	MAE	coverage
Results with TEST models						
Hierarchical [21]	0.695	0.151	0.981	0.979	0.757	0.886
single model [21]	0.697	0.152	1.002	0.993	0.772	0.871
FDA [21]	0.565	0.257	0.987	1.190	0.909	0.900
group contribution [21]	0.671	0.049	0.999	0.803	0.620	0.657
nearest neighbor [21]	0.733	0.014	1.015	0.975	0.745	0.871
TEST consensus [21]	0.739	0.118	1.001	0.911	0.727	0.900
Results with previous methods from our group						
Wu-RF [40]	0.460	1.244	0.955	1.274	0.958	1.000
Wu-GBT [40]	0.505	0.448	0.961	1.235	0.905	1.000
Wu-ST [40]	0.459	0.278	0.933	1.407	1.004	1.000
Wu-MT [40]	0.726	0.003	1.017	0.905	0.590	1.000
Wu-consensus [40]	0.678	0.282	0.953	0.978	0.714	1.000
Ensemble models						
ST-RF	0.439	0.014	0.956	1.312	0.983	1.000
ST-GBT	0.457	0.004	0.966	1.280	0.954	1.000
MT-RF	0.443	0.012	0.960	1.304	0.985	1.000
MT-GBT	0.471	0.003	0.970	1.261	0.953	1.000
AweGNN models						
ST-network	0.448	0.060	0.959	1.315	0.959	1.000
MT-network	0.664	0.000	0.983	1.002	0.741	1.000
Consensus models						
network-consensus	0.583	0.005	0.984	1.112	0.826	1.000
ST-consensus	0.465	0.015	0.933	1.272	0.933	1.000
MT-consensus	0.602	0.000	0.981	1.090	0.820	1.000
final-consensus	0.541	0.001	0.979	1.171	0.872	1.000

3.2.2. *Fathead minnow LC_{50} set*

The fathead minnow LC_{50} set was randomly divided into a training (80% of the entire set) and a test set (20% of the entire set) [21]. Table 3 shows the performance of all of the various models trained and tested on the LC50 data. This is the second smallest data set that we are analyzing in this work. The best TEST model is again the TEST consensus, at an R^2 score of 0.728. Notice that this is lower than the previous TEST consensus on the LC_{50} at an R^2 of 0.739, although the coverage increase

Table 3
Comparison of prediction results for the LC50 test set.

model	R^2	$\frac{R^2-R_0^2}{R^2}$	k	RMSE	MAE	coverage
Results with TEST models						
hierarchical [21]	0.710	0.075	0.966	0.810	0.574	0.951
single model [21]	0.704	0.134	0.960	0.803	0.605	0.945
FDA [21]	0.626	0.113	0.985	0.915	0.656	0.945
group contribution [21]	0.686	0.123	0.949	0.810	0.578	0.872
nearest neighbor [21]	0.667	0.080	1.001	0.876	0.649	0.939
TEST consensus [21]	0.728	0.121	0.969	0.768	0.545	0.951
Results with previous methods from our group						
Wu-RF [40]	0.727	0.322	0.948	0.782	0.564	1.000
Wu-GBT [40]	0.761	0.102	0.959	0.719	0.496	1.000
Wu-ST [40]	0.692	0.010	0.997	0.822	0.568	1.000
Wu-MT [40]	0.769	0.009	1.014	0.716	0.466	1.000
Wu-consensus [40]	0.789	0.076	0.959	0.677	0.446	1.000
Ensemble models						
ST-RF	0.697	0.028	1.018	0.836	0.589	1.000
ST-GBT	0.687	0.000	0.995	0.820	0.562	1.000
MT-RF	0.703	0.029	1.019	0.829	0.582	1.000
MT-GBT	0.706	0.001	0.998	0.795	0.543	1.000
AweGNN models						
ST-network	0.682	0.003	0.987	0.830	0.566	1.000
MT-network	0.749	0.000	0.999	0.735	0.481	1.000
Consensus models						
network-consensus	0.739	0.000	0.996	0.748	0.505	1.000
ST-consensus	0.711	0.000	0.994	0.788	0.540	1.000
MT-consensus	0.747	0.001	1.001	0.739	0.494	1.000
final-consensus	0.737	0.001	0.998	0.753	0.507	1.000

from 0.900 to 0.951. For the Wu models, the best result is a whopping 0.789 R^2 score for Wu-consensus. Wu-consensus also boasts the best overall RMSE and MAE. The other 2 high scoring models from that category are Wu-MT and Wu-GBT, which come at R^2 scores of 0.769 and 0.761, respectively (see Table 3).

Our ensemble models do comparably well when compared to the TEST models, except for the TEST consensus. Our ST-network model did similarly in performance to many of the TEST models, notably FDA, group contribution, and nearest neighbor. Our best model, MT-network, with an R^2 score of 0.749 and RMSE of 0.735 beats all TEST models, and all of our consensus models do equal to or better than the TEST consensus model, except for the ST-consensus, which was weighed down by the oddly poor performance of the ST-GBT model (ST-GBT performed more poorly than did the ST-RF model, with an R^2 score of 0.687 vs. 0.697).

