In too deep? Neural networks versus linear models for critical micelle concentration prediction

Alexander Moriarty,[*,†] Takeshi Kobayashi,[†] Matteo Salvalaglio,[†] Alberto Striolo,[†,‡] and Ian McRobbie[¶]

†*Department of Chemical Engineering, University College London, UK*

‡*Gallolgy College of Engineering, University of Oklahoma, USA*

¶*Senior Vice President, Research and Technology, Innospec Ltd., Ellesmere Port, UK*

E-mail: alexander.moriarty.21@ucl.ac.uk

Abstract

TODO: Write this.

Introduction

Perhaps the most well established predictor for critical micelle concentration (CMC), X_{cmc}, is the Stauff-Klevens relationship, first published in 1953.[1] It formalised the observation that CMC decreases exponentially with an increase in the number of carbons in the hydrocarbon tail, n_c:

$$\log X_{cmc} = A - Bn_c, \quad B > 0 \tag{1}$$

where A and B are empirical constants that depend on the temperature and the homologous series, i.e. the headgroup. The model is simple, yet accurate, and it is easily interpretable: to reduce CMC, extend the surfactant's hydrocarbon tail and this qualitative heuristic is easy to apply. Its drawback as a predictive model is its very limited applicability domain; each set of parameters is only applicable to surfactants with a specific headgroup and a linear carbon tail. One of the goals of quantitative structure-property relationship (QSPR) development is to produce models that are general, so that we can apply them to a diverse range of compounds, design novel molecules with target properties, and interpret the models' results to glean chemical insights.

To that end, there have been a wealth of investigations into making more general models for CMC prediction, which are discussed in Section . The two fundamental differences between them are the choice of molecular descriptors, which are the numerical features that form the basis set for the model inputs, and the functional form of the approximator that maps the descriptor to the property prediction.

Recently, an approach based on graph neural networks (GNNs) has produced highly accurate predictions, whilst having the advantage of being applicable to nonionic, cationic, anionic and zwitterionic surfactants simultaneously.[2] Neural networks have many trainable parameters and a complex functional form. This ensures their versatility as universal approximators, but makes them highly susceptible to overfitting.[3] By using such complex models, we also abandon the parsimony exhibited by Stauff-Klevens, and chemical insights can be much more difficult to derive. Furthermore, deep neural networks' ostensible 'universality' can be misleading: extrapolating the model's results to out-of-domain molecules (ones that are 'dissimilar' from the training data) will yield unreliable and potentially misleading predictions.

In this article, we develop two types of models of very different complexity: a linear model and a GNN. We evaluate the difference in performance and interpretability of the models. We also apply a technique for adding uncertainty quantification to the GNN, which

can indicate whether a molecule is within the model's applicability domain and therefore whether a given prediction is reliable.

Brief review of models for CMC prediction

Broadly, CMC predictive models take four forms: empirical, semi-empirical, theoretical and simulated, or some combination of these. Here we will focus on predictive models for aqueous solutions containing a single surfactant and discuss some of the trade-offs between the different approaches with regards to speed, universality and interpretability.

Theoretical approaches have the potential to be the most useful type of predictive model, if they are accurate and applicable to the desired system, as they are directly related to scientific knowledge and their results can be understood in terms of well studied principles. Puvvada and Blankschtein [4] derived a phenomenological model for studying aqueous nonionic surfactant systems that enabled CMC prediction, as well as modelling of other properties across a range of temperatures. The model they developed was the product of decomposing the process of micellisation into discrete steps that they could describe thermodynamically so as to yield an description of the free energy of micellisation in terms of a set of molecular parameters:

- The tail length, described as the number of carbon atoms.
- The average cross-sectional area of the headgroup, which controls the steric contribution to the free energy. This must be estimated.
- The Tolman length of the tail, which effectively describes the thickness of an 'interaction region' around the tail.[5] This must also be estimated.

A functional form to estimate the parameters was described for linear, nonionic, polyoxyethylene alcohol surfactants. The model attained impressive accuracy for some predictions: a root-mean-squared error (RMSE) of approximately $0.14\,\log\,\mu M$ for the group $C_{10}E_i$,

where $i \in [3, 6]$, and 0.21 $\log \mu M$ for the group $C_{12}E_j$, where $j \in [3, 8]$. However, for other systems, like C_8E_6, the error is much larger. The authors expect that this inaccuracy is because the model overestimates the CMC values for systems in which the micelles do not grow.

