Uncertainty Estimation in Deep Learning-Based Property Models: Graph Neural Networks Applied to The Critical Properties

Aouichaoui, Adem R. N.; Mansouri, Seyed Soheil; Abildskov, Jens; Sin, Gürkan

Published in:
AIChE Journal

Link to article, DOI:
10.1002/aic.17696

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Uncertainty estimation in deep learning-based property models: Graph neural networks applied to the critical properties

Adem R. N. Aouichaoui | Seyed Soheil Mansouri | Jens Abildskov | Gürkan Sin

Department of Chemical and Biochemical Engineering, Technical University of Denmark, Lyngby, Denmark

Correspondence
Gürkan Sin, Department of Chemical and Biochemical Engineering, Technical University of Denmark, Søltofts Plads, Building 228A, Kgs. Lyngby DK-2800, Denmark.
Email: gsi@kt.dtu.dk

Abstract
Deep learning and graph-based models have gained popularity in various life science applications such as property modeling, achieving state-of-the-art performance. However, the quantification of prediction uncertainty in these models is less studied and is not applied in the low dataset size regime, which characterizes many chemical engineering-related molecular properties. In this work, we apply two graph-based models to model the critical-temperature, pressure, and volume and apply three techniques (the bootstrap, the ensemble, and the dropout) to quantify the prediction uncertainty. The overall model confidence is evaluated using the coverage. The results suggest that graph-based models perform better compared with current group-contribution-based property modeling techniques while eliminating the tedious task of developing molecular descriptors.

KEYWORDS
deep learning, graph neural networks, molecular property prediction, QSPR, uncertainty analysis

1 | INTRODUCTION

The thermodynamic, physical, and transport properties of chemical compounds play an integral role in performing many chemical engineering applications and are fundamental in developing computer-aided process engineering (CAPE) tools such as process simulators and product design frameworks. These properties are the product of experimental measurements or derived from such measurements. Critical point properties such as the critical temperature are used in equations of states (EOS) to perform P–V–T calculations. Flammability properties such as the flashpoint are used to determine safe processing and storage conditions of chemicals. Enthalpic properties such as the Gibbs energies of formation can be used to determine reaction equilibriums. Despite their importance, data availability remains an issue, and measuring these properties on the fly whenever the need arises is not a viable solution due to time constraints and equipment cost. Meanwhile, centralized large collections of property data are few, expensive, and in general, limited to the most commonly used compounds. This creates a need for predictive models capable of generating accurate and reliable property predictions in order to explore the chemical design space beyond what is currently available. Quantitative structure-property relations (QSPRs) are one class of property models that could potentially provide such functionality. These models are based on the assumption that a given property is directly related to the structure of the molecule and their development consists of two elements: descriptor selection (feature
The choice of descriptor and model structure is instrumental for a successful QSPR as well as any property modeling studies including neural networks. This choice is usually based on some particular domain knowledge or trial and error. Feature engineering aims to produce an expressive machine-readable translation of the molecular structure also known as a molecular descriptor or fingerprint. The model selection or development consists of choosing a suitable mathematical formulation capable of relating the molecular structural information to the target property of interest. Many approaches have been reported to handcraft and construct the molecular descriptors such as the use of electronic and topological descriptors or the use of a sparse bit vector indicating the presence of molecular substructures such as the extended connectivity fingerprints (ECFPs) or the Morgan circular fingerprints commonly used in the field of cheminformatics. A more familiar approach to QSPR in chemical engineering is the group-contribution (GC) property model, where the molecule is described in terms of substructures of functional groups and where the property value is determined through a summation of the individual contribution of each group. These models have been used to predict various properties such as the normal boiling point and critical point properties. These models have also been applied to address computer-aided molecular design problems such as screening for entrainers in extractive distillation and refrigerants. One disadvantage of these models is their inability to extrapolate outside their domain of applicability since the models’ ability to cover the chemical design space is highly dependent on the training sample. This results in models with a narrow range of applicability that do not generalize well. Additionally, handcrafted molecular descriptors can be cumbersome to produce especially if done manually.

Catalyzed by recent advancements in the field of deep learning, a new modeling approach has emerged where the two elements of the QSPR modeling are combined in a single end-to-end learning approach. Inspired by the application of convolutions for image analysis, graph convolutional neural networks (GCNNs or GNN) also referred to as message passing neural networks (MPNNs) have emerged. These models operate directly on a graph representation made of nodes (vertices) connected with edges, making them suitable to be applied on a graph abstraction of the molecules where the nodes correspond to the atoms and edges correspond to the chemical bonds. The nodes and edges are attributed with a feature vector with information such as the type of atom, the type of bonds, and so on. GNNs or MPNNs are capable of producing a representation of the molecule that accounts for the atoms’ local environment by using these features and stacking convolution layers and updating the feature vector representing the atoms by aggregating the information from neighboring atoms. The graph representation is converted into a vector representation through a readout or pooling operation, which is then used as input to a multi-layer perceptron (MLP) to map the molecular representation to the target property (see Figure 1). By combining the feature generation task and the regression task, the model is capable of adapting and adjusting the representation so it fits the task at hand through backpropagation of the error, eliminating the need for manually creating new task-specific descriptors.