In comparison to the Wu models, we notice that all of our ensemble models are far behind in performance to either Wu-RF or Wu-GBT, with our best R^2 score of 0.706 from our MT-GBT model being overtaken by the lowest ensemble score of 0.727 from the Wu-RF model. The performance of our ST-network model is comparable to Wu-ST with an R^2 of 0.682–0.692, with even closer RMSE and MAE scores. As far as the MT models go, we see a much greater difference between MT-network and Wu-MT. We cannot come close to the R^2 score of 0.789 from the Wu-MT model, which is the best of the models from Wu for this data set [40].

3.2.3. *Tetraphymena pyriformis IGC$_{50}$ set*

The IGC$_{50}$ data set is the second largest of the data sets we are analyzing. The diversity of the molecules is relatively low compared to the other data sets, which allows for more coverage in the TEST models. The amount of data points in the set is large enough to train robust models, which translates into the high R^2 scores for the models that are compared in this section. The results for this data set are shown in Table 4. We notice that the TEST models are far more variant in their results than in the previous 2 data sets, with R^2 scores ranging from 0.600 to 0.764. Again, the TEST consensus gets the highest R^2 score (0.764). Among the Wu models, the Wu-consensus model is again the supreme champion with an R^2 score of 0.802. As usual, the Wu-MT model also did very well, trumping all TEST models with respect to every metric. We also introduce the best IGC$_{50}$ model from the work done by Nguyen et al. [27] in which GBT models were trained on representations derived from differential geometry based descriptors. Though the model was trained without help from other data sets, it was able to defeat the Wu-MT model in all metrics. The GBT Nguyen-best model is however narrowly defeated by the Wu-GBT model with an R^2 score of 0.781 compared to 0.787.

Our ensemble models are relatively low in comparison to the performance of the rest of the models, although our MT-GBT model does perform better than all TEST models except for the TEST consensus model. Our ST-network model outperforms all TEST models single-handedly with an R^2 score of 0.778. All of our remaining models outperform our own ST-network model, and thus also every TEST model as well.

Our ensemble models under-perform yet again in comparison to the RF and GBT models set forth by Wu. When comparing our network models, we see that our ST-network model strongly defeats the Wu-ST model with an R^2 score of 0.749 and even outperforms the Wu-MT model with an R^2 of 0.770. ST-network is, however, beaten by Wu-consensus, and even by the model put forth by Nguyen, that is, Nguyen-best. Our MT-network model is the best scoring model in the whole table in all metrics except for MAE, with an R^2 score of 0.803, RMSE of 0.436, and MAE of 0.310. Only Wu-consensus has a lower MAE of 0.305, but that MAE is beaten by our model, network-consensus, with an MAE of 0.304. Although MT-network technically performs slightly better than Wu-consensus, they are effectively identical in performance.

3.2.4. *Oral rat LD$_{50}$ set*

The oral rat LD$_{50}$ set contains the most molecules. The data set is quite large (7413 molecular compounds), so naturally full coverage is not available for any of the methods proposed, although most models still have very high coverage. The labels of this data set are quite difficult to predict because of the high experimental uncertainty in obtaining the toxicity endpoints, as noted in Zhu et al. [42]. Table 5 shows the results. As in Wu et al. [40], we omit the single model and group contribution TEST methods from the table in our analysis. As always, the TEST consensus provides the greatest results amongst the TEST models. For

Table 4
Comparison of prediction results for the IGC50 test set.

model	R^2	$\frac{R^2 - R_0^2}{R^2}$	k	RMSE	MAE	coverage
Results with TEST models						
hierarchical [40]	0.719	0.023	0.978	0.539	0.358	0.933
FDA [40]	0.747	0.056	0.988	0.489	0.337	0.978
group contribution [40]	0.682	0.065	0.994	0.575	0.411	0.955
nearest neighbor [40]	0.600	0.170	0.976	0.638	0.451	0.986
TEST consensus [40]	0.764	0.065	0.983	0.475	0.332	0.983
Results with previous methods from our group						
Wu-RF [40]	0.736	0.235	0.981	0.510	0.368	1.000
Wu-GBT [40]	0.787	0.054	0.993	0.455	0.316	1.000
Wu-ST [40]	0.749	0.019	0.982	0.506	0.339	1.000
Wu-MT [40]	0.770	0.000	1.001	0.472	0.331	1.000
Wu-consensus [40]	0.802	0.066	0.987	0.438	0.305	1.000
Nguyen-best [27]	0.781	0.004	1.003	0.463	0.324	1.000
Ensemble models						
ST-RF	0.713	0.013	1.006	0.535	0.377	1.000
ST-GBT	0.745	0.000	0.994	0.496	0.329	1.000
MT-RF	0.716	0.013	1.006	0.532	0.377	1.000
MT-GBT	0.753	0.000	0.995	0.489	0.327	1.000
AweGNN models						
ST-network	0.778	0.000	1.003	0.463	0.309	1.000
MT-network	0.803	0.000	0.999	0.436	0.310	1.000
Consensus models						
network-consensus	0.799	0.000	1.001	0.440	0.304	1.000
ST-consensus	0.777	0.000	1.000	0.464	0.309	1.000
MT-consensus	0.795	0.000	0.998	0.445	0.305	1.000
final-consensus	0.789	0.000	0.999	0.451	0.306	1.000