The connection that this model established between a small set of physically meaningful properties that can be estimated and emergent properties of surfactants is extremely useful, especially because it does not explicitly require fitting to any experimental data. However, the procedures described for estimating the Tolman length are only applicable for linear hydrocarbon chains; not the branched case, or for heterogeneous tail groups. Estimating the average cross-sectional area of the head group may also not be trivial.

Semi-empirical approaches are grounded in theory, but have parameters that are optimised based on experimental data. Many semi-empirical approaches to CMC prediction can be described as *segment-based* methods, whereby the surfactant is decomposed into discrete segments which correspond to groups of atoms and bonds.

Li et al.[6] applied a segment-based UNIQUAC model (s-UNIQUAC) and a SAFT equation of state to predict CMCs of linear polyoxyethylene alcohols, by first deriving expressions for the activity coefficient of a surfactant in water. In the s-UNIQUAC model, a segment-based local-composition model was used and the fugacity could then be approximated using the fitted interaction energies between the segments and water. In this case, the segments used were C_2H_4 and C_2H_4O. In the SAFT approach, the surfactant was treated as a chain of soft-sphere segments in order to first derive the Helmholtz energy of the solution and from that to derive the fugacity. In this case, the segments used were CH_2/CH_3 (these were treated as the same segment) and C_2H_4O. The interaction energies of the segments were fitted, as well as parameters of a function describing the soft sphere diameter of a segment in a chain in terms of the chain length.

Cheng and Chen[7] compared the performance of these models on a larger dataset, alongside three other models. Two of these were segment-based models: the polymer-NRTL

model[6] and a UNIFAC model,[8] both of which were cited as inspirations for the s-UNIQUAC model. The authors also employed their own modified Aranovich and Donohue (m-AD) model. The m-AD model calculates the CMC as a mole fraction, x_S^L, approximating it as the reciprocal of the limiting value of the surfactant's activity coefficient in aqueous solution, $\gamma_S^{L,\infty}$:

$$x_S^L = \frac{1}{\gamma_S^{L,\infty}} \tag{2}$$

The m-AD model considers the exchange equilibrium on a three-dimensional lattice of infinitely separated solvent and solute molecules in order to determine $\gamma_S^{L,\infty}$. Notably, the m-AD model is not a segment-based model. Instead, the authors fitted an interchange energy, Δ, separately for each molecule. Of course, if a new parameter must be fitted for every molecule, a model has no predictive ability. Therefore, correlations were examined between Δ and other, readily calculated surfactant properties, which will be discussed later.

Where data from literature was available, the predictive performance of the models on the molecular series C_nE_6, C_nE_8, C_nE_9, $C_{10}E_n$ and $C_{12}E_n$ were compared, and the resulting RMSEs are summarised in Table 1.

Table 1: Comparison of the RMSEs of selected models on polyoxyethylene alcohols. Data from Cheng and Chen[7].

Model	RMSE ($\log \mu M$)
p-NRTL	0.18
s-UNIQUAC	0.14
SAFT	**0.06**
UNIFAC	0.14
m-AD	0.11

TODO: Talk about geometric, topological and electronic descriptors, as well as simulation approaches.

Method

Two datasets were used for training and testing:

The Qin dataset is a dataset of 202 surfactants, curated by a previous work[2] by accumulating results from several publications. To the authors' knowledge, it is currently the largest public dataset of CMCs for several classes of surfactant collected at standard conditions, in an aqueous environment between 20 °C to 25 °C.

The NIST dataset is a dataset of 43 unique surfactants and their aqueous CMCs, extracted from the work of Mukerjee and Mysels[9]. For each surfactant, the mean of the experimental measurements between 20 °C to 25 °C and with no additives was used as the target CMC value. These data were used exclusively for testing, to assess whether sampling bias affects the evaluated performance.

The Qin data were split into training and test subsets, to simulate the real-world scenario of using a model to make inferences about molecules for which no data is available. The training data were used to fit the models, whilst test data were 'locked away' until it had been decided that the model was optimised, and the performance metrics on the test data were used for comparison. For some models, the training data were further split into optimisation and validation subsets; the optimisation data were used when calculating the loss function during model fitting and the validation data were used for on-the-fly evaluation of model performance during training.

To provide a consistent benchmark of model performance, the same data splits were used as Qin et al.[2]. The number of each class of surfactant in the train and test subsets of the data are shown in Table 2.