These models have shown promising results in various tasks and even achieved the state of the art performance beating QSPRs with handcrafted descriptors across a wide range of properties related to quantum chemistry and drug discovery. These applications are characterized by large sizes of publically available datasets such as the QM9 dataset, which includes many quantum mechanic properties such as the atomization energy and the enthalpy of atomization. Since the introduction of the first GNN, many improvements and variations have been proposed with each achieving increasingly better results. These variations mainly reside in the feature extraction process, where each model proposes a different scheme for generating the molecular descriptor. Coley et al. developed a GNN that included atom-level
features calculated over the entire molecule (e.g., the total polar surface area contribution) as part of the initial node feature and applied it for regression and classification tasks including the octanol solubility and aqueous solubility.\textsuperscript{20} Yang et al. proposed a model that is centered on edges instead of nodes and applied it on a total of 19 property datasets related to biophysics and quantum mechanics.\textsuperscript{18} Schweidtmann et al. included higher-order local environment information and successfully used it to predict properties related to fuel ignition quality such as the cetane number, beating current GC models used for the same purpose.\textsuperscript{9} Recently, Xiong et al. proposed the addition of an attention mechanism to the classical message passing neural networks, which enables the model to assign importance weights to the input information and focus on the most relevant parts of the graph.\textsuperscript{23}

The vast majority of published models only aim to predict the endpoints providing no insight into the reliability of such predictions, despite previous studies cementing the importance of providing the uncertainty in the model prediction, especially for molecular properties used as input to thermodynamic models or used to calculate derived properties.\textsuperscript{13} Frügler et al. showed the impact of the uncertainty of fluid properties in screening close to 2000 fluids for organic Rankine cycle, revealing that the performance ranking differs significantly due to the uncertainties.\textsuperscript{1} The study was expanded to other EOS to investigate the sensitivity of these models with regard to the input properties by propagating the uncertainties into the model output using the bootstrap method.\textsuperscript{24} Hukkerikar et al. evaluated the uncertainty in prediction for 18 properties through the parameter covariance matrix and performed linear propagation of the error.\textsuperscript{5} Meanwhile, the bootstrap method was used to assess the uncertainty in predicting the enthalpy of combustion\textsuperscript{25} as well as other properties in conjunction with an artificial neural network.\textsuperscript{26} A recent study also benchmarked various uncertainty-aware GC models with for example, Gaussian process and deep neural networks for property prediction purposes.\textsuperscript{14}

While methods for uncertainty estimation have been established and widely applied for classical QSAR models such as GC models, the field is still considered in infancy for deep learning-based models and especially GNN models.\textsuperscript{27} This could potentially hinder the broader use and acceptance of these models, especially in applications involving engineering decision-making, and considering that the performance of data-driven models is highly dependent on the size, quality, and diversity of the training data.\textsuperscript{27,28} Therefore, models developed in data-deficient applications can cause highly uncertain predictions, leading to vulnerable decision making, further emphasizing the importance of uncertainty estimation.\textsuperscript{28,29} Driven by this, a few methods have been proposed recently to quantify the uncertainty in such models that are either based on frequentist statistics or Bayesian inference.\textsuperscript{30} Frequentist methods are typically based on ensemble learning (deep ensemble).\textsuperscript{30} The method consists of using a collection of “M” distinct and independently trained models rather than a single one.\textsuperscript{31} The relative output frequency of the ensemble corresponds to the output probability.\textsuperscript{30} The variance in the models is introduced by training each predictor on a randomly sampled subset of the overall training data (also known as bagging) or through random initialization of the model weights.\textsuperscript{30} A potentially less computationally expensive way to produce ensembles is the use of snapshot-ensemble, where the optimization progress is saved at different stages by inducing divergence through a cyclic learning rate.\textsuperscript{32} Bayesian methods have also gained attention lately due to the increase and availability of computational power as well as the development of various methods to approximate Bayesian inference.\textsuperscript{30} Bayesian neural networks replace the point estimate of the weights and biases with probability distributions.\textsuperscript{27,30} The posterior distribution is then approximated by Monte-Carlo (MC) sampling from the distribution of the parameters.\textsuperscript{27,30} However, due to the computational expenses entailed in approximating the posterior distribution, less demanding methods have been developed.\textsuperscript{28,33,34} Dropout was originally intended as a regularization method to reduce overfitting and avoid weight co-adaptation by assigning a probability \( \rho \) that a given node in a layer becomes zero following a Bernoulli distribution.\textsuperscript{35} Recent studies have shown that dropout can be interpreted as a Bayesian approximation of the Gaussian Process and by applying this between each layer of the network and performing MC sampling after training, a distribution of the output can be obtained.\textsuperscript{31} This process is called MC dropout and introduces an additional hyperparameter, the dropout rate \( \rho \) for every layer. Concrete dropout was developed in order to make the dropout rate a “learnable” parameter.\textsuperscript{34} Recent effort have been directed to apply these methods to deep learning-based property models such as the GNN. Ryu et al. proposed Bayesian GNN for screening drug candidates where the posterior distribution is approximated with variational inference with a product of Bernoulli distributions using MC dropout.\textsuperscript{28} Noh et al. investigated the uncertainty of a crystal graph neural network applied to predict the energy of formations using MC dropout.\textsuperscript{28} By including the uncertainty information, they achieved a better success ratio (true positive) and discoverability (ratio of identified candidates conforming to DFT calculations). Scalia et al. performed an extensive study on the scalability of some selected uncertainty estimation methods for the GNN model developed by Yang et al.\textsuperscript{18,27} The uncertainty was quantified using: concrete dropout, ensembling, and bootstrapping from the training data (bagging).\textsuperscript{27} Through various case studies, ensembling and bootstrapping outperformed dropout for in-domain and out-of-domain application.\textsuperscript{27} However, common for all methods is that they underestimate the uncertainty for out-of-domain data due to the lack of calibration.\textsuperscript{27} Hirschfeld et al. considered ensemble, bagging, snapshot ensemble, and MC-dropout.\textsuperscript{37} The results did not reveal a particular method to be more reliable across the dataset.\textsuperscript{37} In this work, the approaches presented differ from previous work in the property dataset used for modeling, the specific bootstrap method, and the use of coverage to illustrate the overall reliability of the model prediction.