Table 5
Comparison of prediction results for the LD50 test set.

model	R^2	$\frac{R^2 - R_0^2}{R^2}$	k	RMSE	MAE	coverage
Results with TEST models						
hierarchical [21]	0.578	0.184	0.969	0.650	0.460	0.876
FDA [21]	0.557	0.238	0.953	0.657	0.474	0.984
nearest neighbor [21]	0.557	0.243	0.961	0.656	0.477	0.993
TEST consensus [21]	0.626	0.235	0.959	0.594	0.431	0.984
Results with previous methods from our group						
Wu-RF [40]	0.619	0.728	0.949	0.603	0.452	0.997
Wu-GBT [40]	0.630	0.328	0.960	0.586	0.441	0.997
Wu-ST [40]	0.614	0.006	0.991	0.601	0.436	0.997
Wu-MT [40]	0.626	0.002	0.995	0.590	0.430	0.997
Wu-consensus [40]	0.653	0.306	0.959	0.568	0.421	0.997
Ensemble models						
ST-RF	0.606	0.013	1.003	0.612	0.452	0.998
ST-GBT	0.643	0.002	0.995	0.578	0.424	0.998
MT-RF	0.596	0.012	1.003	0.619	0.456	0.998
MT-GBT	0.641	0.002	0.995	0.580	0.426	0.998
AweGNN models						
ST-network	0.660	0.001	0.995	0.563	0.406	0.998
MT-network	0.658	0.001	0.993	0.565	0.411	0.998
Consensus models						
network-consensus	0.665	0.000	0.995	0.558	0.404	0.998
ST-consensus	0.665	0.001	0.997	0.559	0.406	0.998
MT-consensus	0.664	0.001	0.996	0.560	0.409	0.998
final-consensus	0.667	0.001	0.997	0.557	0.405	0.998

this data set, the effectiveness of the Wu models wanes, with the Wu-MT model failing to perform decisively better than the TEST consensus (although, Wu-MT does indeed do slightly better in RMSE and MAE in comparison to TEST consensus).

For this data set, we see the best performance from our models. Even our ensemble models do quite well. In fact, our ST-GBT and MT-GBT models, with respective R^2 scores of 0.643 and 0.641, outperform any TEST model and any Wu models aside from the Wu-consensus model. All of our non-ensemble models out-perform every single other model, including Wu-consensus, with our best performing model, final-consensus, having an R^2 of 0.667, RMSE of 0.557, and MAE of 0.405. This decisively beats the Wu-consensus model having an R^2 score of 0.653, RMSE of 0.568, and MAE of 0.421.

We notice that for this data set, we finally required the consensus of the GBT models to get our best prediction, as opposed to the other data sets in which we obtained the best result from our MT-network models. Our AweGNN models seem to shine relative to other groups models when the data sets are the largest, in which the networks are able to generate their most generalizable representations, and the predictions are the most accurate.

3.3. Solvation data set

In this section, we explore the results of the solvation data set, model III. Model III has a total of 387 molecules (excluding ions), and is split into a training set of 293 molecules and a test set of 94 molecules. We use the same training set, but we omit molecules based on obscure chemical names in the PubChem database and due to some difficulties with the Schrödinger software in generating mol2 files. The test set is unaltered, so these difficulties leading to our smaller training set of 280 molecules should disfavor our method. In addition to this, there is only one data set for solvation that we analyze, so we cannot apply our MT method. We simply train ST models and report one consensus with the GBT model and the network model, following the same procedure as before. The results are shown below in Table 6, where they are compared to other models mentioned previously [27,36,38].

We see that in this instance, we get our best performance relative to other group's models. This is quite surprising, as we found that with the toxicity data, the AweGNN seemed to perform very well only when the data sets were very large. Our greatest model, the ST-network model, significantly outperforms every other model in every metric. Even our GBT model is able to greatly outperform all other models outside this group, though our RF model performs only satisfactory. In fact, the supplementary material from Nguyen et al. [27] contains models whose kernel parameters were optimized specifically on the test data, giving a further advantage. These can be seen in Table 7. Again, we see that even our GBT model is able to win in every metric against every model in this table, let alone our ST-network with an MAE gap of 0.145, an RMSE gap of 0.229, and an R^2 score gap of 0.032 when compare to the best metrics of any chosen opposing model. This analysis shows that there is great promise for applying the AweGNN method for solvation free energy prediction.