A QSPR pipeline requires choosing two essential functions: a representation function, whose parameters are defined before training the model, and a mathematical form that maps this representation to a prediction. The processes by which these functions are developed are called *feature engineering* and *model selection*, respectively.

Table 2: The number of each class of surfactant contained in the train/test subsets of the CMC datasets.

Data subset		Number of			
Dataset	Train/test	Nonionics	Anionics	Cationics	Zwitterionics
Qin-All	Train	110	30	31	9
	Test	12	4	4	2
Qin-Nonionics	Train	110			
	Test	12			
NIST	Train				
	Test	12	23	6	2

Feature engineering

The ideal molecular representation depends on the task at hand. Ideally, it should be compact, but complete;[10,11] 'as simple as possible, but not simpler.' To that end, the representation should contain enough information to distinguish between isomers that with distinct properties, though concessions can be made if we restrict the model's domain and self-impose limits on the type of isomers we expose the model to, both during training and in use. Representations may also include descriptions of state, such as temperature and pressure,[12] but this is redundant in cases where the training data spans a very limited range of states.

Extended-connectivity fingerprints

In this approach, the molecule is split into atomic environments up to a given radius, r: each environment is centred on an atom and extends r steps along connecting bonds. Effectively, we discard the categorical encoding in favour of introducing more continuous, count-based features, like n_c. The set of all environments in the training data up to radius r is extracted. The resulting feature vector, $\vec{c}$ for a molecule is described by

$$c_m = \text{Count}(\mathcal{E}_m), \tag{3}$$

where $\mathcal{E}_m$ represents the m^{th} atomic environment.

Now, a change in headgroup composition is reflected in a change in subgraph counts, and provided the new subgraph exists in our training data, the model can adjust its prediction accordingly. Branch points in a carbon chain are distinguished from main-chain groups, as they terminate in a CH group, rather than CH_2. This type of representation is called an *extended-connectivity fingerprint* (ECFP).[13]

ECFPs are similar to a segment-based approach; however, unlike segments or groups, subgraphs can overlap. As discussed above, a group contribution approach requires that a canonical 'priority' of the groups be defined prior to featurising molecules. By using ECFPs, the manual identification of important groups and their priorities is skipped; feature importance determination is instead delegated to the model.

However, these fingerprints do not necessarily distinguish between all positional isomers or chain isomers, particularly with smaller values of r, nor are stereoisomers treated differently. Another potential disadvantage is that the number of unique atomic environments is potentially very large relative to the size of the data available, which poses a risk of overfitting. Furthermore, larger environments necessarily envelop smaller ones, which means that there is some duplicate information in the representation: the presence of a $(CH_2)_3$ environment implies the presence of three CH_2 environments, so that there is multicollinearity. This redundancy can impede model fitting and interpretation.

Molecular graph representation

Both of the approaches described so far rely on molecular feature vectors that cannot describe the molecule's topology. A molecular graph is a structure that achieves this, and it is a popular choice for cheminformatics as well as visualisation of molecular structure. In this approach, each atom is considered a *node* and each bond an *edge*. Rather than having a single feature vector to describe the molecule as a whole, each atom is assigned its own feature vector, $\vec{v}_i$, based on properties such as its element, hybridisation state, charge, etc. These feature vectors are concatenated into a node feature matrix, $\mathbf{V}$. The graph's structure

is then defined by a binary adjacency matrix, $\mathbf{A}$:

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } i \text{ bonded to } j, \text{ or } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Molecular graphs are perhaps the most natural representations to visualise; see Figure 1. This is an exact description of the molecule's topology that enables an atomistic machine learning approach.