## 2 | METHODS AND TOOLS

In this study, we investigate two distinct GNN models and benchmark three uncertainty estimation methods on three datasets. All of which are introduced in the following.
2.1 | Graph neural networks

The GNN models chosen are the message passing neural network developed by Gilmer et al.\textsuperscript{17} and the attentive fingerprint GNN developed by Xiong et al.\textsuperscript{23}

2.1.1 | Message passing neural network (MPNN) by Gilmer et al.\textsuperscript{17}

The MPNN is a graph neural network that operates on an undirected graph $G$ with node features $x_v$ and edge features $e_{vw}$. The operation of the MPNN comprises three phases: the message passing phase, the update phase, and the readout phase.\textsuperscript{17} In the message passing phase, the features of the neighboring atoms are combined through a scheme denoted by the message function. The edge features are incorporated in the message function in order to also include bond information in the update of the atom feature vector. The resulting message is then used to update the atom feature vector. By repeating this process, the atom feature vector is influenced by atoms further away potentially capturing proximity effects. The readout phase consists of transforming the graph representation into a vector representation through a pooling function. The pooling function used is the set2set model developed by Vinyals et al.\textsuperscript{38} which produces a graph level embedding or molecular descriptor that can be fed through an MLP to predict the target property. In addition to the hyperparameters for the MLP, the tunable parameters consist of the size of the output representation of the GNN, the size of the hidden edge feature, the number of message passes, the number of layers, and steps in the set2set process. More details can be found in the work by Gilmer et al.\textsuperscript{17} and the Supporting Information.

2.1.2 | Attentive FP (AFP) by Xiong et al.\textsuperscript{23}

The model by Xiong et al.\textsuperscript{23} combines the concept of graph neural network with the attention mechanism\textsuperscript{39} in order to enable the model to focus on specific parts of the molecular graph by assigning different importance weight to the neighbors. Meanwhile, the model by Gilmer et al., assumes that all neighbors influence the hidden state of the node equally.\textsuperscript{17} The attention mechanism consists of three operations: alignment, weighting, and context. First, the node feature is concatenated with the features of the neighbor node and thus each node-pairs state is aligned. The weighting is done through the softmax function to obtain the importance weight of the neighbor node to the given node.\textsuperscript{23} The context operation consists of linearly transforming the neighboring node states and subjecting the output to a weighted sum and nonlinear transformation.\textsuperscript{23} In the view of the MPNN, the output of the context operation is the message function and the update function is done by updating the node state through a gated-recurrent-unit using the previous feature and the output of the context process. An additional graph level embedding is conducted at the end of the atom embedding using the same attention mechanism in order to produce the graph presentation by following the same attention mechanism. Apart from the MPNN, the AFP model has the following hyperparameters: the number of GNN layers, the number of timesteps, and the size of the output representation to be supplied to the MLP. More details can be found in the work by Xiong et al.\textsuperscript{23} and the Supporting Information.

2.2 | Molecular featurization

The initial graph representation and features are generated using the open-source cheminformatics package RDKit\textsuperscript{40} using the SMILES.\textsuperscript{41} The nodes and edges are embedded with various information about the atoms and bonds respectively as seen in Tables 1 and 2. The features used in this study are selected according to the previous studies.\textsuperscript{9,18–20,23} In total, 28 node/atom features and 12 edge/bond features were used for molecular graph featurization.

2.3 | Data

Three datasets were chosen from the AIChE DIPPR 801 Database\textsuperscript{42} to benchmark the models and the uncertainty quantification methods.
The selected properties are the critical temperature \((T_c)\), the critical pressure \((P_c)\), and the critical volume \((V_c)\). The properties were chosen due to their role in phase behavior and performing \(P–V–T\) calculations.

Various summary statistics can be seen in the Supporting Information. The distribution of the data can be seen in Figure 2. In this study, we only considered organic compounds with the following elements \((O)\), \((N)\), \((F)\), \((Br)\), \((S)\), and \((I)\).

### 2.4 Uncertainty estimation methods

#### 2.4.1 Bootstrap

The bootstrap method aims to determine the distributions of the measurement errors from the actual data.\(^{25,43}\) The errors are then propagated through the model to produce different realizations of the model parameter, relaxing the assumption of a Gaussian, and identical independent distributed measurement errors.\(^{43}\) By fitting an initial reference model, “\(N\)” bootstrap samples are created by combining the reference model output with the errors sampled with replacement (errors uniformly distributed with equal probability of realization).\(^{43}\) By repeating the regression procedure for each synthetic dataset, a new set of the model parameters is realized, reflecting the impact of measurement errors on the uncertainty in the parameters.\(^{25}\) The output of all regressed models constructs the output distribution using \(N = 400\) bootstrap samples.