Table 6
Comparison of prediction results for the solvation test set.

model	MAE (kcal/mol)	RMSE (kcal/mol)	R^2
Results from outside sources			
WSAS [38]	0.66	–	–
FFT [36]	0.57	–	–
Results from Nguyen et al. [27]			
$\text{EIC}^{H}_{E,3.5,0.3}$	0.575	0.921	0.904
$\text{EIC}^{H}_{E,3.5,0.3;E,2.5,1.3}$	0.558	0.857	0.920
$\text{EIC}^{H}_{L,3,1.3}$	0.592	0.931	0.906
$\text{EIC}^{H}_{L,3,1.3;L,6.5,0.3}$	0.608	0.919	0.907
Consensus^{H}	0.567	0.862	0.920
Ensemble models			
ST-RF	0.698	1.001	0.893
ST-GBT	0.496	0.666	0.951
AweGNN model			
ST-network	0.373	0.569	0.963
Consensus model			
ST-consensus	0.401	0.583	0.962

4. Discussion

In this section, we will discuss the impact of automated parameter selection, analyze some elements of feature importance, and discuss a new paradigm of auto-parametrized kernel-based networks that could lead to many more possibilities.

4.1. Impact of automating selection of Kernel parameters

When selecting kernel parameters to optimize the choice of representation, our group was previously using the grid search method to determine which parameters were optimal [25]. This is not so much of a hindrance if there are only a handful of parameters to tune, but as the number of parameters to tune increases, the number of models that need to be trained increases exponentially. In addition, grid searches require a discrete set of parameters to experiment with, which is a coarse way to tune parameters and leads to some inefficiency. These problems can severely restrict the potential of these kernel-based representations, and largely restrict us to the use of machine learning algorithms that do not have many tunable parameters, such as RF or GBT.

One of our goals in this work was to alleviate these issues. Automatically updating kernel parameters is a great solution in theory and in practice, as we have seen with the success of our AweGNN in this work. The AweGNN seeks the optimal choice of parameters within a continuous space, so the parameter selection is done in a smooth way and can select the values that are not present in a grid search. In our case, we incorporate the parameter selection into the gradient descent process of training a neural network. This association with the gradient descent allows us to update very many kernel parameters at the same time as the weights of the neural network, but has a moderate computational cost associated with it that is commensurate with the computational cost of a moderately sized network. More specifically, a quick analysis on the IGC_{50} data shows that a network consisting of the GGR layer (producing 400 features) with normalization takes the same amount of time on average to cycle though an epoch as does the network with the GGR + normalization layer along with a 4-layer network of 1600 neurons each, thus the GGR + normalization layer is equivalent to a 4-layer 1600 neuron network in terms of computational cost. This cost is certainly not prohibitive, and ultimately saves us an inordinate amount of time, especially since we are able to tune 200 kernel parameters simultaneously (which would be impossible with a grid search).

As kernel methods require careful tuning to reach their full potential, neural networks were effectively off limits for training effective models with kernel-based representations. The AweGNN proves that kernel-based representations can be used to train high performing models and brings many new possibilities for developing excellent predictive models for biomolecular data.

Table 7
Supplementary results for solvation data set from Nguyen et al. [27].

Method	MAE (kcal/mol)	RMSE (kcal/mol)	R^2
$^*\text{EIC}^{H}_{E,3.5,0.3}$	0.575	0.921	0.904
$^*\text{EIC}^{HH}_{E,3.5,0.3;E,4.0,1.3}$	0.518	0.812	0.929
$^*\text{EIC}^{H}_{L,5,0.3}$	0.579	0.862	0.917
$^*\text{EIC}^{HH}_{L,5,0.3;L,0.5,0.9}$	0.559	0.842	0.922
$^*\text{Consensus}^{H}$	0.524	0.798	0.931

4.2. *Feature importance and analyzing the trajectories of the Kernel parameters*

To find the most important features that influenced our predictions, we look at the feature importance ranking of the random forest (RF) models that were trained on the network-enabled automatic representations (NEARs) generated by the AweGNNs. This analysis will pinpoint the specific element-pair interactions that contributed the most to the creation of our predictive models. Then, we can analyze the average trajectories and final states of the kernel functions from the element-specific groups that generate the most important features. The average η values can tell us about the most beneficial interaction distances between those element pairs, and the average κ values can tell us about the most favorable degree of sharpness of the cutoff for including the edge cases centered around those η values.

By closely examining the feature importance over all data sets and all models, we notice that, in general, the most important interactions tend to be the ones between hydrogen-hydrogen, carbon-carbon, and the interaction between those two elements. This would make sense, since the carbons and hydrogens are the most numerous elements in these structures and will be interacting most with the environment. The second most important features are generated from the interactions between the hydrogen-oxygen and carbon-oxygen pairs. This again seems to be a natural result, for this can be indicating the importance of the polar bonds. Aside from these groups, however, there seems to be no broader pattern. Different data sets seem to prioritize different element specific groups, especially the largest data set, LD_{50}, in which there are a great diversity of element pair interactions that are important, as can be seen in Fig. 7a, which illustrates the feature importance associated with each pair through a heat map.

A more contained display of the feature importance is portrayed in Fig. 7b, based on the RF models trained to predict the toxicity for the IGC_{50} data set with the NEARs generated from the multi-task (MT) AweGNN models. The feature importance values are concentrated within the pairs consisting of the hydrogen, carbon, nitrogen, and oxygen elements. For the complete list of feature importance heat maps of all the data sets, including feature importance measured by the gradient boosting models, you may reference the Supporting information, Figs. S1–S16.