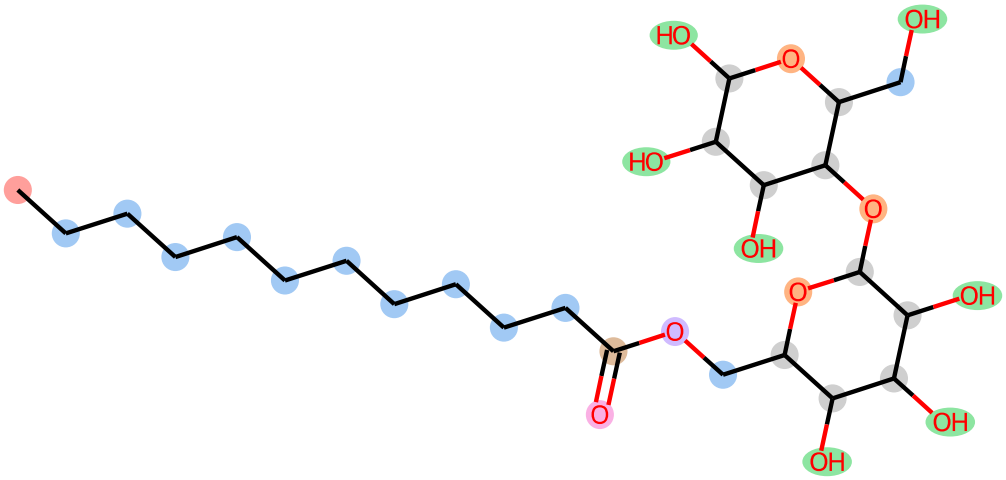

Figure 1: A molecular graph of 6-*O*-dodecanoyl-maltose. Atoms are highlighted based on their feature vectors, $\vec{v}_i$, so that equal feature vectors have the same colour.

Model selection

ECFP model

Based on the prior knowledge encoded in Equation 1, it is reasonable to assume that certain atomic environments have a linear relationship to $\log X_{cmc}$. It therefore seems justified to apply a linear model to the ECFP fingerprints described in Equation 3:

$$\log X_{cmc} = \vec{w} \cdot \vec{c} + b, \tag{5}$$

where $\vec{w}$ is a trained weights vector, the elements of which correspond to the contribution of an atomic environment to the CMC, and b is an intercept (or *bias* term).

However, the issues of the large feature vector size and multicollinearity must be addressed; a naïve fit using ordinary least squares (OLS) would likely produce poor results. To that end, a process of *feature selection* was applied, whereby a subset of the atomic environments were selected for use in the model. There are several approaches to feature selection;[14] here, we chose an approach based on *regularisation*.

In this approach, we include a term in the loss function that depends on the norm of $\vec{w}$. The two types of constraints considered in this paper are ℓ_1 and ℓ_2 regularisation, which correspond to the inclusion of ℓ_1 and ℓ_2 norms, respectively. By combining ℓ_1 regularisation with the least squares regressor, we obtain the least absolute shrinkage and selection operator (LASSO):[15]

$$\min_{\vec{w}} \frac{1}{2n_{\text{samples}}} \left\| \mathbf{C}\vec{w} + \vec{b} - \vec{y} \right\|_2^2 + \alpha \left\| \vec{w} \right\|_1, \tag{6}$$

where n_{samples} is the number of training samples; $\vec{y}$ are the training data's true values of $\log X_{cmc}$; $\vec{b}$ is a vector with elements all equal to b; and α is a user-defined hyperparameter describing the degree of regularisation ($\alpha \geq 0$). $\mathbf{C}$ are the standardised training data feature vectors, $\{\vec{c}'_n \,|\, 1 \leq n \leq n_{\text{samples}}\}$, stacked row-wise into a matrix. Standardising the environment counts ensures that they have zero mean and unit variance:

$$c'_m = \frac{c_m - u_m}{s_m}, \tag{7}$$

where u_m and s_m are the mean and standard deviation of the number of $\mathcal{E}_m$ in each molecule in the training data. This standardisation is necessary to ensure that the regularisation term is not dominated by environments with high variance and that it accounts for common and uncommon environments alike.

By imposing the ℓ_1 penalty, the model is biased towards learning a *sparse* weight vector:

many of its elements will be negligible. The corresponding features can be removed from the representation.

However, LASSO has two major flaws that make it inappropriate for the task of ECFP regression:

- The number of unique atomic environments is greater than the size of the training data, n_{samples}, but LASSO will select at most n_{samples} features.[16] Therefore, some important environments may still be excluded.

- LASSO tends to select only one of a group of highly correlated variables, when there is no reason why that particular one should be prioritised.[17] This is undesirable because we want to include both large and small atomic environments, despite their large correlation, and we might be misled into thinking that some highly correlated environments have no effect on CMC.

ElasticNet addresses these issues by imposing an additional ℓ_2 penalty:[17]

$$\min_{\vec{w}} \frac{1}{2n_{\text{samples}}} \left\| \mathbf{C}\vec{w} + \vec{b} - \vec{y} \right\|_2^2 + \alpha\rho \left\| \vec{w} \right\|_1 + \frac{\alpha(1-\rho)}{2} \left\| \vec{w} \right\|_2^2, \tag{8}$$

where ρ is a user-defined hyperparameter controlling the proportions of the regularisation terms, $0 < \rho < 1$. This removes the hard limit on the number of features that can be selected and also exhibits the 'grouping effect', whereby features with high correlation tend to be assigned similar weights.