#### 2.4.2 Ensemble

The ensemble method was originally proposed as a technique to prevent overfitting and improve the model accuracy.\(^{44}\) The method consists of “ensembling” or combining a series of independently trained homogeneous “weak” learners of size “\(M\)” and combining their output through for example, model averaging to produce a combined prediction.\(^{44}\) The random initialization and the nonconvexity of GNN models along with the suboptimal optimization algorithm all contribute to achieving diverse results of the ensemble.\(^{27}\) The distribution of the outputs reflects the uncertainty in model predictions. The underlying concept is that input similar to the training data will result in similar output since the weights are optimized with regard to these inputs although they are different.\(^{27}\) Meanwhile, data not included in the training will result in more variance since they are more affected by the different sources of variance.\(^{27}\) The choice of the ensemble size “\(M\)” is usually limited by the training speed due to large datasets resulting in large training time. However, this is not the case in this study, and training speed is not a limiting factor, and thus \(M\) is set to 100.

#### 2.4.3 Last layer-dropout

Last layer dropout (LL-dropout) like other dropout variants, aims to perform Bayesian inference on the model parameters by adding a dropout layer on the activation before the last layer only.\(^{45,46}\) The dropout is kept active during evaluation and MC simulations are conducted to obtain a distribution of the predictions.\(^{33,45,47}\) The performance of the model at test time will highly depend on the dropout rate \(p\) chosen. Based on previously conducted studies, the dropout probability is set to 0.2 accordingly.\(^{33,37}\) At test time, the model is evaluated 2000 times.

### Table 3

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>MPNN</th>
<th>AFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial learning rate</td>
<td>([0.01, 0.02])</td>
<td>([0.01, 0.02])</td>
</tr>
<tr>
<td>Number of message passes (GNN layers)</td>
<td>([1, 2, 3, 4])</td>
<td>([1, 2, 3, 4, 5])</td>
</tr>
<tr>
<td>The output size of the GNN</td>
<td>([16, 32, 64])</td>
<td>([16, 32, 64, 128])</td>
</tr>
<tr>
<td>Hidden edge feature size</td>
<td>([16, 32, 64])</td>
<td>N.A.</td>
</tr>
<tr>
<td>Number of Set2set steps</td>
<td>([1, 2])</td>
<td>N.A.</td>
</tr>
<tr>
<td>Number of Set2set layers</td>
<td>([1, 2])</td>
<td>N.A.</td>
</tr>
<tr>
<td>Number of timesteps (number of graph attention layers)</td>
<td>N.A.</td>
<td>([1, 2, 3, 4, 5])</td>
</tr>
</tbody>
</table>

FIGURE 2  Distributions of \(P_c\), \(T_c\), and \(V_c\) target values

TABLE 3  List of tuned hyperparameters tuned for each model and their search domain

AOUICHAOUI ET AL.  5 of 15
2.5  |  Training protocol, hyperparameter selection, and evaluation metrics

2.5.1  |  Evaluation metrics

The model performance/quality is evaluated by calculating the mean absolute percentage error (MAPE), the root mean squared error (RMSE), and the coefficient of determination ($R^2$) for which the formulas can be seen in the Supporting Information. The quality of the model uncertainty is evaluated using the concept of coverage (the fraction of data for which the true target [experimental value] is within the 95% confidence interval [CI]).

2.5.2  |  Data split

The performance of data-driven models is highly dependent on the data used during the training step. It is therefore important to expose the model to most if not all available features in the dataset. This issue becomes more relevant in applications with a small dataset and relatively large input features. To address this issue, we train an initial GNN with a simple architecture using different data splits and evaluate the performance of the model. The data are split 200 different ways in ratios of 80%, 10%, and 10% for training, validation, and testing, respectively. This is done in order to prevent data leakage and create overly optimistic models.

FIGURE 3  |  Boxplot of MAE, RMSE, and $R^2$ for 200 runs of random data splits for $P_c$ (top row), $T_c$ (middle row), and $V_c$ (bottom row)
The split resulting in the best validation score is chosen as the split for the modeling.

### 2.5.3 Early stopping

Deep learning models are highly susceptible to overfitting and regularization techniques must be used. The model performance on the validation set is monitored in each epoch and the parameters resulting in the lowest loss are saved. If the performance does not improve after a given number of epochs (patience) the training is terminated and the best model is loaded. This approach is denoted “early stopping” and aims to terminate the training procedure before the model starts overfitting and the generalization performance (performance beyond the regression data) vis-à-vis the validation set deteriorates. The early stopping patience is set to 20 epochs.

### 2.5.4 Learning rate schedule

Adaptive learning rate is used to train all the models in this work. This makes it possible for the learning rate to adapt to the stage at which the optimization is. This procedure causes large changes at the start of the optimization resulting in learning good weights early and then refining them later with reduced learning rates. This will introduce various new hyperparameters to the training such as the reduction factor (between 1 and 0), the patience before reducing the learning rate. The initial learning rate is determined through a grid-search (see hyperparameter tuning), while the reduction factor is set to 0.5 and the patience is set to 10 epochs.