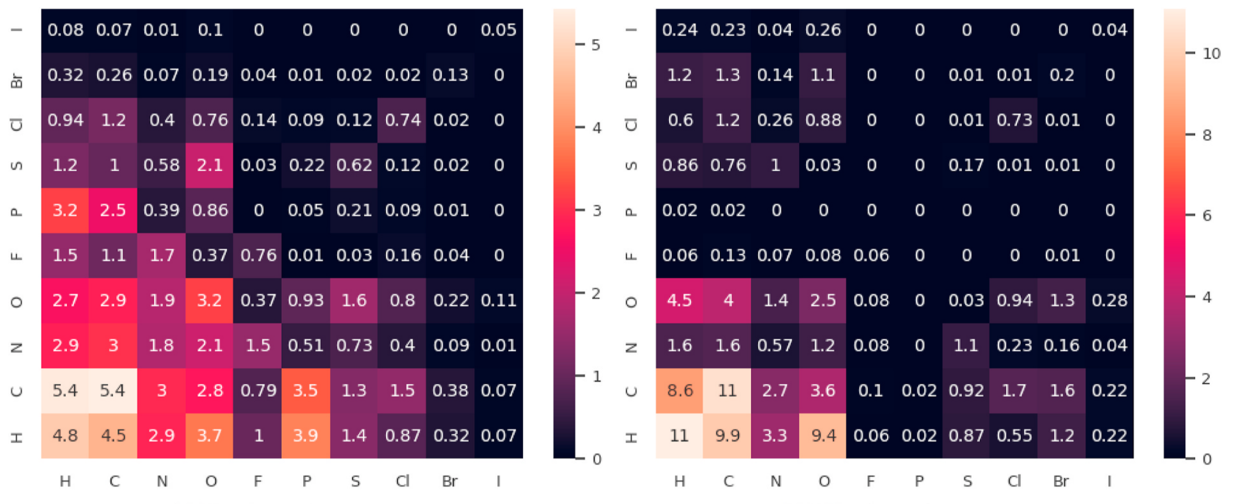
To go into more detail of what is happening in each element-specific group, we analyze how the parameters of the kernel functions of the top 10 groups evolve on average. This evolution can be seen in Fig. 8.

As the training progresses, element pairs have their kernel parameters tend towards various η-κ combinations. Some element pairs have high η values and moderate κ values, such as the H–H and C–N groups, while others tend towards low η and low κ values (i.e, H–O), or even low η and high κ (O–C). There are many physical interpretations that could be valid, for example, the low η and κ value for the H–O group could be a measure of the potential for hydrogen bond interactions with the environment, or even internal short-range interactions that assess the stability of the molecule. In any case, we may note the convergence of the η values especially for they might reveal important cutoff distances that contribute to the measurement of toxicity.

4.3. *Limitations and advantages*

One of the limitations present in the study is with regards to applying the AweGNN to very large molecules or systems of molecules. If too many atoms are involved in the calculation of these features, then the computational cost may become unbearable. Some provisions or rules can be made to avoid heavy computational costs while dealing with large scale biomolecular structures or systems of molecules. For example, our method may be naturally extended to the problem of protein-ligand binding. We can measure the interaction strength between atoms of a specific element type in the protein and atoms of another element type in the ligand, instead of the interactions between the atoms of two element types contained within a single molecule. A standard cutoff distance from the ligand is applied to focus on the strongest interactions, thereby leaving a reduced number of atoms for the analysis. A cutoff that is very large might include too many atoms from the protein and thereby incur a heavy computational cost that is far beyond that of the small molecular data sets that we have used in this study. Small cutoffs can be feasible and perhaps still quite effective since long-range interactions may not carry as much weight in terms of performance.

The advantage of the AweGNN is its ability to shape the kernel function for each element-specific group so as to provide the optimal filter for capturing the most important interactions between elements within a molecule. Though there is much potential in this approach, there is a danger of over-fitting with smaller data sets. This is perhaps why we see the greatest performance with the largest LD_{50} data set, and so practically we may want to train on the largest data sets that we can.



(a) LD_{50} feature importance map (b) IGC_{50} feature importance map

Fig. 7. Feature importance of MT-AweGNNs when applied to different data sets. We generate heat maps based on the average of feature importance across 42 RF models trained on the corresponding NEARs of the MT-AweGNNs. The average feature importance of the 4 features of each element-specific group are summed together to get the final scores shown above in the maps. Fig. 7a shows the results for the LD_{50} MT-RF models and Fig. 7b shows the results for the IGC_{50} MT-RF models. Note that the maps are asymmetric due to the way that the electrostatics-based features of a group are generated.