Because of these advantages, an ElasticNet linear regression model was applied to select the most important ECFP features for predicting $\log X_{cmc}$. Both hyperparameters, α and ρ, must be defined when training the model. In order to select the best values, k-fold cross-validation was employed: the training data were partitioned into k subsets of roughly equal size. k models were trained using $k-1$ subsets, and the final subset was used for validation. The average mean-squared error of the k models, evaluated on their respective validation subsets, is the model's final score. This routine, with $k = 5$, was applied for a range of α and

ρ combinations, and the lowest scoring combination was used. The hyperparameter search space is defined in the Supplementary Information.

The features with non-neglible fitted weights from ElasticNet were then selected for use in the final linear model. This model, ridge regression, uses just ℓ_2 regularisation, so that all of the weights are non-negligible but we still address the issue of multicollinearity:[18]

$$\min_{\vec{w}} \left\| \mathbf{C}\vec{w} + \vec{b} - \vec{y} \right\|_2^2 + \alpha \left\| \vec{w} \right\|_2^2 . \tag{9}$$

A similar cross-validation method to the one described above was used to determine the best α parameter, but using $k = n_{\text{samples}} - 1$. Because only one hyperparameter needs to be determined, there are far fewer trials per fold and therefore a greater number of folds can be used.

It was empirically observed that the combination of ElasticNet feature selection and a final regression with the simpler ridge regression model yielded better results, likely due to using a larger number of folds when determining the best value for α. Both models were implemented using scikit-learn.[19]

Molecular graph model

The basic topology of the graph neural network (GNN) used in this work was identical to the one used by Qin et al.[2]. That is, the first step of the model consists of a stack of graph network layers, which mutate the node features in a molecular graph based on those of bonded atoms. These layers employ the graph convolution network (GCN) architecture introduced by.[20] Layer l computes a new node feature matrix, $\mathbf{V}^{(l)}$, based on the adjacency matrix, $\mathbf{A}$:

$$\mathbf{V}^{(l)} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{V}^{(l-1)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}. \tag{10}$$

Here, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weights and biases, respectively, of layer l and we have also introduced the degree matrix,

$$D_{ii} = \sum_j A_{ij}, \tag{11}$$

so that the term $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ effectively normalises the adjacency matrix based on the degree of each atom.

$\mathbf{V}^{(1)}$ therefore encodes not only information about the atom itself, but its bonded neighbours. This information is used in the subsequent graph convolution, so that $\mathbf{V}^{(2)}$ encodes information about the 2$^{\text{nd}}$ order neighbourhood, *et cetera*. The number of graph layers, L, therefore dictates the 'radius' around each atom that is considered in computing the final feature vector, analogous to creating an ECFP, except that the i^{th} atomic environment is characterised by a continuous, *latent* vector, $\vec{v}_i^{(L)}$.

The next step is a pooling layer, which converts the graph to a single *latent representation vector*, $\vec{v}^{(p)}$, losing the explicit topological information. Several choices of pooling function were trialled:

Mean pooling was used in the previous work. It computes the average over all atoms' feature vectors.

Sum pooling computes the sum of all atoms' feature vectors. This is the most analogous to the ECFPs in that the contribution of an atomic environment scales linearly with the number of times it occurs in the molecule.

Gated attention pooling applies an *attention* mechanism to decide which environments are relevant to the prediction:[21]

$$\vec{v}^{(p)} = \sum_i^N \sigma\left(\mathbf{W}_1\vec{v}_i^{(L)} + \vec{b}_1\right) \odot \left(\mathbf{W}_2\vec{v}_i^{(L)} + \vec{b}_2\right), \tag{12}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are trained weights and $\vec{b}_1$ and $\vec{b}_2$ are biases, σ is the sigmoid activation function, N is the number of atoms in the molecule, and $\odot$ represents

element-wise multiplication.

Attention sum pooling is a simpler variation of the above. By using a softmax function, it performs a weighted average of the atomic environments' contributions:

$$\mathbf{X} = \text{softmax}(\mathbf{V}^{(L)}\vec{w}), \tag{13}$$

$$\vec{v}^{(p)} = \sum_{i}^{N} \mathbf{X}_i \cdot \vec{v}_i^{(L)}, \tag{14}$$

where $\vec{w}$ are trained weights.