### 2.5.5 Hyperparameter tuning

Hyperparameters are highly influential for model performance. These are determined using a grid-search where every possible combination is evaluated. The search space is seen in Table 3. To leverage the model complexity and performance, Bayesian information criterion (BIC) is used to select the appropriate hyperparameters as it promotes less complex models (few parameters) and good model performance. The BIC is expressed in Equation (1), where “Nd” is the data sample size, “SSE” is the sum of squared errors produced by the model, and “Np” is the number of parameters in the model.

\[
\text{BIC} = N_d \cdot \log \left( \frac{\text{SSE}}{N_d} \right) + N_p \cdot \log (N_d)
\]

Important to note is that the regression task (the MLP) for both models is fixed to two fully connected layers with a hidden size corresponding to half the size output of the feature extraction.
section (GNN). The activation function chosen is the rectified linear unity (ReLU) function.

2.5.6 | Other training settings and implementation

The batch size used during training and validation corresponds to the full length of the training and validation set respectively, due to the low data size. Therefore, the exact gradient is not computationally cumbersome, which is typically the case in cases with significantly larger datasets (e.g., 10,000 s to millions).46 The maximum number of epochs was set to 500 epochs and the Adam optimizer was used46 with the MSE being the loss function. The work has been implemented using the PyTorch deep learning framework and the Deep Graph Library (DGL)51 for GNN modeling.

**FIGURE 4** Parity plots for $P_c$ (top), $T_c$ (middle), and $V_c$ (bottom) across all data folds (left: training, middle: validation, right: training) using the best message passing neural network (square) and attentive FP (triangle) models
3 | RESULTS AND DISCUSSION

3.1 | Effect of data split on the model performance

For each of the 200 random splits of data, an MPNN model was chosen with a single message passing step, output size of 32, a hidden edge size of 64, and a single layer - single step set2set readout. The distribution of the mean absolute error (MAE), the RMSE, and the $R^2$ along with the upper/lower quantiles and outliers are shown in Figure 3.

The average $R^2$ for the $P_c$ dataset is 0.95, 0.91, and 0.90 for the training, validation, and test set, respectively. Meanwhile, for the $V_c$ dataset, the average $R^2$ is calculated to 0.95, 0.92, and 0.90 for the training, validation, and test set, respectively over the 200 independent runs. As for the $V_c$ dataset, these have been determined to be 0.95, 0.90, and 0.84 for the training, validation, and test set, respectively. Despite the models achieving satisfactory $R^2$ values over the 200 runs, it is important to note that there is a significant spread and variance observed. This is due to the combined effect of data splitting and the random initialization of the weights. The overall spread seems to decrease with an increase in dataset size ($T_c$ being the largest dataset and exhibiting the least spread), however, it is unclear whether this can be generalized and should be investigated using a property dataset that is large enough to evaluate this. The same trends are observed for the remaining metrics (MAE and RMSE). The data split achieving the highest $R^2$ for the validation set was selected for the remainder of the study.

3.2 | Hyperparameter tuning

A total of 288 and 128 hyperparameter combinations were tested for respectively MPNN and AFP, respectively. The architecture with the lowest BIC on the validation set was chosen. The hyperparameters can be seen in Tables 4 and 5 for both models. The performance metrics can be seen in Tables 6 and 7 for MPNN and AFP, respectively.

Using the BIC promoted the usage of the smallest possible architecture in most cases (except for MPNN for $T_c$). Using BIC, not only promotes the least complex model, but as a consequence, it also speed-up subsequent uncertainty quantification methods due to a lower number of parameters.

The parity plots showcasing the agreement between the experimental data and the predictions for all data across the training, validation, and test set of the tuned models can be seen in Figure 4.

The results obtained from the GNN models are compared to the group-contribution-based model (GC) developed by Hukkerikar et al., which is a widely used model and is considered a state-of-the-art model with regards to performance and size of chemical design space it can describe. Hukkerikar et al. obtained $R^2$ values of 0.96, 0.99, and 0.99 for $P_c$, $T_c$, and $V_c$ respectively. These results indicate that GNN models can rival the performance of such models. It must however be noted that the size of the dataset used is different and that the rigorous concept of data splitting required for cross-validation is not used in GC-based studies in the literature. It is, however, expected that with increasing dataset size, the performance of GNN models will further improve given that the quality of data is not compromised.

3.3 | Uncertainty analysis

All methods for estimating the uncertainty rely on a population of models, therefore the mean and standard deviation of various performance statistics as the coverage for each method and each dataset are reported for the test data in Table 8. Meanwhile, Table 9 contains the same information but considers the entirety of the dataset.

Table 9 also reports the coverage obtained by the uncertainties produced by the GC model using maximum likelihood estimation (MLE) and the bootstrap method in order to compare the uncertainties of the two types of models (GC-based and GNN-based). For more details on these methods, the reader is referred to the work by Hukkerikar et al. and Frütićer et al. For more details on these methods, the reader is referred to the work by Hukkerikar et al. and Frütićer et al.