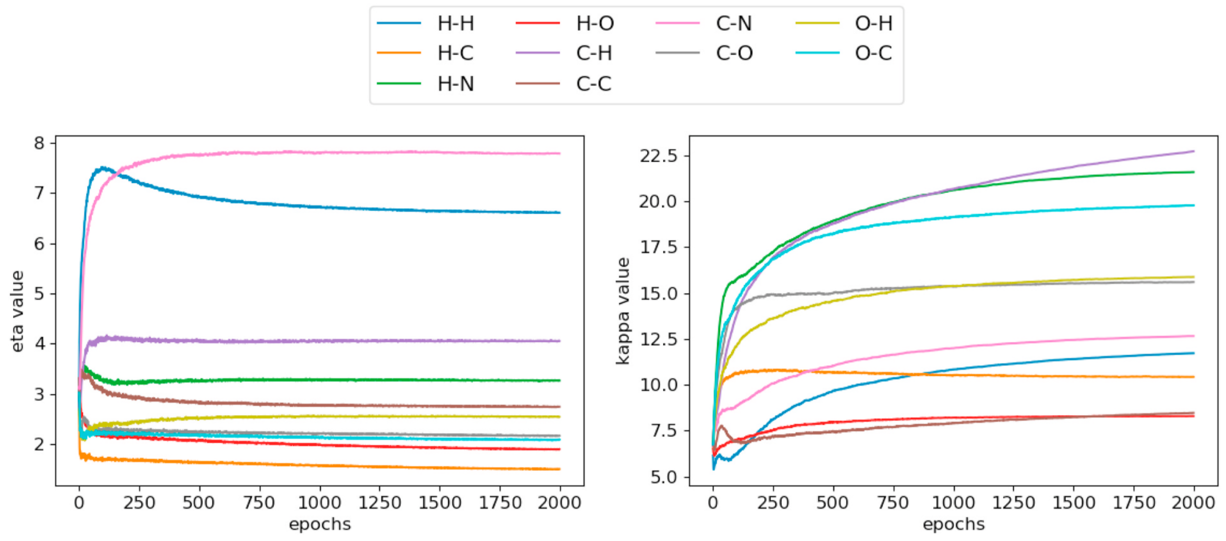


Fig. 8. Average trajectories of kernel parameters for 42 MT-AweGNN models. The graph on the left shows how the average of the η values of our 42 MT models for 8 different element-specific groups changes as the models are trained. On the right, we see the κ counterpart of this analysis.

However, we notice also that the AweGNN was incredibly successful with the small solvation energy data set; and for the two smallest data sets, LC_{50}-DM and LC_{50}, the results for the single task network are similar to the RF models, which are known to be quite robust against over-fitting. Thus, the danger of over-fitting may not be as present as originally thought. Also, it may be the case that the multi-task method is less effective when applying the AweGNN, for it may fit its features too specifically to a particular task, and therefore be influenced by the larger data sets more so than other methods, like Wu's [40].

The computational cost associated with training our multi-task (MT) AweGNN model, which is our largest computational endeavor, is under 135 min. The largest single task model (LD_{50} model), takes under 120 min, while the second largest IGC_{50} data set is trained in approximately 20 min, with the training time for the smaller data sets being significantly lower (a handful of minutes). While the training time of our MT-AweGNN takes almost 70% longer than, for instance, the MT-Wu model (80 min training time) [40], we can see that the MT AweGNN can still perform better on the larger data sets, so the 70% increase in training time may be worth the performance boost when provided with more samples. In addition to this, we note that our features are simple kernel-based features, and do not require auxiliary physical descriptors to achieve stellar performance. The AweGNN can in fact use those same auxiliary features to bolster its own predictions, and is more flexible in that it can also combine multiple kernel types to generate multi-scale representations that have been highly successful in previous works [27,28]. Finally, as our features are kernel-based, we noted in section 4.1 that these features would usually require a grid-search to optimize the kernel parameters, but in our case, we are able to automate this process while simultaneously training our network, thereby saving time in that respect.

5. Conclusion

Recent years have witnessed much effort in developing mathematical representations for the machine learning predictions of chemical and biological properties that are crucial to drug discovery. Advanced mathematical tools from fields such as graph theory [26,28], differential geometry [27], algebraic topology [40], and other mathematical area, have been developed and demonstrated their superb performance. Many of these representations were generated with a choice of kernel functions in which each function depends on its own choice of parameters. These kernel-based molecular representations then have to be fine tuned to optimize their effectiveness for training machine learning models. Increasing parameter choices leads to a rapid increas in the computational cost of optimization. This problem deteriorates in the case of deep neural networks because of the already cumbersome task of choosing network hyperparameters. Motivated by the automated feature extraction in convolutional neural network (CNN), we propose an auto-parametrized element-specific graph neural network (AweGNN) to automate and optimize the parameter selection in our geometric graph approach. The resulting representation from training the AweGNN is called a network-enabled automatic representation (NEAR). NEARs can be used as input features for other machine learning models, such as random forest (RF) and gradient-boosting tree (GBT) models.