After training the model, $\vec{v}^{(p)}$ effectively acts as a machine-learned representation of the molecule that captures only the information about its topology and composition that is useful for predicting the CMC. This task of finding an optimised representation is a feature of neural networks that happens implicitly during training, called *representation learning*.[22] The final step is a readout neural network: a multi-layer perceptron which acts as a nonlinear approximator to map this latent representation vector to the CMC property prediction. Each layer in this neural network, called a 'dense' layer, outputs a new vector, $\vec{v}^{(l)}$:

$$\vec{v}^{(l)} = \mathbf{W}^{(l)}\vec{v}^{(l-1)} + \vec{b}^{(l)}, \tag{15}$$

The full network's architecture is illustrated in Figure 2. The model was implemented using the open-source library Spektral.[23]

A neural network's topology describes the types of layers used, i.e. their functional form, and the connection between them. Layers that are parameterised by a weight matrix, $\mathbf{W}$, may have different 'sizes', meaning that the dimensionality of their output is arbitrary and can be adjusted by changing the dimensions of $\mathbf{W}$. The graph layers, dense layers and the gated attention pool all have this property. These sizes, the type of pooling layer and the number of each graph and dense layers, are all hyperparameters that can be adjusted prior to training. To determine the best combination of hyperparameters for predicting CMCs,