3.3.1 | Bootstrap

The optimal model resulting from the hyperparameter tuning was used as the reference model for the bootstrap method. The residual distribution plots of the reference model can be seen in the Supporting Information for MPNN and AFP, respectively. Using the residual errors produced by the two models on the training set, the 400 bootstrap samples are generated and the model is trained for each of these synthetic datasets. The 2.5% and 97.5% quantiles are determined for the output, which corresponds to the 95% CI. Figure 5 shows the 95% CI for the training, validation, and testing sets as well as the experimental target values for the $V_c$ dataset using the bootstrap method. It can be seen that the uncertainties produced by the AFP model are much narrower than those produced by the MPNN model. This is supported by the MAPE calculated, where the MAPE calculated for the AFP models is lower than those calculated by the MPNN. In particular, these difference between the model prediction error obtained by MPNN vs. AFP is pronounced for the estimation of $V_c$. This highlights the importance of estimation of uncertainty as an additional decision-making criterion in deep learning type model development. Similar observations are made with regards to the $T_c$ and $P_c$ dataset where the width of the confidence interval is narrower for the AFP model than for the MPNN as seen in the Supporting Information. This is further supported by the MAPE shown in Tables 8 and 9. Another important observation is that the width of the confidence interval (here is expressed in terms of the MAPE) seems to be positively correlated to the coverage, where a high coverage exhibits large MAPE and the low coverage exhibits low MAPE.

3.3.2 | Ensemble

The tuned hyperparameters are used to train the 100 members of the ensemble and the 95% CI is inferred from the construct distribution
### TABLE 8  Coverage and reported mean coefficient of determination \( (R^2) \), root mean squared error \( (\text{RMSE}) \), mean absolute percentage error \( (\text{MAPE}) \), and standard deviations (in parenthesis) for the testing set for all dataset and uncertainty methods

<table>
<thead>
<tr>
<th>Coverage</th>
<th>Bootstrap</th>
<th>Ensemble</th>
<th>LL-dropout</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( P_c )</td>
<td>( T_c )</td>
<td>( V_c )</td>
</tr>
<tr>
<td>MPNN</td>
<td>81</td>
<td>70</td>
<td>88</td>
</tr>
<tr>
<td>AFP</td>
<td>55</td>
<td>48</td>
<td>52</td>
</tr>
<tr>
<td>( R^2 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPNN</td>
<td>0.97 (0.01)</td>
<td>0.95 (0.02)</td>
<td>0.91 (0.05)</td>
</tr>
<tr>
<td>AFP</td>
<td>0.96 (0.01)</td>
<td>0.97 (0.01)</td>
<td>0.99 (0.01)</td>
</tr>
<tr>
<td>( \text{RMSE} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPNN</td>
<td>3.41 (0.55)</td>
<td>37.23 (6.09)</td>
<td>71.61 (18.67)</td>
</tr>
<tr>
<td>AFP</td>
<td>3.47 (0.39)</td>
<td>27.15 (2.02)</td>
<td>24.74 (2.49)</td>
</tr>
<tr>
<td>( \text{MAPE} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPNN</td>
<td>6.69 (0.83)</td>
<td>4.75 (0.79)</td>
<td>10.41 (2.64)</td>
</tr>
<tr>
<td>AFP</td>
<td>6.63 (0.71)</td>
<td>3.58 (0.28)</td>
<td>4.32 (0.37)</td>
</tr>
</tbody>
</table>

Note: GC\(^+\) refers to the model by Hukkerikar et al.\(^4\) The value reported in the parenthesis is the standard deviation of the metric. “-” indicates that either the metric has not been calculated or is not applicable. Value in “( )” is the standard deviation.

### TABLE 9  Coverage and reported mean coefficient of determination \( (R^2) \), root mean squared error \( (\text{RMSE}) \), mean absolute percentage error \( (\text{MAPE}) \), and standard deviations (in parenthesis) for the complete data for all dataset and uncertainty methods

<table>
<thead>
<tr>
<th>Coverage</th>
<th>Bootstrap</th>
<th>Ensemble</th>
<th>LL-dropout</th>
<th>MLE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( P_c )</td>
<td>( T_c )</td>
<td>( V_c )</td>
<td>( P_c )</td>
</tr>
<tr>
<td>MPNN</td>
<td>84</td>
<td>76</td>
<td>94</td>
<td>80</td>
</tr>
<tr>
<td>AFP</td>
<td>68</td>
<td>61</td>
<td>66</td>
<td>64</td>
</tr>
<tr>
<td>GC(^+)</td>
<td>60</td>
<td>54</td>
<td>63</td>
<td>-</td>
</tr>
<tr>
<td>( R^2 )</td>
<td>0.98 (0.01)</td>
<td>0.96 (0.02)</td>
<td>0.96 (0.03)</td>
<td>0.98 (0.01)</td>
</tr>
<tr>
<td>GC(^+)</td>
<td>0.98 (0.01)</td>
<td>0.98 (0.01)</td>
<td>0.99 (0.01)</td>
<td>0.98 (0.01)</td>
</tr>
<tr>
<td>( \text{RMSE} )</td>
<td>2.80 (0.25)</td>
<td>34.20 (5.81)</td>
<td>42.50 (13.27)</td>
<td>2.31 (0.26)</td>
</tr>
<tr>
<td>GC(^+)</td>
<td>2.50 (0.23)</td>
<td>28.45 (1.25)</td>
<td>14.52 (1.16)</td>
<td>2.25 (0.24)</td>
</tr>
<tr>
<td>( \text{MAPE} )</td>
<td>5.99 (0.81)</td>
<td>4.52 (0.82)</td>
<td>8.41 (2.92)</td>
<td>4.51 (0.92)</td>
</tr>
<tr>
<td>GC(^+)</td>
<td>5.27 (0.56)</td>
<td>3.86 (0.18)</td>
<td>3.60 (0.35)</td>
<td>4.66 (0.64)</td>
</tr>
</tbody>
</table>
for each datapoint. The AFP model produces overly confident models as the confidence interval is much narrower than that of MPNN as seen for $V_c$ in Figure 6. However, compared to the bootstrap, a low coverage does not always correspond to a high MAPE as seen in Tables 8 and 9 for $P_c$ as the coverage for AFP (64%) is higher than that of MPNN (80%) despite having a higher MAPE (6.12% vs. 5.93%). The corresponding visualization of the confidence intervals can be seen in the Supporting Information. The same trend for $V_c$ as observed using the bootstrap is also observed when using the ensemble: the MPNN model results in the widest confidence intervals and also the highest coverage, whereas AFP has a much narrower confidence interval and a corresponding lower coverage. The same holds true also for $T_c$ as seen in the Supporting Information.