AweGNN and NEAR-based ensemble methods are validated with five data sets from quantitative toxicity and solvation predictions. Both toxicity and solvation are important for lead hit, and structure optimization in drug discovery. Four quantitative toxicity data sets: 96 h fathead minnow LC_{50}, 48 h *Daphnia Magna* LC_{50} data set (or LC_{50}-DM), 40 h Tetrahymena pyriformis IGC_{50} data set, and the oral rat LD_{50} data set were used in our work. Our models were compared to state-of-the-art models in various literature, i.e., the Toxicity Estimation Software Tool (TEST) [21] listed by the United States Environmental Protection Agency (EPA), a previous work by Wu et al. [40], and another work by Nguyen et al. [27]. Given that we had 4 toxicity data sets of various sizes with similar prediction tasks, we were able to employ the multi-task (MT) learning method to greatly improve our performance. Our top models were able to out-compete all TEST models in all but the smallest data sets. When comparing with the top Wu models, our top models were able to perform just as well or better when restricting to the largest 2 data sets, although our general models for the smaller data sets were still able to perform relatively well when compared to many of the earlier models.

To broaden our application, we also tested the AweGNN on the solubility data set, Model III, used in the work by Nguyen al [27]. For this instance, we were not able to apply MT learning because we only had one data set to work with. Despite this, we were able to greatly outperform any other models that we compared [27,36,38]. Although there is a positive correlation between the relative effectiveness of the AweGNN and the data set size with respect to the 4 toxicity data sets, we see that in the case of solubility, the AweGNN can perform exceptionally well even with a very small data set. Our work showcases the impressive predictive capabilities of the AweGNN and ultimately introduces new potential to improve the effectiveness of previous mathematical methods.

Model availability

The software for the Geometric Graph Representation (GGR) layer and the AweGNN models is available at https://github.com/timothyszocinski/AweGNN/.

Data availability

The mol2 files that were used for the toxicity and solvation data sets can be found at https://weilab.math.msu.edu/Database/.

Declaration of competing interest

We do not know any existing conflict interest.

Acknowledgments

This work was supported in part by NIH grant GM126189, NSF Grants DMS-1721024, DMS-1761320, DMS-2052983, DMS-2053284, and IIS1900473, NASA 80NSSC21M0023, Michigan Economic Development Corporation, George Mason University award PD45722, Bristol-Myers Squibb BMS-65109, Pfizer, and University of Kentucky Start-up fund. The authors thank The IBM TJ Watson Research Center, The COVID-19 High Performance Computing Consortium, and NVIDIA for computational assistance.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.compbiomed.2021.104460.

References

[1] K.S. Akers, G.D. Sinks, T.W. Schultz, Structure–toxicity relationships for selected halogenated aliphatic chemicals, Environ. Toxicol. Pharmacol. 7 (1) (1999) 33–39.

[2] P.J. Ballester, A. Schreyer, T.L. Blundell, Does a more precise chemical description of protein–ligand complexes lead to more accurate prediction of binding affinity? J. Chem. Inf. Model. 54 (3) (2014) 944–955.

[3] M. Barycki, A. Sosnowska, K. Jagiello, T. Puzyn, Multi-objective genetic algorithm (MOGA) as a feature selecting strategy in the development of ionic liquids' quantitative toxicity–toxicity relationship models, J. Chem. Inf. Model. 58 (12) (2018) 2467–2476.

[4] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, K. Smith Cython, The best of both worlds, Comput. Sci. Eng. 13 (2) (2010) 31–39.

[5] S. Brandt, F. Sittel, M. Ernst, G. Stock, Machine learning of biomolecular reaction coordinates, J. Phys. Chem. Lett. 9 (9) (2018) 2144–2150.

[6] K.T. Butler, D.W. Davies, H. Cartwright, O. Isayev, A. Walsh, Machine learning for molecular and materials science, Nature 559 (7715) (2018) 547–555.

[7] R. Caruana, Learning to Learn, Springer, 1998.

[8] D. Ciregan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 3642–3649.

[9] S.J. Darnell, L. LeGault, J.C. Mitchell, KFC server: interactive forecasting of protein interaction hot spots, Nucleic Acids Res. 36 (suppl_2) (2008) W265–W269.

[10] L. Deng, G. Hinton, B. Kingsbury, New types of deep neural network learning for speech recognition and related applications: an overview, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 8599–8603.

[11] A. Golbraikh, M. Shen, Z. Xiao, Y.-D. Xiao, K.-H. Lee, A. Tropsha, Rational selection of training and test sets for the development of validated qsar models, J. Comput. Aided Mol. Des. 17 (2) (2003) 241–253.

[12] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.

[13] J. Jiang, R. Wang, M. Wang, K. Gao, D.D. Nguyen, G.-W. Wei, Boosting tree-assisted multitask deep learning for small scientific datasets, J. Chem. Inf. Model. 60 (3) (2020) 1235–1244.

[14] A. Karim, A. Mishra, M.H. Newton, A. Sattar, Efficient toxicity prediction via simple features using shallow neural networks and decision trees, ACS Omega 4 (1) (2019) 1874–1888.

[15] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, 2014 arXiv preprint arXiv:1412.6980.

[16] S.K. Lam, A. Pitrou, S. Seibert Numba, A llvm-based python jit compiler, in: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, 2015, pp. 1–6.