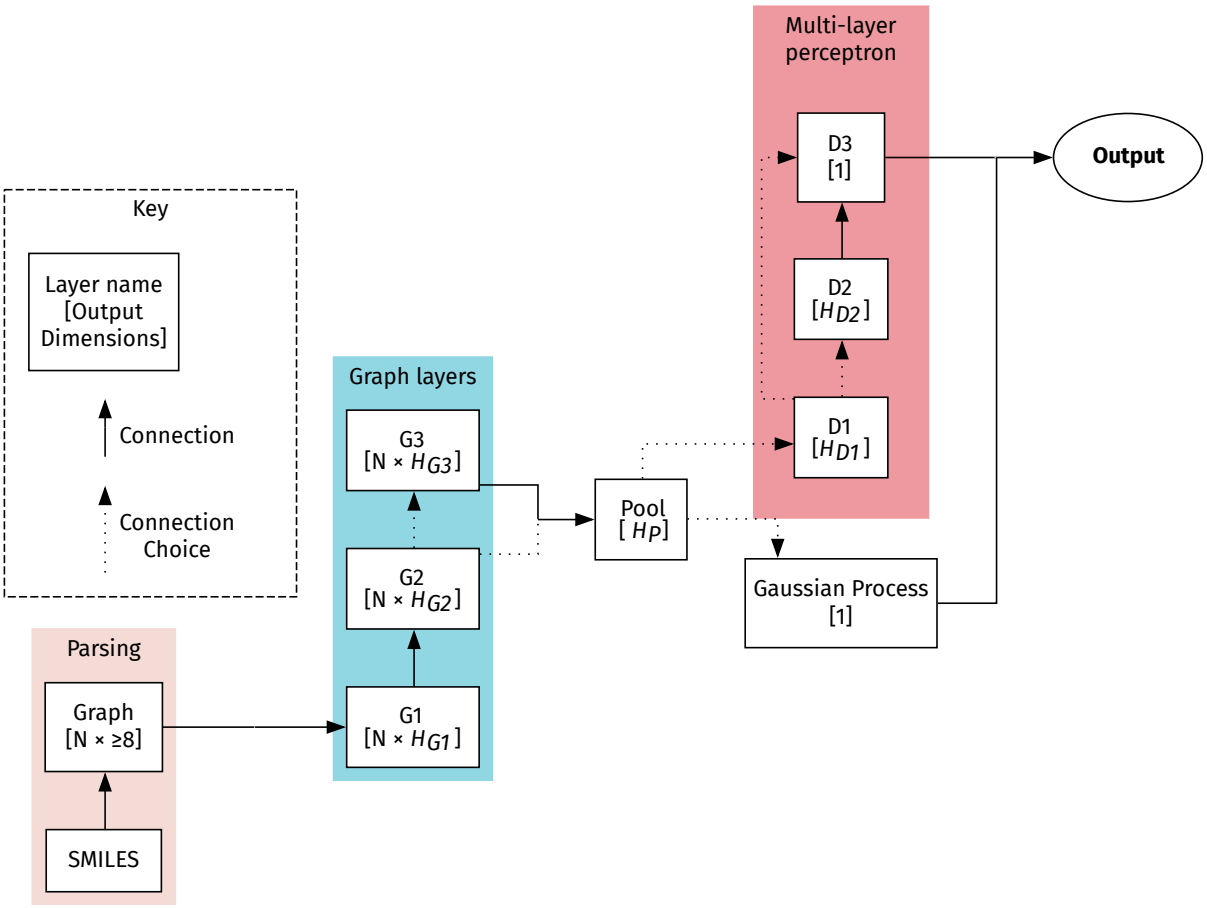

Figure 2: Schematic of the neural network architecture. Here, N represents the number of constituent atoms/ions in the input molecule and H represents a hyperparameter.

an automated searching procedure was employed.

Optimising GNN hyperparameters

The Hyperband approach,[24] implemented in Keras Tuner,[25] was used to to select a good combination of hyperparameters for the model. Hyperband provides a way to efficiently evaluate the performance of a large search space of hyperparameter configurations. The algorithm trials several combinations of hyperparameters, initially allocating only a small number of resources to each trial. The hyperparameters for the trials with the best performance are then allocated more resources, whilst the remainder are discarded. A reduction factor of 3 was chosen, meaning that 2/3 of the trials were discarded after each iteration. This procedure iterates until the best configuration is found.

The algorithm can be executed multiple times if resources are available, to obtain a more reliable result; the training procedure is stochastic and therefore the performance of two trials with the same hyperparameters may be different. In this case, a single run was performed. The training data was partitioned into an optimisation subset and a validation subset in a ratio of 9:1. The trials were fit to the optimisation subset and evaluated based on the RMSE of their predictions on the validation subset.

Adding uncertainty with a Gaussian process

To improve the model's reliability, a *surrogate* model was employed that could yield uncertainty estimates alongside CMC predictions. The approach is based on the Convolution-Fed Gaussian Process of Tran et al.[26]. The model first computes the latent representation vector, $\vec{v}^{(p)}$, of an input molecule using a trained GNN. $\vec{v}^{(p)}$ is then standardised, similar to Equation 7:

$$v_n^{(p)\prime} = \frac{v_n^{(p)} - u_n}{s_n}, \tag{16}$$

but in this case the standardisation applies across each latent feature, n. Again, u_n and s_n were determined from the training molecules' latent representations.

The standardised latent representation vectors of the training data serve as index points for a Gaussian process (GP); see Figure 2. The GP's predicted mean and standard deviation define a predicted normal distribution of a molecule's CMC, $\log X_{cmc} \sim \mathcal{N}(\mu, \sigma)$.

In this work, the GPs were defined using a Matérn kernel with parameter 1/2. Furthermore, the multi-layer perceptron component of the GNN used to calculate $\vec{v}^{(p)}$ was employed as the GP's mean function. The kernel parameters were optimised with an Adam optimiser.[27] The same optimisation/validation splitting was used as for the GNN hyperparameter search and training was stopped after 1000 iterations without improvement in the validation predictions' RMSE. This implementation was based on GPFlow.[28]

Results

ECFP models

GNN models

Combined GNN and GP models

Performance on NIST data

The layout of this should be as follows:

- Describe ECFP results: the number of groups after parsing, the number found during optimisation, include best params in supplementary information.
- The performance of ECFP on training data.
- Describe hyperband results: the number of trials, the best validation error and the resulting topology.
- Describe performance of GNNs on test data.
- Describe performance of UQ on test data. Show parity plot of its predictions.

- Describe performance of all models on NIST data.
- Show parity plot of all models on NIST data.

Conclusion

Supporting Information Available

Source code for featurisation and model training, graph neural network logs and metrics for hyperparameter optimisation and final training, and individual model predictions.

References

(1) Klevens, H. B. Structure and Aggregation in Dilate Solution of Surface Active Agents. *Journal of the American Oil Chemists Society* **1953**, *30*, 74–80.

(2) Qin, S.; Jin, T.; Van Lehn, R. C.; Zavala, V. M. Predicting Critical Micelle Concentrations for Surfactants Using Graph Convolutional Neural Networks. *The Journal of Physical Chemistry B* **2021**, *125*, 10610–10620.

(3) Bejani, M. M.; Ghatee, M. A Systematic Review on Overfitting Control in Shallow and Deep Neural Networks. *Artificial Intelligence Review* **2021**, *54*, 6391–6438.