3.3.3 | LL-dropout

The combination of LL-dropout and AFP produces the largest coverage across datasets achieving coverage of 90%, 81%, and 94% for $P_c$, $T_c$, and $V_c$, respectively as seen in Tables 8 and 9. This also corresponds to the highest MAPE achieved for the AFP. As for the MPPN, the LL-dropout produces the lowest coverage across datasets although the MAPE values are the highest observed across the data (except for $T_c$ when using bootstrap). Depiction of the confidence of the model for $V_c$ using LL-dropout can be seen in Figure 7. The remainder of the CI visualization can be found in the Supporting Information.

3.4 | Benchmarking and comparison

The coverage achieved by all GNN models regardless of the uncertainty quantification technique used exceeds that obtained through bootstrap for the benchmark model GC+\. As for the remainder of the models, techniques, and datasets, it is different from case to case. Using MPNN, the highest coverage achieved for $T_c$, $P_c$, and $V_c$ is through the bootstrap. However, when using the AFP model, LL-dropout achieved the highest coverage among all techniques. In fact, the combination of using the AFP model with LL-dropout achieves the highest coverage for all datasets. However, it must also be noted that for small data applications, variance introduced from the random initialization, early stopping and the stochastic gradient descent algorithm becomes more prominent compared to applications with much bigger data sizes. Therefore, even these results obtained might contain some small degrees of uncertainty. Better coverage does not necessarily mean a “better” model or more “reliable” technique, but it would potentially have a higher chance of covering the real experimental value within its interval. Furthermore, while coverage indicates the fraction of true values within the uncertainty

**FIGURE 5** Target experimental values of $V_c$ (dots) and the calculated 95% CI (shaded area) using bootstrap for all data splits using message passing neural network (top) and attentive FP (bottom)
bound, it is also important to consider the width of the uncertainty bounds especially if the intended use is to perform MC simulation using these properties and their uncertainty as input to a mathematical model (e.g., EOS).

There is an increasing interest in developing and evaluating the feasibility of emerging machine learning, especially deep learning, applications in the field of property prediction for pure components as well as drug design and discovery to materials design for fuel cells/batteries among others. As the field is rapidly progressing with new models and algorithms, establishing some degree of empirical rigor to ensure consistent performance assessment and comparison becomes an important point of concern. Within this context, in this contribution, we focused on evaluating and comparing prediction uncertainty of the deep learning models applied for property prediction problems, which has important implications for many model-based engineering applications. The first observation that needs to be emphasized is that deep learning models work with relatively smaller datasets. Indeed, there is a common perception in the chemical engineering field that deep learning requires large datasets for proper training and testing, which is indeed the case in many applications such as image analysis. As our study shows this perception is not necessarily valid. One of the reasons that help deep learning to work with the smaller dataset is that there is the flexibility of incorporating domain-specific knowledge into modeling. In this case, molecular chemistry knowledge is used when defining features for representing the two-dimensional structure of the molecules. Seen from this perspective, graph neural networks offer an alternative method to generate features as input to modeling structure–property relationships. In fact, domain knowledge is also used in GC property models a commonly used modeling approach for estimating pure component critical properties. The main difference between the two modeling approaches is among others the way such features are defined. While GNN offers to automate this procedure, in GC modeling studies, there is a lack of a systematic approach to defining/generating higher-order groups in the molecules, which are necessary to distinguish various isomers from each other (especially valid for the third-order functional groups defined in GC").

In our second observation, despite using established protocols for training, testing, and hyperparameter tuning as suggested in the community, uncertainty is present in the model predictions which is also observed for many deep learning applications. The degree of uncertainty for example, confidence intervals and coverage of measured data in these intervals differ among different uncertainty quantification methods as well as different models. This has been the case also in comparison to uncertainty quantification methods for general applications of deep learning. It is important that each model uncertainty prediction is benchmarked with a couple of uncertainty estimation methods. Important to note is also that in terms of the GCN models, it is not possible to separate the uncertainty in the feature extraction task from that of the regression task because these two tasks are

FIGURE 6 Target experimental values of $V_c$ (dots) and the calculated 95% CI (shaded area) using ensemble for all data splits using message passing neural network (top) and attentive FP (bottom)
unified in an end-to-end learning approach and are interlinked through the backpropagation. Furthermore, the latent representation of the compounds (the output of the feature extraction task) is not unique from model to model and will depend on various factors such as the hyperparameters, the architecture, and the algorithms used to produce the latent representation (including the update and readout functions).