[17] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[18] L. Li, C. Li, E. Alexov, On the modeling of polar component of solvation energy using smooth Gaussian-based dielectric function, J. Theor. Comput. Chem. 13 (2014) 1440002, 03.

[19] R. Liu, M. Madore, K.P. Glover, M.G. Feasel, A. Wallqvist, Assessing deep and shallow learning methods for quantitative prediction of acute chemical toxicity, Toxicol. Sci. 164 (2) (2018) 512–526.

[20] A.V. Marenich, C.J. Cramer, D.G. Truhlar, Performance of AM6, SM8, and SMD on the sampl1 test set for the prediction of small-molecule solvation free energies, J. Phys. Chem. B 113 (14) (2009) 4538–4543.

[21] T.M. Martin, User's Guide for T.E.S.T. (*Version 4.2) (Toxicity Estimation Software Tool): A Program To Estimate Toxicity From Molecular Structure*, USEPA, 2016.

[22] T.M. Martin, P. Harten, R. Venkatapathy, S. Das, D.M. Young, A hierarchical clustering methodology for the estimation of toxicity, Toxicol. Mech. Methods 18 (2–3) (2008) 251–266.

[23] Z. Meng, D.V. Anand, Y. Lu, J. Wu, K. Xia, Weighted persistent homology for biomolecular data analysis, Sci. Rep. 10 (1) (2020) 1–15.

[24] D.L. Mobley, K.L. Wymer, N.M. Lim, J.P. Guthrie, Blind prediction of solvation free energies from the Sampl4 challenge, J. Comput. Aided Mol. Des. 28 (3) (2014) 135–150.

[25] D.D. Nguyen, Z. Cang, G.-W. Wei, A review of mathematical representations of biomolecular data, Phys. Chem. Chem. Phys. 22 (8) (2020) 4343–4367.

[26] D.D. Nguyen, G.-W. Wei Agl-Score, Algebraic graph learning score for protein–ligand binding scoring, ranking, docking, and screening, J. Chem. Inf. Model. 59 (7) (2019) 3291–3304.

[27] D.D. Nguyen, G.-W. Wei, DG-GL: differential geometry-based geometric learning of molecular datasets, Int. J. Numer.methods.Biomed. Eng. 35 (3) (2019), e3179.

[28] D.D. Nguyen, T. Xiao, M. Wang, G.-W. Wei, Rigidity strengthening: a mechanism for protein–ligand binding, J. Chem. Inf. Model. 57 (7) (2017) 1715–1721.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: an Imperative Style, High-Performance Deep Learning Library, 2019 arXiv preprint arXiv:1912.01703.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[31] S.J. Reddi, S. Kale, S. Kumar, On the Convergence of Adam and beyond, 2019 arXiv preprint arXiv:1904.09237.

[32] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Network. 61 (2015) 85–117.

[33] N. Spinu, M.T. Cronin, S.J. Enoch, J.C. Madden, A.P. Worth, Quantitative adverse outcome pathway (QAOP) models for toxicity prediction, Arch. Toxicol. 94 (2020) 1497–1510.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[35] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to Sequence Learning with Neural Networks, 2014 arXiv preprint arXiv:1409.3215.

[36] B. Wang, C. Wang, K. Wu, G.-W. Wei, Breaking the polar-nonpolar division in solvation free energy prediction, J. Comput. Chem. 39 (4) (2018) 217–233.

[37] E. Wang, H. Sun, J. Wang, Z. Wang, H. Liu, J.Z. Zhang, T. Hou, End-point binding free energy calculation with MM/PBSA and MM/BBSA: strategies and applications in drug design, Chem. Rev. 119 (16) (2019) 9478–9508.

[38] J. Wang, W. Wang, S. Huo, M. Lee, P.A. Kollman, Solvation model based on weighted solvent accessible surface area, J. Phys. Chem. B 105 (21) (2001) 5055–5067.

[39] R. Wang, D.D. Nguyen, G.-W. Wei, Persistent spectral graph, Int. J. Numer. methods.Biomed. Eng. 36 (9) (2020), e3376.

[40] K. Wu, G.-W. Wei, Quantitative toxicity prediction using topology based multitask deep neural networks, J. Chem. Inf. Model. 58 (2) (2018) 520–531.

[41] K. Xia, K. Opron, G.-W. Wei, Multiscale multiphysics and multidomain models—flexibility and rigidity, J. Chem. Phys. 139 (19) (2013) 11B614_1.

[42] H. Zhu, T.M. Martin, L. Ye, A. Sedykh, D.M. Young, A. Tropsha, Quantitative structure-activity relationship modeling of rat acute toxicity by oral exposure, Chem. Res. Toxicol. 22 (12) (2009) 1913–1921.

[43] H. Zhu, A. Tropsha, D. Fourches, A. Varnek, E. Papa, P. Gramatica, T. Oberg, P. Dao, A. Cherkasov, I.V. Tetko, Combinatorial QSAR modeling of chemical toxicants tested against tetrahymena pyriformis, J. Chem. Inf. Model. 48 (4) (2008) 766–784.