(4) Puvvada, S.; Blankschtein, D. Molecular-thermodynamic Approach to Predict Micellization, Phase Behavior and Phase Separation of Micellar Solutions. I. Application to Nonionic Surfactants. *The Journal of Chemical Physics* **1990**, *92*, 3710–3724.

(5) de Miguel, R.; Rubí, J. M. Gibbs Thermodynamics and Surface Properties at the Nanoscale. *The Journal of Chemical Physics* **2021**, *155*, 221101.

(6) Li, X.-S.; Lu, J.-F.; Li, Y.-G.; Liu, J.-C. Studies on UNIQUAC and SAFT Equations for Nonionic Surfactant Solutions. *Fluid Phase Equilibria* **1998**, *153*, 215–229.

(7) Cheng, J.-S.; Chen, Y.-P. Correlation of the Critical Micelle Concentration for Aqueous Solutions of Nonionic Surfactants. *Fluid Phase Equilibria* **2005**, *232*, 37–43.

(8) Voutsas, E. C.; Flores, M. V.; Spiliotis, N.; Bell, G.; Halling, P. J.; Tassios, D. P. Prediction of Critical Micelle Concentrations of Nonionic Surfactants in Aqueous and Nonaqueous Solvents with UNIFAC. *Industrial & Engineering Chemistry Research* **2001**, *40*, 2362–2366.

(9) Mukerjee, P.; Mysels, K. J. *Critical Micelle Concentrations of Aqueous Surfactant Systems*; 1971; pp 51–65.

(10) Faber, F.; Lindmaa, A.; von Lilienfeld, O. A.; Armiento, R. Crystal Structure Representations for Machine Learning Models of Formation Energies. *International Journal of Quantum Chemistry* **2015**, *115*, 1094–1101.

(11) Himanen, L.; Jäger, M. O. J.; Morooka, E. V.; Federici Canova, F.; Ranawat, Y. S.; Gao, D. Z.; Rinke, P.; Foster, A. S. DScribe: Library of Descriptors for Machine Learning in Materials Science. *Computer Physics Communications* **2020**, *247*, 106949.

(12) Chen, C.; Ye, W.; Zuo, Y.; Zheng, C.; Ong, S. P. Graph Networks as a Universal Machine Learning Framework for Molecules and Crystals. *Chemistry of Materials* **2019**, *31*, 3564–3572.

(13) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *Journal of Chemical Information and Modeling* **2010**, *50*, 742–754.

(14) Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R. P.; Tang, J.; Liu, H. Feature Selection: A Data Perspective. *ACM Computing Surveys* **2017**, *50*, 94:1–94:45.

(15) Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* **1996**, *58*, 267–288.

(16) Efron, B.; Hastie, T.; Johnstone, I.; Tibshirani, R. Least Angle Regression. *The Annals of Statistics* **2004**, *32*, 407–499.

(17) Zou, H.; Hastie, T. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **2005**, *67*, 301–320.

(18) McDonald, G. C. Ridge Regression. *WIREs Computational Statistics* **2009**, *1*, 93–100.

(19) Pedregosa, F. et al. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825–2830.

(20) Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. 2017.

(21) Li, Y.; Tarlow, D.; Brockschmidt, M.; Zemel, R. Gated Graph Sequence Neural Networks. 2017.

(22) Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press, 2016; pp 524–554.

(23) Grattarola, D.; Alippi, C. Graph Neural Networks in TensorFlow and Keras with Spektral. 2020.

(24) Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; Talwalkar, A. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research* **2018**, *18*, 1–52.

(25) Chollet, F., et al. Keras. 2015.

(26) Tran, K.; Neiswanger, W.; Yoon, J.; Zhang, Q.; Xing, E.; Ulissi, Z. W. Methods for Comparing Uncertainty Quantifications for Material Property Predictions. *Machine Learning: Science and Technology* **2020**, *1*, 025006.

(27) Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. 2017.

(28) Matthews, A. G. d. G.; van der Wilk, M.; Nickson, T.; Fujii, K.; Boukouvalas, A.; León-Villagrá, P.; Ghahramani, Z.; Hensman, J. GPflow: A Gaussian Process Library Using TensorFlow. *Journal of Machine Learning Research* **2017**, *18*, 1–6.

TOC Graphic

I'm looking for ideas for the TOC graphical entry – common practice seems to be to draw a really abstract picture of a molecule going through a network, but that doesn't seem like the best thing to do here because I'm specifically comparing against other models.