Going forward for process systems engineering applications in general and property estimation in particular, the choice of which modeling approach, be it a traditional GC approach or a more sophisticated deep learning model, should be determined by the purpose of the engineering application. For screening/feasibility assessment of chemicals such as the work presented by Frütiger et al.\(^1\) and Cignitti et al.,\(^12\) a model with larger coverage is to be preferred such as the MPNN model with the bootstrap method for uncertainty estimation or the AFP model with LL-dropout (as uncertainty may be large but we are assured the predictions are aligned with experimental data collected thus far). However, for applications requiring detailed design and optimization where the model uncertainty and accuracy are more important,\(^56\) a property model that can provide higher accuracy/lower uncertainty with affordable computational cost, is preferred such as the case with a GC model. The LL-dropout method usually underestimates uncertainty which leads to poor coverage in testing data (e.g., the MPNN model). The latter is indeed the most important aspect in predictive applications (e.g., the MPNN model).

The bootstrap technique has a solid theoretical background\(^43\) based on the central limit theorem, which states that as sample N (here in MC sampling) increases, one converges to an estimation of the “true mean” of predictions (here property model) with error rate inversely proportional to sampling (error = \(\sigma / \sqrt{N}\)). As such the theory can be applied/has been applied to many prediction problems across scientific disciplines covering both small and large datasets/observations. However from the practical implementation of the bootstrap technique, one needs to take care that both the sampling number and the reference model are properly tuned to represent a good model of the underlying data. A good modeling practice is to use proper optimization techniques (e.g., grid-search or Bayesian optimization) for the hyperparameter tuning of the machine learning model to avoid common pitfalls such as bias toward data selection or poorly trained models. It is important to remark that sampling of data for model training and testing can condition the model performance. To avoid this bias, it is imperative that one repeats the cross-validation several times using random splitting of data (e.g., 100) and choose the optimal model accordingly. With this established good practice, one can avoid such pitfalls as bias in model training and data sampling. In summary, the bootstrap method is also our recommendation as to the preferred uncertainty estimation method especially to propagate measurement

---

**Figure 7** Target experimental values of \(V_c\) (dots) and the calculated 95% CI (shaded area) using LL-dropout for all data splits using message passing neural network (top) and attentive FP (bottom)
error/uncertainty in deep learning-based models. As a final remark, the systematic study of feature importance and interpretability of graph neural network models remain an important and relevant challenge. This requires continuous research efforts and development, which is the current focus of research throughout the machine learning community such as ablation studies or embedding domain-specific knowledge.

4 | CONCLUSION

In this study, two GNN models have been successfully applied to model the critical temperature, critical pressure, and critical volume for various organic compounds, respectively. The model hyper-parameters were determined through a systematic approach using grid-search. The obtained results rival those obtained from more classical QSPR methods such as the GC models. This outcome is considered promising especially seen from the perspective that, not all available data are used for training the GNNs, whereas all the data is used for training the GC models. Furthermore, the uncertainty related to the models has been quantified using three different approaches: the bootstrap method, the ensemble method, and the LL-dropout method. The degree of coverage is highly dependent on the model used, the technique applied as well as the data used for regression, which is in agreement with observation in other fields that are more mature in the application of deep learning models such as image recognition. However, for all datasets, models, and techniques used, the model performance and coverage of deep learning models either rival or exceed those produced by the GC models used as a benchmark for comparison. This also demonstrates that deep learning performs well even in applications characterized by scarce data. On the other hand, the choice of the model and uncertainty estimation technique depends on the end of use application and whether the aim is to have the largest coverage or have the best mean prediction. Overall appropriate benchmarking and comparison of different model alternatives with relevant metrics including also prediction uncertainty should be mandatory for deep learning models in both research and application tasks.

AUTHOR CONTRIBUTIONS

Adem R N. Aouichaoui: Conceptualization (lead); data curation (lead); methodology (equal); software (lead); validation (lead); visualization (lead); writing – original draft (lead); writing – review and editing (equal). Seyed Soheil Mansouri: Funding acquisition (equal); project administration (equal); supervision (equal); writing – review and editing (equal). Jens Abildskov: Funding acquisition (project); project administration (equal); supervision (equal); writing – review and editing (equal), Gürkan Sin: Conceptualization (lead); funding acquisition (lead); methodology (equal); project administration (equal); supervision (equal); writing – review and editing (equal).

DATA AVAILABILITY STATEMENT

Research data are not shared.

ORCID

Adem R N. Aouichaoui https://orcid.org/0000-0002-3297-6054
Seyed Soheil Mansouri https://orcid.org/0000-0002-0924-4947
Jens Abildskov https://orcid.org/0000-0003-1187-8778
Gürkan Sin https://orcid.org/0000-0003-0513-4502

REFERENCES